KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNŞAAT MÜHENDİSLİĞİ ANABİLİM DALI

YAPI BİLGİ MODELLEMESİ VE MAKİNE ÖĞRENİMİNE DAYALI OTOMATİK OLARAK ELDE EDİLEN İŞ PROGRAMLARI ARACILIĞIYLA DÖRT BOYUTLU MODELLERİN GELİŞTİRİLMESİ

DOKTORA TEZİ

Ümit BAHADIR

MART 2025
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

İNŞAAT MÜHENDİSLİĞİ ANABİLİM DALI

YAPI BİLGİ MODELLEMESİ VE MAKİNE ÖĞRENİMİNE DAYALI OTOMATİK OLARAK ELDE EDİLEN İŞ PROGRAMLARI ARACILIĞIYLA DÖRT BOYUTLU MODELLERİN GELİŞTİRİLMESİ

Ümit BAHADIR

ORCID : - - -

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"DOKTOR (İNŞAAT MÜHENDİSLİĞİ)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 10 / 03 /2025
Tezin Savunma Tarihi : 26 / 03 /2025

Tez Danışmanı : Prof. Dr. Vedat TOĞAN
ORCID : - - -

Trabzon 2025

ÖNSÖZ

Bu çalışma, Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü İnşaat Mühendisliği Anabilim Dalında Doktora Tezi olarak hazırlanmıştır. Tezin her aşamasında destek ve teşvikleri ile beni motive eden, öğrencisi olmaktan her zaman onur ve gurur duyacağım değerli hocam Prof. Dr. Vedat TOĞAN'a sonsuz teşekkürlerimi bir borç bilirim.

Doktora tez izleme komitesi ve aynı zamanda jüri üyesi kıymetli hocalarım Doç. Dr. Sibel MAÇKA KALFA ve Doç. Dr. Hasan Basri BAŞAĞA'ya eleştiri ve önerilerinden dolayı teşekkür ederim. Tez savunma sınavı jüri üyeliğini kabul eden değerli hocalarım Prof. Dr. Beliz ÖZORHON ve Doç. Dr. Onur Behzat TOKDEMİR'e katkılarından dolayı teşekkür ederim.

Tez çalışmalarım sırasında bana destek olan ve yardımlarını esirgemeyen Prof. Dr. Tayfun DEDE, Dr. Öğr. Üyesi Osman Tuğrul BAKİ, Dr. Fatemeh MOSTOFİ, Öğr. Gör. Dr. Ediz BOZ, Arş. Gör. Dr. Sebahat ŞİMŞEK ve Arş. Gör. İpek Naz SEMERCİOĞLU'na ayrıca teşekkürlerimi sunarım.

Bugünlere gelmemde ellerinden gelen tüm imkanları sağlayan, özellikle hayatımın bu önemli aşamasında desteklerini eksik etmeyen, haklarını hiçbir zaman ödeyemeyeceğim değerli aileme minnet ve şükranlarımı sunmayı bir borç bilirim. Bu çalışma boyunca desteğini ve hoşgörüsünü benden esirgemeyen sevgili eşim Meltem BAHADIR'a ve güler yüzüyle bana her zaman moral veren kızım Asel Defne BAHADIR'a sonsuz sevgilerimi sunarım. Bu çalışmanın, ülkemizin bilimsel alanda ilerlemesine ve yeni yapılacak olan çalışmalara katkıda bulunmasını temenni ederim.

Ümit BAHADIR

Trabzon 2025

TEZ ETİK BEYANNAMESİ

Doktora Tezi olarak sunduğum “Yapı Bilgi Modellemesi ve Makine Öğrenimine Dayalı Otomatik Olarak Elde Edilen İş Programları Aracılığıyla Dört Boyutlu Modellerin Geliştirilmesi” başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Vedat TOĞAN’ın sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 26/03/2025

Ümit BAHADIR

İÇİNDEKİLER

Sayfa No

Doktora Tezi

ÖZET

YAPI BİLGİ MODELLEMESİ VE MAKİNE ÖĞRENİMİNE DAYALI OTOMATİK OLARAK ELDE EDİLEN İŞ PROGRAMLARI ARACILIĞIYLA DÖRT BOYUTLU MODELLERİN GELİŞTİRİLMESİ

Ümit BAHADIR

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
İnşaat Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Vedat TOĞAN
2025, 207 Sayfa, 32 Sayfa Ek

İnşaat projelerinde iş programlarının oluşturulması, genellikle deneyimli planlayıcılar tarafından manuel yöntemlerle yürütülmekte olup, bu yöntem yoğun emek harcanan, zaman alıcı ve hata riski yüksek bir süreç olarak tanımlanmaktadır. Çeşitli çözüm yöntemleri geliştirilmiş olsa da çoğu proje hala manuel iş akışlarıyla yürütülmektedir. Bu durum, sektörün dijitalleşme ve otomatik çözümlere olan ihtiyacını ortaya koymaktadır. Tez kapsamında, iş programı hazırlama sürecini hızlandırmak ve hataları azaltmak amacıyla Yapı Bilgi Modellemesi (BIM) ve makine öğrenimi (ML) yöntemleri bütünleşik bir çerçevede ele alınmıştır. 24 projeden derlenen 1390 aktivite üzerinde altı farklı ML modelleriyle aktivite-ardıl ilişkileri tahmin edilmiş, parametre optimizasyonu sonucunda en yüksek doğruluk oranları elde edilen modeller seçilmiştir. Bu veriler, Revit ortamına entegre edilen "BIM Auto Scheduling" eklentisiyle tez kapsamında geliştirilen Dynamo kodlarını otomatik olarak çalıştırarak iş programının üretilmesini sağlamaktadır. Ardından, Navisworks ortamında geliştirilen "BIM Auto Scheduling – Import CSV" eklentisi ile aktivite-yapı elemanı eşleştirmeleri otomatikleştirerek 4B simülasyonun daha hızlı ve hatasız bir şekilde oluşturulması gerçekleştirilmiştir. Önerilen model, inşaat projeleri için basit ve kullanıcı dostu bir otomatik iş programı oluşturma aracı olarak öne çıkmaktadır. Bu yaklaşım, manuel planlamanın yol açtığı zaman kaybı ve hata risklerini en aza indirerek, proje paydaşlarına kapsamlı bir iş planı sunmaktadır.

Anahtar Kelimeler: Yapı Bilgi Modellemesi, Makine öğrenimi, Otomatik iş programı, 4B simülasyon, Revit eklentisi, Navisworks eklentisi

Ph.D. Thesis

SUMMARY

DEVELOPMENT OF FOUR-DIMENSIONAL MODELS THROUGH AUTOMATICALLY GENERATED CONSTRUCTION SCHEDULES BASED ON BUILDING INFORMATION MODELING AND MACHINE LEARNING

Ümit BAHADIR

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Civil Engineering Graduate Program
Supervisor: Prof. Dr. Vedat TOĞAN
2025, 207 Pages, 32 Pages Appendix

In construction projects, schedule preparation is generally carried out manually by experienced planners, a method that is labor-intensive, time-consuming, and prone to errors. Although various solutions have been developed, most projects still rely on manual workflows, underscoring the industry need for digitalization and automated solutions.

Within the scope of this thesis, an integrated framework combining Building Information Modeling (BIM) and machine learning (ML) techniques is proposed to accelerate the schedule preparation process and reduce errors. Predecessor-successor relationships were predicted on 1,390 activities compiled from 24 projects using six different ML models, with the highest accuracy models selected through parameter optimization. These data are then used to automatically generate a schedule by executing the Dynamo codes developed in this thesis via the “BIM Auto Scheduling” add-in integrated into the Revit environment. Subsequently, activity-element matching is automated in Navisworks using the “BIM Auto Scheduling – Import CSV” add-in, enabling faster and more accurate creation of a 4D simulation. The proposed model emerges as a simple, user-friendly tool that minimizes the time loss and error risks of manual planning while providing stakeholders with a comprehensive project schedule.

Key Words: Building Information Modeling, Machine learning, Schedule automation, 4D simulation, Revit add-in, Navisworks add-in

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER VE KISALTMALAR DİZİNİ

AEC : Mimarlık, Mühendislik ve İnşaat (Architecture, Engineering and Construction)
AI : Yapay Zeka (Artificial Intelligence)
ANN : Yapay Sinir Ağları (Artificial Neural Network)
ARM : Birliktelik Kural Madenciliği (Association Rule Mining)
BFS : Genişlik Öncelikli Arama (Breadth-First Search)
BIM : Yapı Bilgi Modellemesi (Building Information Modeling)
CAD : Bilgisayar Destekli Tasarım (Computer-Aided Design)
CBR : Vaka Temelli Akıl Yürütme (Case-Based Reasoning)
CNN : Evrişimsel Sinir Ağlar (Convolutional Neural Networks)
CPM : Kritik Yol Yöntemi (Critical Path Method)
ÇŞİDB : Çevre, Şehircilik ve İklim Değişikliği Bakanlığı
DFS : Derinlik Öncelikli Arama (Depth First Search)
DL : Derin Öğrenme (Deep Learning)
DNN : Derin Sinir Ağları (Deep Neural Networks)
ELU : Üstel Doğrusal Birim (Exponential Linear Unit)
FN : Yanlış Negatif (False Negative)
FP : Yanlış Pozitif (False Positive)
FS : Bitiş-Başlangıç (Finish-to-Start)
GA : Genetik Algoritmalar (Genetic Algorithms)
GAN : Üretici Çekişmeli Ağlar (Generative Adversarial Networks)
GAS : Grafik Tabanlı Otomatik İş Programı (Graph-Based Automated Scheduling)
GAT : Grafik Dikkat Ağlar (Graph Attention Networks)
GCN : Grafik Evrişimli Ağlar (Graph Convolutional Networks)
GNN : Grafik Sinir Ağlar (Graph Neural Networks)
IPD : Entegre Proje Teslimi (Integrated Project Delivery)
KBS : Bilgi Tabanlı Sistemler (Knowledge-Based Systems)
LLM : Büyük Dil Modeli (Large Language Model)
LOD : Detay Seviyesi (Level of Development/Detail)
LSTM : Uzun Kısa Süreli Bellek (Long Short-Term Memory)

ML : Makine Öğrenimi (Machine Learning)
NLP : Doğal Dil İşleme (Natural Language Processing)
ReLU : Düzeltilmiş Doğrusal Birim (Rectified Linear Unit)
RNN : Tekrarlayan Sinir Ağlar (Recurrent Neural Networks)
SL : Sığ Öğrenme (Shallow Learning)
SVM : Destek Vektör Makineleri (Support Vector Machines)
TN : Doğru Negatif (True Negative)
TP : Doğru Pozitif (True Positive)
UI : Kullanıcı Arayüzü (User Interface)
WBS : İş Kırılım Yapısı (Work Breakdown Structure)
2B : 2 Boyut(lu)
3B : 3 Boyut(lu)
4B : 4 Boyut(lu)
5B : 5 Boyut(lu)
6B : 6 Boyut(lu)
7B : 7 Boyut(lu)

Not: Bu listede verilmeyen bazı semboller ve kısaltmalar metin içerisinde ilgili oldukları yerlerde tanımlanmışlardır.

1. GENEL BİLGİLER

1.1. Giriş

İnşaat sektörü, ulusal ve küresel ölçekte ekonomik ve toplumsal gelişime önemli katkılar sağlayarak (Biswas vd., 2021); altyapı, konut, endüstriyel tesisler ve diğer yapıların inşasıyla kalkınma süreçlerinde kritik bir rol üstlenmektedir (Günaydın Ünsal, 2022). Sektörde oluşan yoğun rekabet, kesin teslim tarihleri ve yüksek maliyet baskısı gibi faktörler inşaat projelerinin başarısını önemli ölçüde etkilemektedir (Wefki vd., 2024). İnşaat projelerinin başarılı bir şekilde yürütülmesi açısından, zaman, bütçe ve kalite unsurları temel performans göstergeleri olarak kabul edilmektedir (Project Management Institute, 2021). Bu unsurların eş zamanlı ve etkin bir biçimde yönetilmesi, sektördeki en büyük zorluklarından biridir (Mostofi, 2024). Projelerin artan ölçeği ve karmaşıklığına bağlı olarak özellikle büyük çaplı projelerde ortaya çıkan düşük verimlilik, zaman gecikmeleri, maliyet aşımları ve kalite problemleri gibi sorunlarla yüzleşilmeye devam edilmektedir (Meng, 2012; Yarnold vd., 2021; Hajirasouli vd., 2022; Blanco vd., 2023). Öte yandan bu sorunlar, paydaşlar arasındaki ilişkileri zora sokarak sektörde güven kaybına yol açabilmektedir (Aljebory & QaisIssam, 2019). Büyük projelerin sıklıkla hedeflenenden %20 daha uzun sürmesi ve bütçeyi %80'e kadar aşabilmesi, proje planlama ve kontrol süreçlerinin geliştirilmesi gerektiğini açıkça ortaya koymaktadır (Changali vd., 2015).

İş programları, zaman, bütçe ve kalite unsurlarını takip etmek için kullanılan en önemli araçlardan biri olarak öne çıkmaktadır (ElMenshawy & Marzouk, 2021). İyi hazırlanmış bir iş programı, proje ekibi ve paydaşlar arasında etkili bir iletişim aracı olarak görev yapmakta (Son vd., 2017) ve aynı zamanda kaynakların etkin kullanımı, ekip sayılarının belirlenmesi, proje verimliliğinin ölçülmesi ve nakit akışının yönetimi gibi projeyi başarıya ulaştırmak için kritik öneme sahip konulara da temel oluşturmaktadır (Jung vd., 2024). Bununla birlikte, inşaat planlamasının karmaşık ve çok aşamalı bir süreç olması, iş programlarının da hazırlık ve yönetim aşamasında çeşitli zorluklar yaratmaktadır (Al-Sinan vd., 2024).

Birbirleriyle ilişkili birçok aktiviteyi barındıran iş programları, geleneksel yaklaşım çerçevesinde manuel olarak hazırlandığında hem zaman alıcı bir süreç hem de hataya açık

bir yöntem haline gelmektedir (Al-Sinan vd., 2024; Tauscher vd., 2009). Çeşitli proje paydaşları tarafından sürekli güncellenmesi gereken iş programlarına dair veriler, sıklıkla kağıt tabanlı yöntemlerle veya manuel girişlerle işlenmekte, bu da iş programlarındaki tutarsızlığı ve hata riskini yükseltmektedir (Shourangiz vd., 2011; Sompolgrunk vd., 2022). Ayrıca, her yeni inşaat projesi için benzer süreçleri tekrarlayarak sıfırdan iş programı oluşturmak, özellikle büyük ölçekli ve karmaşık projelerde ciddi bir verimlilik kaybı yaratmaktadır (Sigalov & König, 2017). Düşük kaliteli veya eksik çizelgeler, proje aşamalarının belirsiz kalmasına, paydaşlar arasında iletişim kopukluklarına ve dolayısıyla sözleşme anlaşmazlıklarına kadar uzanan bir dizi soruna sebep olabilmektedir (Singh vd., 2023). Uzman proje paydaşlarının yoğun çabasıyla yürütülen iş programı oluşturma aşaması; faaliyetlerin sıralanması, sürelerin belirlenmesi ve kaynak ihtiyaçlarının tanımlanması gibi çok sayıda karmaşık detay içermektedir (Amer & Golparvar-Fard, 2021). Etkili bir inşaat sürecini planlamak için iyi geliştirilmiş bir iş programı gerekli olup, programların çok sayıda aktivite barındırabildiğinden manuel olarak oluşturulması hataya davetiye çıkarmaktadır (Mikulakova vd., 2010). Teknolojik gelişmelerin hız kazanması ve mevcut ticari yazılımlara rağmen, iş programlarının halen büyük ölçüde manuel yöntemlerle hazırlanması ve sürekli güncel tutulabilmesi için yoğun manuel girdi gerektirmesi, sektörün verimliliği olumsuz etkilemekte ve dijital dönüşümünü zorlaştırmaktadır (Kim vd., 2013; Saini vd., 2017; Amer vd., 2021).

Son yıllarda inşaat sektöründe iş programları ve planlama süreçlerine ilişkin çeşitli çalışmalar ve teknolojik gelişmeler ortaya konmuş olmasına rağmen, büyük projelerin %98'inin hâlâ ortalama %30 maliyet aşımlarına ve %40 oranında program gecikmelerine maruz kaldığı belirtilmektedir (Hong vd., 2023). Bu nedenle, planlama sürecinin hızlandırılması ve daha güvenilir bir iş programı oluşturma ortamının sağlanması gerekmektedir. İnşaat projelerinde başarısızlığa yol açan sebeplerin başında gelen zaman ve bütçe aşımları, geleneksel yöntemlerin yetersiz kaldığını ve teknolojinin etkin şekilde kullanımının kaçınılmaz hale geldiğini de göstermektedir (Al-Sinan vd., 2024).

İnşaat sektörünün dijital dönüşüm sürecinde Yapı Bilgi Modellemesi (Building Information Modeling, BIM), sektörün verimliliğini artırmak ve inşaat yöneticilerinin proje yönetim süreçlerine yönelik karar almalarını destekleme potansiyeliyle geliştirilen en önemli teknolojilerden biri olarak dikkat çekmektedir (Baniashemi vd., 2018; Asadi vd., 2019; Banihashemi & Zarepour Sohi, 2022; Alathamneh vd., 2024). BIM'in 3 boyutlu (3B) modelleme, veri paylaşımı, disiplinler arası iş birliği ve 4B/5B gibi ek boyutları

içerme yeteneği, inşaat planlamasında nesne temelli ve sayısal veri yönetimine olanak sağlamakta ve planlama süreçlerini iyileştirmek için yeni fırsatlar ortaya çıkartmaktadır (Kim vd., 2013; Miettinen & Paavola, 2014). Böylelikle, tek bir dijital platform üzerinden tasarım, planlama, uygulama ve bakım verilerinin paylaşılması mümkün hâle gelmekte ve proje paydaşları arasındaki veri alışverişi önemli ölçüde kolaylaşmaktadır (Jin vd., 2019; Sheikhkhoshkar vd., 2019; Crowther & Ajayi, 2021; Doukari vd., 2022). BIM ayrıca, metraj, kaynak tahsisi ve iş programı gibi kritik konuların daha şeffaf ve güncel bilgiler ışığında yönetilmesine imkân tanıyarak inşaatın her aşamasında daha doğru kararlar alınmasını sağlamaktadır (Elghaish & Abrishami, 2020).

BIM ve iş programlarının bütünleşik kullanımıyla elde edilebilecek 4B modelleme, tasarım ile iş programlarını eş zamanlı olarak yönetme avantajı sunmaktadır. Böylelikle, benzer projeler için otomatik veya yarı otomatik şekilde oluşturulan iş programlarından faydalanmak mümkün hâle gelebilmektedir (Amer vd. 2022). Bunun yanı sıra, BIM tabanlı iş programlarının, klasik manuel yaklaşımlara kıyasla daha esnek bir şekilde güncellenebildiği, bu sayede anlık değişikliklere hızlı ve güvenilir tepkiler verilebildiği de belirtilmektedir (Amer vd. 2022). Ancak, BIM'in yaygınlaşmasının önünde yüksek maliyet, yazılımların birlikte çalışabilirlik sorunları ve sektörün dönüşüme direnç göstermesi gibi engeller bulunmaktadır (Vidalakis vd., 2020; Wefki vd., 2024). Özellikle küçük ve orta büyüklükteki (KOBİ) inşaat firmalarının BIM kullanımının sınırlı olduğu, hatta planlama ve programlama süreçlerinde BIM araçlarına dair farkındalıklarının dahi düşük seviyede kaldığı belirtilmektedir (Rajabi vd., 2022; Al-Sinan vd., 2023).

Son yıllarda BIM uygulamalarının yanı sıra makine öğrenmesi (Machine Learning, ML) ve yapay zekâ (Artificial Intelligence, AI) algoritmaları da inşaat sektöründe giderek daha fazla önem kazanmaktadır. ML, geniş veri setlerinden öğrenme ve model oluşturma kabiliyetine sahip olması nedeniyle; proje süre tahmini, maliyet öngörüleri, iş sıralaması optimizasyonu ve proje risk analizleri gibi konularda yenilikçi çözümler üretmektedir (Salama & Moselhi, 2019; Wang & Rezazadeh Azar, 2019). Öte yandan parçacık sürü optimizasyonu (Particle Swarm Optimization) ve diğer sezgisel arama yöntemleri, karmaşık ve çok boyutlu proje planlama problemlerine uygulanarak, manuel yöntemlerin optimum çözümlerden uzak kalma riskini önemli ölçüde azaltmaktadır (Jung vd., 2024). Bu algoritmalar, proje yöneticilerine ağ sıkıştırma veya farklı senaryoları karşılaştırma konusunda güçlü bir araç sunmaktadır. Bu sayede, geniş bir çözüm uzayı hızlıca

taranabilmekte ve maliyet–süre dengesine yönelik en uygun yaklaşım elde edilebilmektedir (Elghaish & Abrishami, 2020).

ML ile desteklenen BIM tabanlı sistemler, otomatik veri çıkarma, iş paketlerinin oluşturulması ve iş programı optimizasyonu gibi konularda geleneksel yaklaşımlara kıyasla çok daha hızlı ve esnek çözümler sunarak, projenin tüm paydaşlarına gerçek zamanlı ve doğrulanabilir bilgiler sağlamaktadır (ElMenshawy & Marzouk, 2021). Ancak, pratik inşaat uygulamaları ile ilgili az sayıda araştırma bulunmaktadır (Patel, 2023).

Bu doğrultuda, inşaat sektöründe manuel iş programı oluşturmanın yol açtığı hatalar, zaman kaybı ve verimsizlik; BIM ve ML gibi teknolojilerle bütünleşik planlama modelleri geliştirilerek büyük ölçüde azaltılabilmektedir (Wefki vd., 2024). Sektörün hâlen düşük teknolojili süreçlere ve parçalara ayrılmış iş akışlarına bağımlı olması, teknolojik dönüşümü ve inovasyonu yavaşlatan bir unsur olarak öne çıkmaktadır. Ancak İnşaat 4.0 konseptinin getirdiği dijitalleşme dalgası ve entegre proje teslimi (Integrated Project Delivery, IPD) gibi yöntemlerin benimsenmesiyle, kapsamlı ve otomatikleştirilmiş çerçevelere duyulan ihtiyaç her geçen gün daha belirgin hâle gelmektedir (Al-Sinan vd., 2023). Bu doktora tez çalışması, söz konusu ihtiyaçlara yanıt vermek üzere, BIM tabanlı ve ML destekli bir iş programı otomasyon sistemi tasarlamayı ve böylece inşaat sektöründeki verimlilik, maliyet ve süre problemlerinin çözümüne katkıda bulunmayı amaçlamaktadır. Bu sayede, inşaat projelerinin daha sürdürülebilir, rekabetçi ve teknoloji odaklı bir yapıya kavuşması hedeflenmektedir.

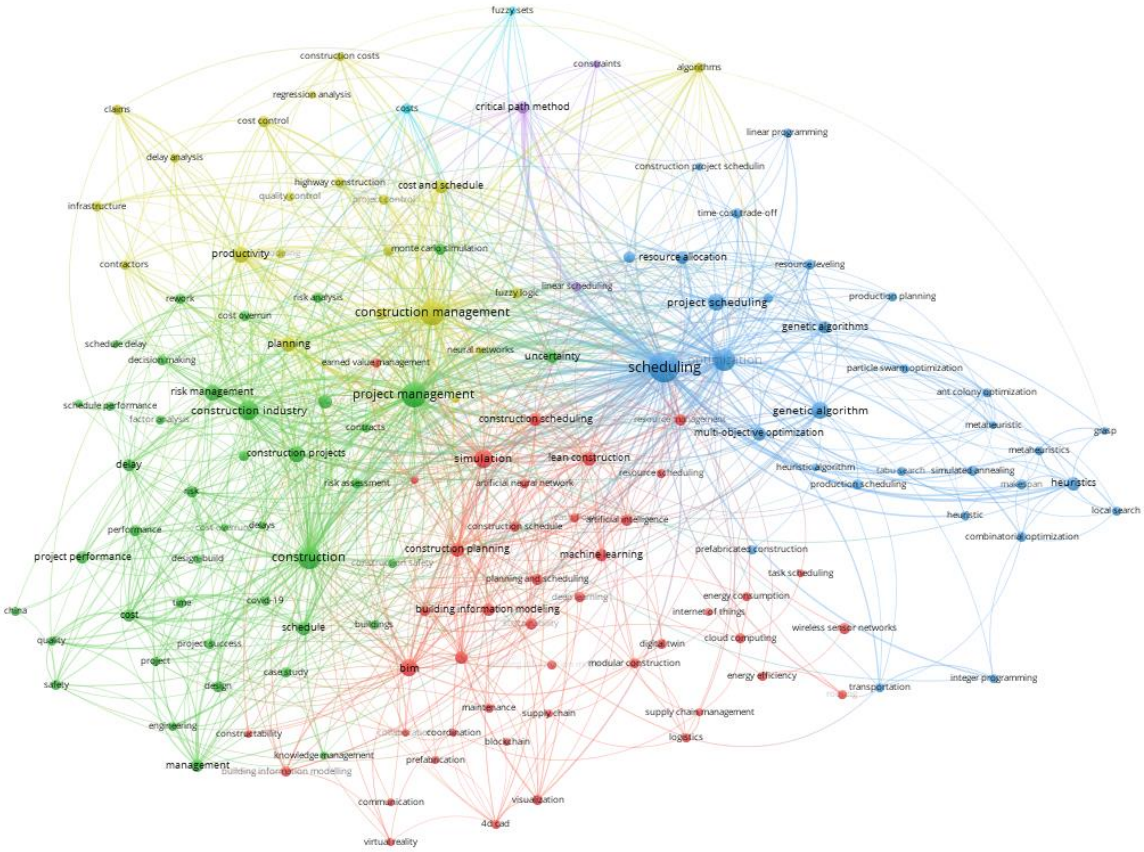
1.2. Literatür Taraması

Bu bölümde, tez çalışması kapsamında son yıllarda gerçekleştirilen çalışmalara, çalışmaların eksikliklerine ve sınırlamalarına yer verilmektedir. Planlama ve iş programı oluşturma süreçlerini kapsamlı biçimde irdelemek, birlikte ele alınan veya daha az çalışılan kavramları tespit etmek ve süreç genelinde hangi eksik ya da boşlukların ortaya çıktığını belirlemek üzere bir bibliyometrik analiz gerçekleştirilmiştir. Bu analiz, sürece etki eden çeşitli unsurların incelenmesine ve inşaat sektöründe ağırlıklı olarak benimsenen bakış açısını öne çıkarmaya olanak tanımıştır. Elde edilen veriler, planlama ve iş programı oluşturma süreçlerinin alt süreçlerini belirlemek ve bu süreçleri iyileştirmek için tez çalışması kapsamında önerilen stratejilerin geliştirilmesinde ve odaklanılan kavramların şekillenmesinde yol gösterici olmuştur.

Bibliyometrik analiz, yayınların özet, makale ve başlık kısımlarındaki kavramsal ilişkileri görünür kılarak araştırmacılara ayrıntılı bilgilere erişme olanağı sunmaktadır (Bankar & Lihitkar, 2019). Bu tez çalışmasında, uluslararası düzeydeki hakemli literatürü, geniş çapta özet bilgileri ve atıf veri tabanı sağlaması nedeniyle Scopus tercih edilmiştir (Martin-Martin vd., 2018). Analiz aşamasında yalnızca İngilizce dilinde yayımlanmış çalışmalar dikkate alınmıştır.

Analiz öncesinde, "construction schedule" ve "construction scheduling" anahtar sözcükleriyle, 2000 yılından başlanarak 2024 yılı Eylül ayına kadar Scopus'ta tarama yapılmıştır. Belge türü, "article" ve "review" olacak biçimde sınırlandırılmış ve yayınlara ait "alıntı bilgileri (citation information)", "bibliyografik bilgi (bibliographical information", "özet ve anahtar kelimeler (abstract & keywords)" verileri .csv formatında dışa aktarılmıştır. Metinsel veriler arası ilişkileri görselleştirme ve çok sayıda metni işleme kapasitesi nedeniyle VOSviewer (Van Eck & Waltman, 2011) kullanılmıştır. Bu yazılım, inşaat proje yönetimi, atık yönetimi, iş güvenliği ve BIM gibi alanlardaki benzer bibliyometrik analizlerde de yaygın biçimde tercih edilmektedir (He vd., 2017; Jin vd., 2019).

VOSviewer sürecinde, yazılımın tespit ettiği kavramlar arasından en az 20 kez yinelenenler esas alınmıştır. Bu sayede analiz sırasında, 20'den daha az yinelenen kavramlar ağ/görselleştirme analizine dahil edilmemiş ve böylece daha sık geçen ve örüntüyü güçlü biçimde yansıtan kavramlar üzerinde yoğunlaşılmıştır. Elde edilen sonuçlar Şekil 1'de gösterilmektedir. Bu görselleştirmede, farklı kavramlar yuvarlak düğümler üzerinden temsil edilmekte, söz konusu düğümler arasındaki ilişki çizgilerle ifade edilmektedir. Yuvarlakların boyutu ise incelenen anahtar kelimeler ile kurdukları yakınlığın derecesine göre artmakta ya da azalmaktadır.



Şekil 1. Bibliyometrik analiz çıktısı

Analiz sonucunda, incelenen makalelerdeki en çok kullanılan anahtar kelimelerin “scheduling”, “project management”, “construction management” ve “optimization” olduğu görülmüştür. İncelenen anahtar kelimeler ile sıkça birlikte geçen kavramlar da yine “scheduling”, “project management”, “construction management” ve “optimization” olarak öne çıkmaktadır. Buna karşın “machine learning”, “deep learning”, “construction safety” ve “building information modeling” gibi kavramların, anahtar kelimeler ile daha sınırlı bir ilişkiye sahip olduğu ve literatürde daha az ele alındığı tespit edilmiştir. Ayrıca 4B BIM’in de anahtar kelimelerle birlikte yer alan en yaygın kavramlar arasında bulunmaması, inşaat planlaması, BIM ve 4B modelleme arasındaki potansiyel araştırma boşluğunu ortaya koymaktadır.

Bibliyometrik analiz sonucunda, literatürde yer alan çalışmalar üç kısma ayrılmıştır. Birinci kısımda, BIM tabanlı iş programı oluşturan çalışmalar, ikinci kısımda 4B modelleme ve simülasyon oluşturulmasını inceleyen çalışmalar ve son kısımda ise AI tabanlı iş programı oluşturmaya odaklanan çalışmalardan bahsedilmektedir. Aşağıdaki bölümlerde sırasıyla, bu üç kategoride içerisinde yer alan çalışmalar detaylandırılmaktadır.

1.2.1. BIM Tabanlı İş Programı Oluşturma

Son on yılda, inşaat endüstrisinde BIM teknolojisinin kullanımında belirgin bir artış görülmektedir (ElMenshawy & Marzouk, 2021). Özellikle görselleştirme ve koordinasyonu artırmak amacıyla BIM kullanımına paralel olarak, iş programı oluşturma aşamasında da BIM'in giderek yaygınlaştığı ifade edilmektedir. Örneğin, Kim vd. (2013) IFC dosyası üzerinden yapı elemanlarına (döşemeler, duvarlar, kapılar, pencereler, asma tavanlar, çatı elemanları) ait bilgileri kullanarak aktivitelerin oluşturulması, sürelerin hesaplanması ve sıralama kurallarının uygulanmasına odaklanmıştır. Elde edilen iş programı MS Project ile ilişkilendirilmiş ve kullanıcının tercihlerine göre daha fazla geliştirme yapabileceği bir ortam sunulmaya çalışılmıştır. IFC'yi veri alışverişi için kullanan bir başka çalışmada (Büchmann-Slorup & Andersson, 2010), iş programı oluşturma sürecinin sağlam bir iş akışı veya iyi yapılandırılmış bir çerçeve olmaksızın, çoğunlukla uygulayıcının sezgisine ve geçmiş deneyimlerine bağlı yürütüldüğü vurgulanmıştır. Ayrıca, bu alanda BIM'in yeteneklerinin tam anlamıyla kullanılmadığı da belirtilmektedir. Benzer şekilde Faghihi vd. (2014)'de, iş programı oluşturmanın proje yöneticilerinin proje süresini takip etmek için kullandıkları en önemli araçlardan biri olduğunu vurgulamaktadır. Bununla birlikte, iş programlarının çok fazla zaman ve çaba harcanan ve esas olarak planlayıcının bilgisine ve önceki deneyimine bağlı olarak geleneksel bir şekilde oluşturulduğunu belirtmiştir. Ayrıca aynı çalışmada, genetik algoritma (Genetic Algorithms, GA) kullanılarak IFC dosyasından gerekli verileri çıkaran bir yaklaşım geliştirilmiştir. Ancak iki çalışmada da model çıktılarının MS Project veya Primavera P6 gibi proje yönetimi yazılımlarıyla ilişkilendirilmediği görülmektedir.

BIM tabanlı iş programı oluşturmaya yönelik başka bir yaklaşım, Park & Cai (2015) tarafından sunulmuş olup, dört ana aşamadan oluşan bir yöntemle inşaat iş programlarının otomatikleştirilmesi amaçlanmıştır. Bu aşamalarda önce, modelden nesne bilgileri (örneğin, nesne türü, alt seviye, üst seviye, genişlik vb.) elde edilmekte, ardından iş kırılım yapısı (Work Breakdown Structure, WBS) tanımlaması yapılarak sıralama kuralları ve her bir nesnenin ilişkisi belirlenmektedir. Daha sonra aktiviteler oluşturulmakta ve elde edilen iş programı MS Project ile uyumlu bir formata aktarılmaktadır. Wang & Rezazadeh Azar (2019)'da benzer biçimde, elemanların boyutları, miktarları, mekansal bilgileri, malzemeleri ve diğer ilgili nitelikleri dahil olmak üzere BIM'den veri çıkaran bir sistem geliştirmiştir. Proje iş paketlerini oluşturmak, sürelerini hesaplamak ve ilişkilerini

belirlemek için inşaat kurallarını, ön bilgileri ve üretim oranlarını kullanan araştırmacılar, sistemlerini betonarme çerçeveli binalar üzerinde test etmiş ve son aşamada MS Project'e aktarım gerçekleştirmiştir. ElMenshawy & Marzouk (2021) ise, benzer çalışmalardan farklı olarak aktiviteler arası ilişki (FS, SS, SF, FF) sayısını artırmış ve elde edilen verileri Primavera P6'ya aktarabilecek bir Excel dosyası elde etmiştir. Bununla birlikte, bu çalışma 3B modelde her kat için tek tip kolon kullanılması ve farklı kolon tiplerine izin verilmemesi, kiriş elemanlarını dikkate almaması ve manuel olarak tanımlanmış bir aktivite sıralamasına bağlı kalması gibi sınırlamalara sahiptir.

Önceki çalışmalar incelendiğinde, Primavera P6 ve MS Project gibi proje yönetimi yazılımları ile bağlantı kuran örneklerin mevcut olduğu görülmektedir. Bununla birlikte, Navisworks gibi inşaat endüstrisinde yaygın kullanılan BIM tabanlı proje yönetim araçları ile doğrudan bir entegrasyon sağlanmadığı dikkat çekmektedir. Söz konusu araştırmalar, BIM modellerinden proje planı oluşturma ve aktiviteleri planlama yeteneğine sahip olmalarıyla öne çıksa da BIM modellerinin otomatik iş programı oluşturmak amacıyla daha kapsamlı biçimde kullanılabileceği düşünülmektedir.

1.2.2. 4B Modelleme ve Simülasyon

Aktivitelerin sırasını ve model elemanlarıyla olan ilişkilerini görselleştirerek sanal simülasyon sağlamak amacıyla 3B modelin, proje iş planı ile entegre edilmesi sonucunda 4B modelleme yaklaşımı geliştirilmiştir (Koo & Martin, 2000). Bu yaklaşımla elde edilen 4B simülasyonlar, proje paydaşlarının anlayışlarını ve iletişimlerini geliştirmek (Balakina vd., 2018; Dawood & Sikka, 2009), iş programındaki eksik aktiviteleri ve mantıksal hataları tespit etmek (Koo & Martin, 2000), zaman-mekan çatışmalarını belirlemek (Akinci vd., 2002) ve kaynakları daha verimli bir şekilde kullanmak (Bansal, 2017) gibi konularda önemli bir gözlem ve analiz olanağı sunmaktadır. Ayrıca görsel incelemelerin yanı sıra, destekleyici algoritmalar yardımıyla 4B model, yapının dayanımı (Zhang & Hu, 2011), şantiye erişilebilirliği ve çalışma alanı güvenliği (Martinez-Aires vd., 2018) gibi konularda zamana bağlı analizler gerçekleştirmeye de imkân tanımaktadır. 4B modelleme ve simülasyonun sağladığı bu analizler ile projelerdeki gecikmelere, maliyet aşımlarına, verimlilik azalmasına ve güvenlik sorunlarına neden olabilecek olası sorunların önüne geçmek için inşaat öncesi erken denetim sağlanabilmektedir (Altun & Akçamete, 2019).

Günümüzde 4B modelleme genellikle iş programı ile başlamaktadır. Planlayıcı, uzmanlığını kullanarak 2B çizimlere veya 3B modele dayalı hesaplamalara göre bir iş programı geliştirmektedir. Ancak, 3B modelin ayrıntı düzeyinin iş programındaki ayrıntı düzeyiyle uyumlu olduğu bir 4B model oluşturmak için planlayıcı ve projecilerin (mimar, inşaat mühendisi, makine mühendisi vs.) koordineli çalışmaları gerekmektedir (Ciribini vd., 2016). Daha sonra, aktiviteler, model öğeleriyle manuel veya yarı otomatik adı verilen bağlantı kurma yöntemleri ile ilişkilendirilmektedir.

Manuel bağlama yönteminde, aktiviteler ve model öğeleri aynı ortamda toplanır ve her öğe bağlantılı aktivitesine tek tek manuel olarak atanır. Bu nedenle, bu işlem; tekrarlayıcı, zaman alan, hatalardan kaçınmak için özel dikkat gerektiren ve hangi öğenin hangi aktiviteyle bağlantılı olduğunu belirlemek için uzman deneyimine ihtiyaç duyulan bir süreç olmaktadır (Weldu & Knapp, 2012; Hamledari vd., 2017).

Yarı otomatik bağlantı yönteminde ise, modeldeki her bir eleman için bir aktivite kimliği parametresi oluşturulur ve elemanlar ilgili kimliklere birer birer atanır. Daha sonra model, simülasyon aracına aktarılır ve her bir aktivite kimliği için arama kümeleri oluşturulur ve elemanlar bu kimliklerine göre gruplandırılır. Ancak, bu yöntemde de kimliklere elemanların atanması ve arama kümeleri oluşturma aşamaları manuel gerçekleştirilmektedir. Bu bağlantı yönteminde, ilgili tüm aktivite kimliklerini oluşturmak için planlayıcı ve diğer proje paydaşları arasındaki koordinasyon oldukça önemlidir. Eksik veya yanlış aktivite kimliği atamaları nedeniyle yanlış 4B modeller oluşturulmasıyla sıklıkla karşılaşılmaktadır (Altun & Akçamete, 2019).

Bazı çalışmalar, bu manuel süreci kolaylaştırmaya yönelik yöntemler önermiştir. Örneğin, Park & Cai (2015), otomatik iş programı hazırlamayı kolaylaştırmak amacıyla BIM'in eleman kırılım yapısı ile iş programının iş kırılım yapısı arasında uyumluluk sağlayan bir yaklaşım sunmuş ve elemanlarla WBS listelerini bağlamıştır. Bu çalışmadan farklı olarak Ciribini vd. (2016), aktivitelerin bağlantı tanımlayıcısı olarak modeldeki elemanlara parametreler tanımlamıştır. Önceden tanımlanmış kurallar kullanılarak aktiviteler, elamanlarla otomatik olarak eşleştirilmiştir. Bu süreç, 3B model elemanlarının iş programındaki aktivitelerle manuel olarak bağlanması zorluğunu ortadan kaldırmaktadır; ancak parametreleri model elemanlarına atamak zaman gerektiren ve tekrarlayan doğası nedeniyle insan hatalarına neden olabilen bir süreç olmaktadır.

Kim & Cho (2015), iş programı geliştirmeyi desteklemek için model elemanları arasındaki öncül/ardıl bilgilerini, elamanların geometrik ilişkilerine göre eşleştiren

algoritmalar geliştirmiştir. Benzer bir şekilde Altun & Akçamete (2019), 3B model ile iş programını iki aşamada ilişkilendiren bir model oluşturmuştur. İlk aşama, model elamanlarına otomatik olarak atanan 4B aktivite kimliklerinin oluşturulmasını içermektedir. İkinci aşamada ise iş programı için aktivite kimliği listesi oluşturulmakta ve model öğelerini aktivitelerle eşleştirmek için arama kümeleri kullanılmaktadır. İş programı elde edildikten sonra MS Project ve Navisworks kullanılarak 4B simülasyon oluşturulmuştur; ancak çalışmada her bir model elemanına yalnızca bir aktivite kimliği atanmıştır. Aynı model elemanına (örneğin, duvar) birden fazla 4B aktivite kimliği (örneğin, tuğla, yalıtım, sıva, boya vb.) atanabilmesi için 4B aktivite kimliği oluşturma ve atama süreçlerinde iyileştirmelere ihtiyaç duyulmaktadır. Geliştirildikleri yöntemin bir diğer sınırlaması ise, yöntemin modellenmemiş elemanları (örneğin, kalıp, donatı) veya fiziksel olmayan görevleri (örneğin, inşaat malzemesinin temini) oluşturamamasıdır.

Fazeli vd. (2022), bütünleşik bir 4B BIM yaklaşımı geliştirerek özellikle GA'nın otomatik süre tahmininde en iyi performansı gösterdiğini ortaya koymuştur. Bu çalışmada, proje bileşenlerinin başlangıç ve bitiş tarihlerini otomatik hesaplayarak manuel süre hesaplamalarını ve insan kaynaklı hataları azaltmak hedeflenmektedir. Bununla birlikte, Navisworks Timeliner ile entegrasyon sırasında başlangıç/bitiş tarihleri, tatil günleri ve ilişki tablosunun kullanıcı tarafından manuel olarak girilmesi gerektiği belirtilmiştir. Ayrıca, yapı bileşeni olarak yalnızca mimari duvarlar, döşemeler ve asma tavanların dikkate alınması ve yöntemin daha çeşitli ve karmaşık projelerde henüz test edilmemiş olması, çalışmanın başlıca sınırlılıkları arasında yer almaktadır. Benzer biçimde, Wefki vd. (2024), CAD çizimleri veya manuel olarak hazırlanmış Excel dosyalarından elde ettikleri verileri kullanarak 3B BIM modelini oluşturmuş ve Navisworks aracılığıyla 4B modeli geliştirmiştir. Ancak, çalışmada yalnızca yapısal elemanlar dikkate alınmış ve proje kapsamı bu doğrultuda sınırlandırılmıştır.

Özetle, 4B modelleme ve simülasyon ile ilgili literatürdeki çalışmalar incelendiğinde, manuel süreçlerin fazlalığı ve buna bağlı olarak zaman alan ve hata riskinin yüksek olduğu bir sürecin varlığı dikkat çekmektedir. Çalışmaların sınırlamaları da iş programı oluşturma aşamasında eksik aktivitelere ve dolayısıyla 4B modelden hatalı simülasyon elde edilmesine sebep olmaktadır. Bu nedenle, daha kısa sürede ve daha doğru 4B modeller oluşturmak için literatürdeki çalışmaların iyileştirilmesi ve otomasyon düzeyinin artırılması gerektiği düşünülmektedir.

1.2.3. Yapay Zeka Tabanlı İş Programı Oluşturma

1980'lerin başlarından bu yana, inşaat sektöründe ve akademide iş programı hazırlama ve planlama görevini otomatikleştirmeye yönelik pek çok girişim olduğu bilinmektedir (Desgagné-Lebeuf vd., 2019). Buna rağmen, günümüzde dahi birçok proje hâlâ tamamen manuel iş programı oluşturma ve planlama iş akışlarıyla yürütülmektedir (Al-Sinan vd., 2024).

İnşaat sektöründe iş programını otomatikleştirmeye yönelik ilk girişimler, karmaşık sorunları çözmek, inşaat operasyonlarını modellemek ve analiz etmek için AI algoritmaları üzerine inşa edilen bilgi tabanlı sistemleri (Knowledge-Based Systems, KBS) kullanmıştır (Faghihi vd., 2015). Son yıllarda gerçekleştirilen, aktiviteler arasındaki sıralama mantığını belirlemeye yönelik araştırmalarda da (Wang & Rezazadeh Azar, 2019; Naticchia vd., 2019; Volk vd., 2018; Garcia-Lopez, 2017), manuel olarak önceden tanımlanan aktivite listeleri ve bunların bağımlılık ilişkileri temel alınmaktadır. Örneğin, bu sistemlerdeki girdi faaliyetleri okunurken, duvar boyama gibi bir aktivite, duvar inşa etme ve sıva yapma gibi önceki aktivitelere bağlanmaktadır.

Mevcut AI tabanlı iş programı hazırlama yöntemleri ve sistemlerinin çoğu, planlama bilgisinin manuel olarak oluşturulmasını gerektirmektedir. Her aktivite gösterimi, sıralama ilişkileri, süreç şablonu veya bileşen ilişkisi, uzmanlar tarafından manuel olarak tanımlanmalı ve sisteme girdi olarak verilmelidir (Jung vd., 2024). Tipik bir apartman binasıyla ilgili planlama yapmak için bile kolonlar, kirişler, döşemeler, duvarlar, şaftlar, mekanik borular, elektrik kabloları, kapılar, kapı kolları gibi planlama gerektiren yüzlerce yapı elemanı listelenebilir. Kalıp, donatı ve beton dökme veya duvar inşa etme, sıva ve boya gibi her bir inşaat elemanına birden fazla işlem de uygulanabilir. Bu elemanlar ve işlemler göz önüne alındığında, her elemanı, işlemi ve ilişkileri manuel ve açık bir şekilde modellemek, aşırı derecede maliyetli ve yoğun emek harcanan bir süreçtir (Amer vd., 2021).

Literatürde tanıtılan sistemlerin çoğu, önceki projelerden elde edilen en iyi iş programı uygulama bilgilerini otomatik olarak yapılandırmamakta ve yeniden kullanamamaktadır. Bir başka deyişle, geçmiş verilerden öğrenmeyi desteklememektedir. Mevcut yöntemlerdeki iş programlarının bilgi tabanlarında yapılacak herhangi bir değişikliğin manuel olarak sisteme tanıtılması gerekmektedir. El ile oluşturulan proje özellikleri aracılığıyla en iyi uygulama bilgilerini korumak ve yeniden kullanmak için vaka

temelli akıl yürütme (Case-Based Reasoning, CBR) yöntemlerini kullanarak geçmiş verilerden öğrenmeyi amaçlayan çalışmalar da bulunmaktadır (Hartmann vd., 2012; Mikulakova vd., 2010). Bu sistemler, her iş planlama durumunu, bir sorun ve önerilen bir çözüm olarak kaydederek bilgiyi korumaktadır. Yeni bir planlama problemi ile karşılaşıldığında, sistem yeni vaka ile veri tabanında saklanan vakalar arasındaki benzerliği değerlendirmekte ve benzerlik bulursa, çözümü yeni probleme uyarlamakta ve kullanıcıya sunmaktadır. Sistem ayrıca gelecekte başvurmak üzere yeni bir durum olarak yeni sorunu ve yeni çözümü de korumaktadır (Xu & Muñoz-Avila, 2008; Ryu & Lee, 2004). Bu tür yöntemlerde ve sistemlerde, bir planlama senaryosunu tanımlamak için özelliklerin ve benzerlik ölçülerinin belirlenmesi, gelecekte kullanılmak üzere veri tabanında depolanması ve yeni duruma uyacak şekilde otomatik olarak bir algoritma tasarlanması, bir insan planlayıcısının manuel müdahalesini gerektiren karmaşık görevlerdir (Kim vd., 2013).

İnşaat projelerinin karmaşıklığı sürekli artmakta, farklı sistemler ve bileşenler arasındaki etkileşimleri manuel olarak modellemek giderek daha çok zorlaşmaktadır (Jung vd., 2024). Doğal dil işleme (Natural Language Processing, NLP), ML ve veri madenciliği tekniklerindeki gelişmeler büyük miktardaki iş programı verisine manuel müdahaleyi ortadan kaldırmak için bir fırsat sunmaktadır (Amer vd., 2023; Torres-Calderon vd., 2019). Amer & Golparvar-Fard (2019), inşaat faaliyetlerinin bileşenleri hakkında otomatik olarak akıl yürüten ve aktivite etiketlemesi yapan derin öğrenmeye dayalı bir sistem geliştirerek bu yönde öncü adımlar atmışlardır. Gelişmiş sıralama ve zamanlama tahminleri için uzun kısa süreli bellek (Long Short-Term Memory, LSTM) ile birlikte tekrarlayan sinir ağlar (Recurrent Neural Networks, RNN) modellerini kullanmışlardır. Aynı araştırmacılar (Amer & Golparvar-Fard, 2021) bir başka çalışmalarında ise; mevcut iş programı verilerini denetimsiz ML yöntemleri ile eğiterek, %76 ve %98 arasında değişen doğrulukla aktivite sıralama bilgisi elde ettiklerini belirtilmiştir. Ancak bu çalışmada, aktiviteler arası farklı ilişki tipleri (FF, FS, SS, SF) göz ardı edilmiş ve tüm öncül/ardıl ilişkileri tek tip olarak incelenmiştir. Ayrıca, bina elemanlarının konum bilgileri de dikkate alınmamıştır ve konuma bağlı iş akışlarını yakalama ve öğrenme yeteneği sınırlandırılmıştır. Benzer bir çalışma, mevcut verilerden otomatik olarak inşaat görevlerinin bir ontolojisini çıkarmak için geleneksel sözdizimsel ayrıştırma tekniklerini kullanan Zhao vd. (2020) tarafından da yapılmıştır. Ayrıca, Jung vd. (2024), inşaat zamanlaması faaliyetlerini otomatik olarak ASTM UniFormat II kategorileriyle eşleştirebilen dönüştürücü (transformer) tabanlı bir NLP modeli (UniformatBridge) geliştirmiştir. Bu model ile kısıtlı inşaat sıralaması

bilgisini ASTM Uniformat II Seviye 2 ve Seviye 3 sınıflarında sırasıyla %93 ve %87'lik F1 puanıyla elde etmiştir. Bu sayede, otomatik 4B BIM oluşturma ve maliyet-ödeme uygulamaları gibi iş akışlarını kolaylaştırarak manuel kodlama ihtiyacının en aza indirilmesi amaçlanmıştır.

Sonuç olarak, AI temelli otomatik iş programı oluşturma yöntemlerinde, mevcut verileri analiz eden ve insan girdisi olmadan inşaat bilgisini otomatik olarak öğrenebilen sistemlerin henüz tam olarak geliştirilemediği ve gerçek yaşam projelerinde yeteri sayıda test edilmediği görülmektedir. Mevcut literatür incelendiğinde, derin öğrenme (Deep Learning, DL) modellerinin sıralı ve zamansal desenleri etkili bir şekilde yakalayabilmek için genellikle büyük veri kümelerine ihtiyaç duyduğu ve bu nedenle daha çok büyük ölçekli projelere odaklanan çalışmaların yer aldığı anlaşılmaktadır. Bununla birlikte, küçük ve orta ölçekli inşaat projelerine yönelik benzer çalışmaların sayısının sınırlı olduğu ve bu alanda kapsamlı araştırma eksikliği bulunduğu gözlenmektedir. Bu kapsamda, gelecekteki araştırmaların hem farklı ölçeklerdeki projelerden elde edilecek geçmiş verileri kullanarak inşaat planlamasını otomatik öğrenme üzerine odaklanması hem de daha fazla sayıda gerçek yaşam projesinde test edilmesi gerektiği düşünülmektedir. Bu sayede, daha esnek, uyarlanabilir ve insan hatasına daha az duyarlı yaklaşımların geliştirilmesinin mümkün olacağı ön görülmektedir.

1.3. Problem Tanımı

İnşaat projeleri büyüdükçe, zaman, maliyet ve kalite unsurlarını aynı anda yönetmek giderek zorlaşmakta ve bu nedenle geleneksel iş programı hazırlama yöntemleri yetersiz kalmaktadır. Manuel iş programları gerek proje başlangıcında gerekse proje süresi boyunca büyük oranda uzman müdahalesi gerektirmektedir. Bu durum ciddi verimlilik kayıplarına ve hata riskine yol açarken, iş programların doğruluğunu ve inşaat projelerinin başarısını olumsuz etkilemektedir (Al-Sinan vd., 2024). BIM teknolojisi, 4B modelleme ve simülasyonla birlikte, projelerdeki hataları erkenden tespit etme ve paydaşlar arasında iletişimi geliştirme gibi önemli katkılar sağlamasına rağmen, eleman-aktivite eşleştirmelerinin çoğunlukla manuel veya yarı otomatik yöntemlerle yapılması, yüzlerce öğenin tek tek tanımlanmasını gerektirdiği için kullanıcı uzmanlığına ve zaman alıcı süreçlere aşırı bağımlı kalınmasına sebep olmaktadır (Wefki vd., 2024). Bu bağımlılık, insan hatası riskini artırmakta ve büyük veri setlerinin barındırdığı potansiyeli de

sınırlamaktadır. Mevcut AI ve BIM tabanlı yöntemlerde, önceki projelerin iş programları veya üretim oranları/adam-saat değerleri gibi veri kaynaklarını otomatik işleyip yeniden kullanma kapasitesinin zayıf olması, tekrarlanabilir ve ölçeklenebilir bir bilgi tabanı oluşturmayı güçleştirmektedir. Ayrıca, çoğu araştırma ya yalnızca yapısal elemanları ya da yalnızca mimari elemanları esas almakta; bu kapsamda, iki eleman grubunu bir arada ele alıp Primavera P6, MS Project veya Navisworks gibi yaygın proje yönetim yazılımlarıyla güçlü bir bütünleşme sunan kapsamlı yaklaşımlar oldukça sınırlı sayıda kalmaktadır (ElMenshawy & Marzouk, 2021). Ticari ürünler ve literatürdeki prototip çözümler incelendiğinde, kullanıcılara minimum seviyede manuel girişle otomatik veya yarı otomatik iş programı hazırlama olanağı sunan basit ve esnek arayüzlerin eksikliği göze çarpmakta, bu durum saha uygulamalarına geçişi ve yaygın kullanımını geciktirmektedir.

ML ve AI yöntemleri, büyük veri setlerinden yararlanarak proje planlamasına katkı sunacak yenilikçi çözümler üretebilse de özellikle DL modellerinin sıralı ve zamansal desenleri etkili bir şekilde yakalayabilmek için genellikle çok geniş veri kümelerine ihtiyaç duyması, literatürdeki çalışmaların daha çok büyük ölçekli projelere odaklanmasına yol açmaktadır. Bu durum, küçük ve orta ölçekli inşaat projelerinin yeterince incelenmemesi sonucunu doğurmakta ve ML tabanlı iş programı oluşturma yaklaşımlarının kapsamını daraltmaktadır. Bu nedenle, söz konusu yaklaşımları tam otomasyonla birleştiren çalışmaların sınırlılığı, inşaat sektöründe gerçek anlamda dijital dönüşümü engelleyen temel noktalardan biri olarak öne çıkmaktadır.

Proje süresinin tahmininde ve iş programı oluşturulmasında özellikle metraj cetvelinin doğru hazırlanması büyük önem taşımaktadır. Metrajlar, genellikle 2B projeler üzerinden manuel olarak hesaplandığından, bu süreç hem çok emek ve zaman isteyen hem de hataya açık bir uygulama hâline gelmektedir. Son dönem anket sonuçları, sektör profesyonellerinin önemli bir kısmının hâlâ 2B yazılımları kullandığını ve BIM'in metraj hazırlamadaki potansiyelinden ise kısıtlı ölçüde yararlanıldığını göstermiştir (Collins & Redden, 2022). Literatür, BIM'in metraj süreçlerine dâhil edilmesinin giderek artan bir farkındalıkla benimsendiğini belirtmekle birlikte 3B model oluşturma sürecinin karmaşıklığı nedeniyle yaşanan zorluklara da dikkat çekmektedir (Babatunde vd., 2020).

Bu çerçevede, inşaat projelerinde iş programı hazırlama süreçlerinde yaşanan manuel emek ve zaman kaybı ile yüksek hata riski, sektörün dijital dönüşümünü de sekteye uğratmaktadır. BIM ve makine öğrenmesinin bütünleşik kullanımının sunduğu avantajlar değerlendirildiğinde, proje paydaşlarının planlama iş yükünü azaltacak, geçmiş projelerin

bilgilerini otomatik öğrenerek yeni projeler için tekrarlanabilir bir “akıllı iş programı oluşturma sistemi” kurulmasına ihtiyaç olduğu anlaşılmaktadır. Ancak, var olan çalışmalarda büyük veri ihtiyacı, çok sayıda kısıt, sınırlı aktivite kapsama alanı, manuel müdahaleye duyulan fazla gereksinim ve proje yönetimi araçlarıyla yetersiz entegrasyon gibi problemler hâlâ çözülememiştir. Dolayısıyla, doğru metraj bilgisiyle desteklenen bir BIM modeli oluşturup bunu ML yöntemleriyle birleştiren, aynı zamanda kullanıcı dostu ve proje yönetimi yazılımlarıyla entegre olabilen bir çözüm, hâlihazırda süregelen manuel iş yükü ve verimlilik sorunlarını önemli ölçüde hafifletebilecektir.

Bu doktora tez çalışması, söz konusu sorunları bütüncül bir yaklaşımla ele alarak; BIM tabanlı modellemeyi, ML tabanlı veri yönetimini ve 4B iş programı simülasyonunu eş zamanlı kullanabilen, kullanıcı dostu ve otomasyon odaklı bir çerçeve geliştirmeyi hedeflemektedir. Böylece, hem geleneksel yaklaşımlardaki manuel müdahaleleri ve hata riskini en aza indiren hem de farklı disiplinlerden gelen model ve planlama verilerini bütünleştiren, proje yönetim yazılımlarıyla kolayca entegre olabilen bir “otomatik iş programı oluşturma” çözümü sunarak literatürdeki ve sektördeki bu önemli boşluğu doldurmayı amaçlamaktadır.

1.4. Tezin Amacı ve Kapsamı

Bu tez çalışmasının temel amacı, inşaat projelerinde BIM ve ML yaklaşımlarının bütünleşik bir çerçevede kullanımı ile kullanıcı dostu ve otomasyon odaklı iş programı hazırlanması ve 4B simülasyon elde edilmesidir. Özellikle 3B BIM modellerinden elde edilebilen yapı elemanlarına ait kapsamlı verilerin ve mevcut iş programlarının ML yöntemleriyle işlenerek elde edilen aktivite sıralama bilgilerinin proje planlamasında kullanılabilir hâle getirilmesi amaçlanmaktadır. Böylece geleneksel yöntemlerle yoğun emek ve zaman gerektiren, ayrıca insan hatasına açık olan iş programı oluşturma süreçlerinin daha hızlı, güvenilir ve tekrarlanabilir bir yapıya kavuşturulması öngörülmektedir.

Tez kapsamında öncelikle, geçmiş projelerden aktivite sıralamaları öğrenilerek yeni projelere adapte edilebilmesi için ML tabanlı bir çerçeve oluşturulması planlanmaktadır. Ardından, 3B BIM modellerinde yer alan geometrik ve niteliksel bilgilerin otomatik olarak elde edilmesi ve makine öğrenmesinden elde edilen sıralamalar ile entegre edilerek iş programına dönüştürülmesi sağlanacaktır. Sonraki aşamada, 4B model ve simülasyon

desteğiyle zaman-mekân ilişkilerinin görsel olarak izlenmesi ve inşaat faaliyetlerine dair karar mekanizmalarının güçlendirilmesi hedeflenmektedir. İlgili yaklaşımın hem büyük ölçekli kamu ihaleleri hem de özel sektör projelerinde iş programlarının güvenilirliğini artırması ve karar vericilere proje başlangıcından itibaren öngörü ve müdahale imkânı sunması beklenmektedir.

Bu tez çalışmasıyla geliştirilecek model, proje yönetimi yazılımlarıyla bütünleşik çalışmayı ve sektörün kullanım alışkanlıklarını dikkate alarak kurgulanmıştır. Bu sayede, geleneksel manuel yöntemlere kıyasla daha az uzman müdahalesi gerektiren ve benzer projelere kolayca uyarlanabilen bir planlama sürecinin mümkün kılınması öngörülmektedir. Ayrıca, tezde sunulan çözümün kamu ve özel sektör tarafından yaygın şekilde benimsenebilmesi amacıyla saha verileriyle test edilmesi ve sonuçların bilimsel ölçütlerle değerlendirilmesi de çalışmanın kapsamındadır. Elde edilecek bulguların, BIM ve makine öğrenmesinin inşaat sektörü özelinde daha etkin kullanımı için akademik literatüre katkı sunması ve gelecekteki araştırmalara yol göstermesi beklenmektedir.

1.5. Tezin Kısıtlamaları

Bu tezde sunulan BIM ve ML tabanlı yaklaşımlar, veri kalitesi ve erişilebilirliği hususuna önemli ölçüde bağımlıdır. ML modellerinin doğru ve genellenebilir sonuçlar üretebilmesi için geniş ve güvenilir veri setlerine ihtiyaç duyulmakta, ancak mevcut projelerden temin edilebilecek veriler proje ölçeği, kurumsal veri politikaları ve dokümantasyon alışkanlıkları nedeniyle sınırlı kalabilmektedir. Ayrıca tez çalışmasında, yalnızca üst yapı projeleri dikkate alınarak araştırma kapsamı daraltılmıştır. Bu tercihin temel nedeni, farklı türlerdeki (örneğin, altyapı ve tünel inşaatları gibi) projelerin kendine özgü süreç ve parametrelere sahip olmasıdır. Elde edilen bulgular, betonarme veya çelik karkas gibi üst yapı projeleri için daha doğrudan uygulanabilir nitelik taşımakta, diğer proje türleri için ise ilave uyarlamalar veya veri toplama çalışmaları gerekmektedir.

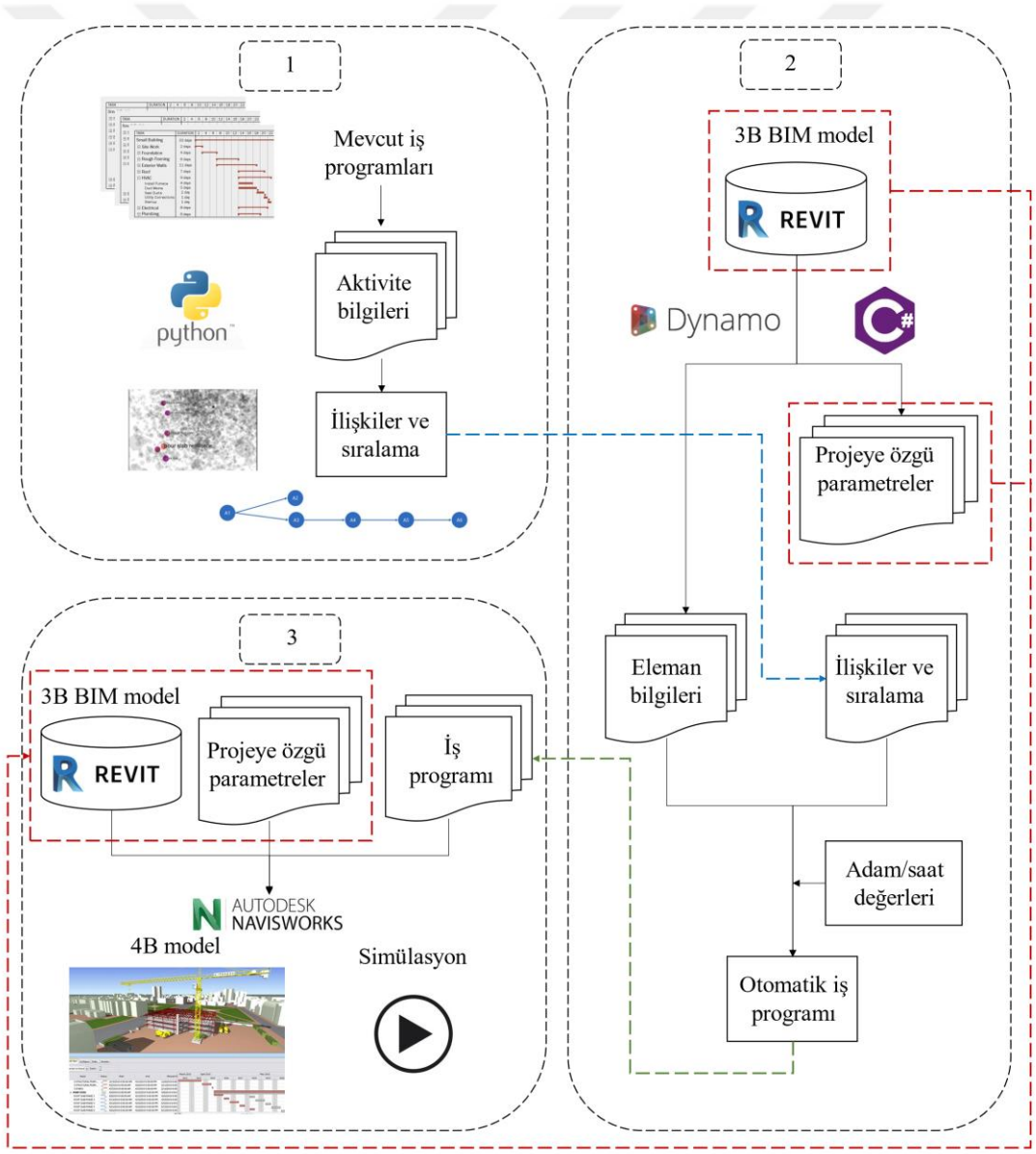
Revit yazılımıyla gerçekleştirilen üç boyutlu modellemelerde, elemanların tanımlanması ve sınıflandırılması için belirli kriterlerin uygulanması gerekli görülmüştür. Eleman isimlendirme ve parametre atama gibi kuralların tutarlı biçimde izlenmesi, otomatik iş programı oluşturulmasında kritik öneme sahiptir. Farklı adlandırma yöntemleri veya eksik parametre bilgileri, 3B modelden elemanlara ait bilgilerin elde edilmesi ve

aktivite sıralaması sürecini olumsuz etkileyerek modelin etkinliğini ve doğruluğunu sınırlandırabilmektedir.

Bunlara ek olarak, donatı metrajı hesaplarında yapı birim alanına isabet eden yaklaşık donatı miktarı dikkate alınmıştır. Öte yandan, bu miktarın değiştirilebilmesi için geliştirilen arayüz üzerinden kullanıcıya izin de verilmektedir.

1.6. Araştırma Yöntem Özeti

Araştırmanın amacına ulaşmak için izlenen yöntem üç ana aşamadan oluşmaktadır. Yönteme ait iş akış şeması Şekil 2'de verilmektedir.



Şekil 2. Araştırma metodolojisi özet akış şeması

İlk olarak, gerçekleştirilmiş yapım projelerine ait iş programlarından elde edilen veriler kullanılarak aktiviteler arası ilişkiler öğrenilerek bu ilişkilerin otomatik olarak oluşturulması gerçekleştirilmiştir. Bu süreçte, Birliktelik Kural Madenciliği (Association Rule Mining, ARM), node2vec ve DeepWalk gibi ML yöntemleri ile Grafik Evrişimli Ağlar (Graph Convolutional Network, GCN), Grafik Dikkat Ağlar (Graph Attention Network, GAT) ve GraphSAGE gibi DL yöntemleri kullanılmıştır.

İkinci aşamada, Revit programı kullanılarak modellenmiş 3B yapıdan elemanlara ait bilgiler otomatik olarak alınmış ve iş programı oluşturulmuştur. Açık kaynaklı Dynamo ve C# araçlarıyla Revit'e entegre edilen bir eklenti (add-in) sayesinde, yapı elemanlarına ilişkin isim, malzeme ve ölçüler gibi parametreler belirlenmiş ve iş programı aşamasında faydalanmak için elemanlara yeni parametreler tanımlanmıştır. Standart olarak verilen ancak kullanıcı tarafından değiştirilmesine de izin verilebilen adam-saat, ekip sayısı ve günlük çalışma süresi gibi değerler doğrultusunda, aktivite süreleri otomatik olarak güncellenebilmektedir. Bu bilgiler sayesinde iş programı elde edilmiş ve .csv formatında dış ortama aktarılmıştır. Ayrıca, .xml formatında Navisworks'te elemanlarla iş programındaki aktiviteleri eşleştirmek için "Search Sets" dosyası da oluşturulmuş ve dış ortama aktarılmıştır.

Son olarak, Navisworks programı kullanılarak 4B modelleme ve simülasyon gerçekleştirilmiştir. Revit'ten elde edilen 3B model, .xml formatında Navisworks Search Sets dosyası ve .csv dosyası, C# ile geliştirilen eklenti sayesinde Navisworks Timeliner aracılığıyla aktivitelerle model elemanları eşleştirilmiştir. Bu sayede, projeye ilişkin 4B simülasyon hazır hale getirilmiş ve kullanıcıya görsel bir iş akışı sunulmuştur.

1.7. İnşaat Projelerinde İş Programı Oluşturulması ve Önemi

Proje; yeni ve benzersiz bir ürün veya hizmet oluşturmak için başlangıcı ve bitişi olan geçici faaliyetler olarak tanımlanmaktadır (PMI, 2021). Bir projenin başarıya ulaşabilmesi için proje yönetiminin temel ilkelerine hâkim olmak önemlidir. Proje yönetimi; projenin hedeflenen süre, maliyet ve kalite gibi parametrelere uygun şekilde planlanması, uygulanması ve denetlenmesi biçiminde tanımlanmaktadır (Kuruoğlu, 2007). Özellikle inşaat projeleri, farklı mevsim durumları, coğrafi konum ve zemin koşullarının yanı sıra tedarik zinciri ve iş gücü gibi geniş bir yelpazede risk ve belirsizlik barındırdığı

için, bu karmaşık yapıyı koordine edebilmek amacıyla proje yönetimine duyulan gereksinim giderek daha stratejik hâle gelmektedir (Karakurumer, 2024).

Proje yönetiminin en önemli aşamalarından biri, iş programının hazırlanmasıdır. İş programı, projenin hangi aşamada, hangi ekip ve kaynaklarla, ne kadar sürede yürütüleceğini belirlemekte ve kontrol sürecini kolaylaştırmaktadır (DeWitt, 2016). Programın bulunmadığı veya yetersiz biçimde oluşturulduğu durumlarda, sahada yaşanan belirsizlikler artmakta ve koordinasyon eksikliği proje sürecinde ciddi sorunlara yol açmaktadır (Saraç Çıracıoğlu, 2021). İş programına dayalı planlama faaliyeti, disiplinler arası bir çerçevede yürütülerek hem paydaşların görev tanımlarını hem de malzeme ve ekipman tedarikini daha net bir tablonun parçası hâline getirmektedir (Çapraz, 2024).

Zaman planlaması, inşaat projelerinin en kritik unsurlarından biridir ve proje başarısının değerlendirilmesinde temel ölçütler arasında yer almaktadır (Faghihi vd., 2015). Sürenin uzaması, ek iş gücü masrafları, tedarik süreçlerinde artış veya makine ve ekipman kullanım süresinin uzaması şeklinde maliyet kalemlerinde yükselişe sebep olabilmektedir (Yeşilyurt, 2017). Diğer taraftan, maliyetin beklenmedik şekilde artış göstermesi de projenin zaman planlamasını doğrudan etkilemekte ve gecikme ya da kalite standartlarında ödün verme gibi istenmeyen durumları beraberinde getirebilmektedir. Dolayısıyla iş programının temel kurgusu, süre ve maliyet arasındaki bu hassas ilişkiyi gözeten bir anlayışla oluşturulmalıdır (Mostofi, 2024).

İnşaat sektöründe geleneksel planlama, çoğunlukla projenin başında hazırlanan Gantt (Çubuk) Diyagramı ya da Kritik Yol Yöntemi (Critical Path Method, CPM) ağ şeması üzerinden yürütülmektedir (Özcan, 2021). Ancak proje ilerledikçe ortaya çıkan tasarım değişiklikleri veya öngörülmeyen sorunlar sonucunda planların hızla revize edilmesi gerekmektedir. Sahada çok sayıda paydaşın birbirinden uyumsuz biçimde çalışması, güncellenen bilgi akışının bütünleştirilmesini zorlaştırarak, gecikmeler ve ek maliyetlere yol açmaktadır (El-Sabek & McCabe, 2017). Geleneksel planlama yöntemleri, özellikle büyük ölçekli projelerde çoklu ve birbirine bağımlı faaliyetlerin aynı anda yönetilmesi sürecinde belirgin zorluklar yaratmaktadır (Saraç Çıracıoğlu, 2021).

Bu bağlamda, teknoloji destekli planlama yaklaşımları ve yazılımlar (örneğin, Primavera, Microsoft Project), iş programı hazırlama ve kontrol süreçlerinde daha esnek ve etkileşimli yönetim seçenekleri sunmaktadır (Özcan, 2021). Aynı zamanda BIM gibi yeni teknolojiler, tasarım ve planlama verilerinin tek bir dijital platformda toplanmasını

mümkün kılmakta, farklı disiplinler arasındaki koordinasyonu ve iş takibini kolaylaştırarak sürekli iyileştirme odaklı bir verimlilik artışı sağlamaktadır (Çapraz, 2024).

Tüm bu unsurlar dikkate alındığında, inşaat sektöründe iş programı oluşturmak ve bu programa dayalı bir proje yönetimi yaklaşımı benimsemek, projenin taslak aşamasından teslim aşamasına kadar tüm süreçlerde belirleyici rol üstlenmektedir. Proje ve proje yönetimi kavramlarının önemi, proje yönetimine neden gereksinim duyulduğu, planlamanın veya iş programının neden yaşamsal olduğu, sürenin maliyet ve kalite gibi diğer alanlarla etkileşimi, geleneksel planlama süreçleri ve bunların yetersizlikleri ile gelişen teknoloji destekli yaklaşımlar bir arada değerlendirildiğinde, iş programı oluşturmanın bir seçim değil, projelerdeki başarının sağlanması bakımından zorunlu bir adım olduğu açıkça görülebilmektedir.

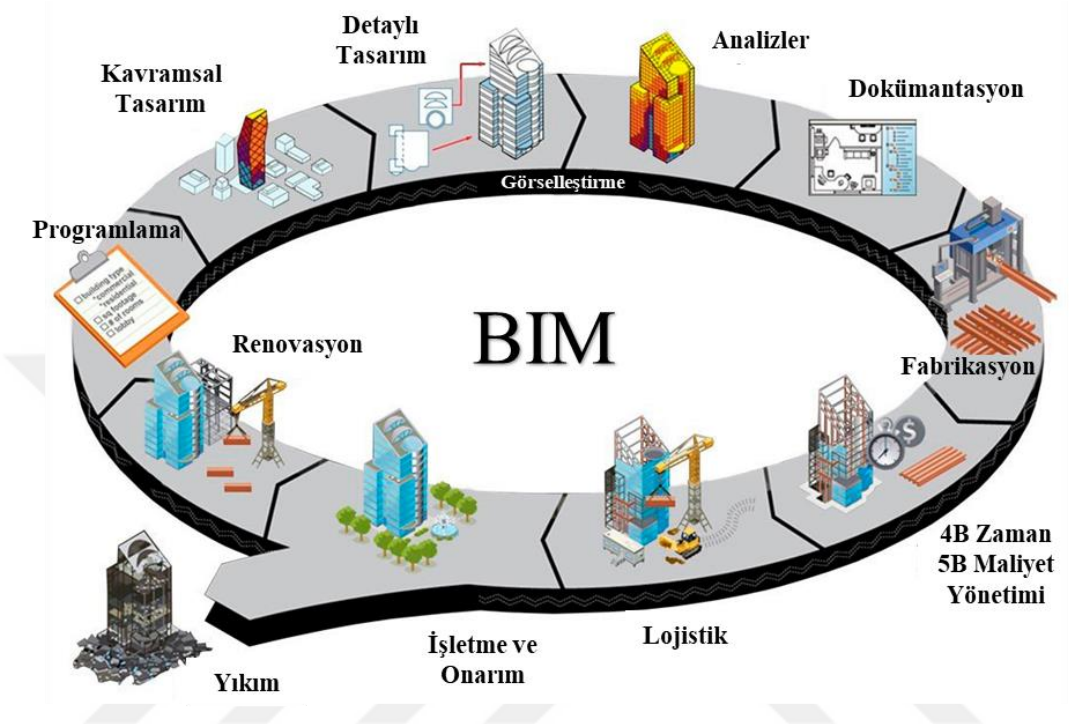
1.8. Yapı Bilgi Modellemesi (BIM)

Tez çalışmasının bu bölümünde, BIM'in temel ilkeleri ve uygulama yöntemleri ele alınmakta ve mimarlık, mühendislik ve inşaat (Architecture, Engineering and Construction, AEC) alanlarında nasıl kullanıldığına ilişkin ayrıntılı bilgilere yer verilmektedir. İlerleyen alt başlıklarda önce BIM'in tanımı ve tarihsel gelişimi irdelenmekte, daha sonra BIM'in avantajları ve dezavantajları incelenmekte ve son olarak da 4B modelleme yaklaşımının projelerin zaman boyutlu yönetimine sağladığı katkılar tartışılmaktadır. Bu çerçevede, güncel uygulamaların ve gelecek öngörülerinin ışığında BIM'in sektör pratiğine kazandırdığı yenilikler ortaya konulmaktadır.

1.8.1. BIM'in Tanımı ve Tarihsel Gelişimi

BIM, AEC sektöründeki verilerin bütüncül biçimde dijital olarak yönetilmesini mümkün kılan bir yaklaşım ve yaşam döngüsü boyunca alınacak kararlar için güvenilir bir bilgi kaynağı olarak tanımlanmaktadır (NBIMS, 2012). BIM ile birlikte bir yapının yaşam döngüsü Şekil 3'te gösterilmektedir. Bu süreç, yapıların tasarımından inşaatına ve işletme-bakım aşamasına kadar pek çok evrede paydaşlar arasındaki bilgi akışını düzenli ve doğru biçimde sağlamaya odaklanmaktadır (Enegbuma vd., 2014; Paneru vd., 2023). Söz konusu yaklaşım çerçevesinde, projeye ilişkin her türlü veri (örneğin, geometrik, mekânsal,

fonksiyonel veya maliyet bilgileri) tek bir akıllı model üzerinden aktarılmakta olup, böylece ilgili kişi veya kurumlar güncel ve güvenilir bilgilere eşzamanlı olarak erişebilmekte ve proje sürecine etkin biçimde katılabilmektedir (Bahadır, 2018).

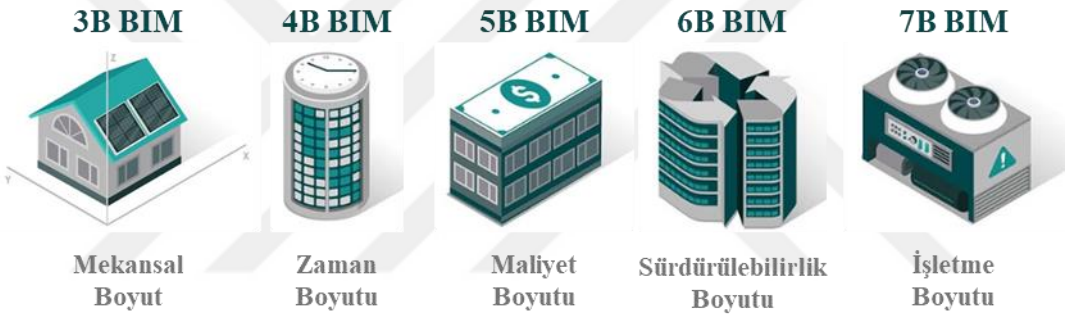


Şekil 3. BIM ile bir yapının yaşam döngüsü (URL-1, 2024)

BIM, ilk ortaya çıktığı dönemde yalnızca bir 3B modelleme yazılımı olarak algılanmaktayken, 1990'lı yıllardan itibaren dijital dönüşümün getirdiği teknolojik yenilikler sayesinde giderek daha fazla önem kazanan bir uygulama alanı hâline gelmektedir (Das, 2024). Özellikle proje yönetiminde karmaşık verilerin düzenlenmesini kolaylaştırması ve bilgi kaybını önlemesi, BIM'in başlıca avantajları arasında yer almaktadır (Borrmann vd., 2018). Geleneksel yöntemlerde proje paydaşları arasındaki bilgi akışı çoğunlukla parçalı ve tekrar odaklı bir yol izlerken, BIM süreci her paydaşın kendi uzmanlık alanından gelen bilgiyi ortak bir platformda paylaşabilmesini mümkün kılmaktadır (Alshammari vd., 2021). Böylece disiplinler arası koordinasyon gelişmekte, muhtemel çakışmaların erken dönemde tespit edilmesi sağlanmakta ve proje yönetimi boyunca oluşabilecek sorunlar azaltılmaktadır (Das, 2024). Tarihsel gelişim süreci incelendiğinde, BIM'in 1990'lardan bu yana artan bir ivmeyle benimsenmesi ve sektörün farklı bileşenleri tarafından içselleştirilmesi, inşaat projelerinin bütüncül bir yaklaşımla yönetilmesine olanak sağlamaktadır (Hochscheid & Halin, 2020). Bu doğrultuda BIM,

teknoloji odaklı yeniliklerin öncüsü olarak, sürdürülebilir ve etkin bir proje yönetim anlayışının vazgeçilmez bir parçası olmaya devam etmektedir (Wang & Chen, 2023).

BIM'in terminolojik olarak tam anlamıyla öne çıkması ise 2002 yılında sanal tasarım, inşaat ve tesis yönetimi süreçlerini tanımlanmasıyla olmuştur (Harris, 2010). 2000 yılında Charles River tarafından geliştirilen ve parametrik bileşenler fikrini kullanan Revit yazılımı, BIM teknolojilerinin küresel ölçekte bilinirliğini artıran önemli bir örnek olarak sunulmuştur (Lee vd., 2015; Quirk, 2012). Başlangıçta yalnızca 3B ile sınırlıymış gibi görünen BIM süreci, zaman (4B), maliyet (5B), sürdürülebilirlik (6B) ve işletme-bakım (7B) gibi boyutların da sisteme eklenmesiyle çok daha kapsamlı bir hâl almıştır (Fleming, 2016; Wu vd., 2019). Şekil 4'te örneklenen bu anlayış, inşaat projelerinin daha bütüncül bir yaklaşımla yönetilmesine olanak sağlamaktadır.



Şekil 4. Çok boyutlu BIM süreci (URL-2, 2024)

Yalnızca proje planlama, tasarım ve yapım süreçlerinde değil, işletme ve bakım aşamalarında da daha verimli bir yönetim yaklaşımı sunmayı hedefleyen BIM, dijital model tabanlı bilgi alışverişini projenin tüm yaşam döngüsüne yaymaktadır (Lu vd., 2020). İnşaat sektöründe karmaşık ve çok katmanlı süreçlere sahip projelerin karar mekanizmalarına olumlu katkıda bulunan bu yaklaşım, proje kapsamındaki riskleri azaltmakta, kaynakların sürdürülebilir kullanımını desteklemekte ve genel kaliteyi yükseltmektedir (Ali vd., 2022). Bu nedenlerle BIM, inşaat projelerine yönelik bütüncül bakış açısı arayışının önemli bir aracı olarak konumunu güçlendirmeye devam etmektedir.

AEC sektörünün, BIM ile ilgili gelişmelere uzun yıllar boyunca odaklanmasının bir sonucu olarak, pek çok BIM yazılım aracı geliştirilmiş ve kullanıcıların erişimine sunulmuştur (Arayıcı vd., 2011). BIM yazılımları, proje çizim ve dokümantasyon aşamalarını otomatikleştirerek, tasarım değişikliklerinin tüm belge ve görsellerde

eşzamanlı güncellenmesini mümkün kılmaktadır (Singh vd., 2011). Bu durum, proje dokümanları arasındaki tutarlılığın korunmasını, tasarım alternatiflerinin hızlı biçimde oluşturulmasını ve hataların asgariye indirilmesini sağlayarak hem zaman hem de maliyet noktasında tasarruf sağlanmakta ve uygulama kalitesi artmaktadır (Eastman vd., 2008).

Farklı ülkelerdeki kurumsal politikalar ve pazar dinamikleri, BIM'in yaygınlaşmasında önemli bir rol oynamaktadır. Amerika Birleşik Devletleri (ABD) ve Birleşik Krallık (UK) gibi ülkeler, kamu projelerinde BIM kullanımını zorunlu hâle getiren düzenlemeleriyle ön plana çıkmaktadır (Charef vd., 2018). ABD'de, ABD Genel Hizmetler İdaresi (GSA) kamu projelerini BIM esaslı yürütmekte ve sektör paydaşlarının bu teknolojiyi benimsemesine öncülük etmektedir (Smith, 2014; Edirisinghe & London, 2015). Benzer biçimde, UK hükümeti de 2011 yılında yayımladığı strateji doğrultusunda 2016'dan itibaren 5 milyon pound üzeri kamu projelerinde asgari BIM Seviye 2'yi zorunlu kılmıştır (Pittard & Sell, 2016). Finlandiya, Danimarka ve Norveç gibi İskandinav ülkeleri de devletin liderliği altında BIM uygulamalarını yaygınlaştırmayı başarmıştır (Kıvırcık, 2016). Almanya ve Fransa'da ise özel sektör ve kamu projeleri paralel biçimde BIM benimsediğinden, sektör genelinde yüksek adaptasyon oranları gözlemlenmektedir (McGraw Hill, 2014). Öte yandan, Asya ülkelerinde Singapur hükümetinin "Construction Productivity and Capability Fund" (CPCF) gibi teşvik mekanizmaları geliştirdiği, Çin'de ise pazar büyüklüğü ve uluslararası firmaların katkıları sayesinde BIM'e geçişin hız kazandığı ifade edilmektedir (McGraw Hill, 2014). Japonya'da BIM uygulamalarının nispeten yüksek olduğu görülmekte ve bu teknolojiye yapılan yatırımların geri dönüşünün olumlu olduğu raporlanmaktadır. Buna karşın, yüksek teknoloji altyapısıyla bilinen Güney Kore'de BIM benimsemesi beklenenden yavaş seyretmektedir (Smith, 2014). Brezilya ve Hindistan gibi gelişmekte olan ekonomilerde de uluslararası inşaat şirketlerinin varlığı BIM'in yaygınlığını artırmakta, ancak yerel paydaşların uyum süreçleri hâlen başlangıç düzeyinde kalmaktadır (McGraw Hill, 2014).

Türkiye'de ise inşaat şirketlerinin son yıllarda BIM yatırımlarını artırdığı ve özellikle ulaşım ve altyapı projelerinde bu teknolojinin kullanılmaya başladığı görülmektedir (Özorhon, 2018). Henüz Avrupa ülkeleri kadar yaygın olmasa da kamu kurumlarının da kısmen BIM'e dair koşullar öne sürdüğü görülmekte, bu durum sektör genelinde farkındalığı ve talebi artırmaktadır. Yazılım üreticileri, yerli firmalara BIM tabanlı çözümler sunmaya yoğunlaşmakta olup, inşaat sektöründeki büyük ölçekli projelerde BIM entegrasyonu konusunda bir arayış ve öğrenme süreci devam etmektedir (Maçka Kalfa,

2018). Buna karşın, Elmalı & Bayram (2022) tarafından yapılan bir araştırmada, Türkiye'de BIM benimsenmesinin hala başlangıç aşamasında olduğu, inşaat mühendisleri ile mimarlar ve kamu ile özel sektör arasında BIM farkındalığı konusunda farklılıklar bulunduğu belirtilmektedir. Buna karşın uzun vadede BIM'in Türkiye inşaat sektöründe daha da yaygınlaşacağı ve sektörel verimlilik ile sürdürülebilirlik hedeflerine katkı sunacağı öngörülmektedir (Kaya & Özener, 2024). Zaman içerisinde daha ileri boyutlara ulaşması beklenen BIM'in, inşaat projelerine yönelik bütüncül bakış açısı arayışında öncü rolünü sürdürmesi ve sektörün verimlilik ve sürdürülebilirlik hedeflerine katkı sağlaması beklenmektedir (Temel, 2022).

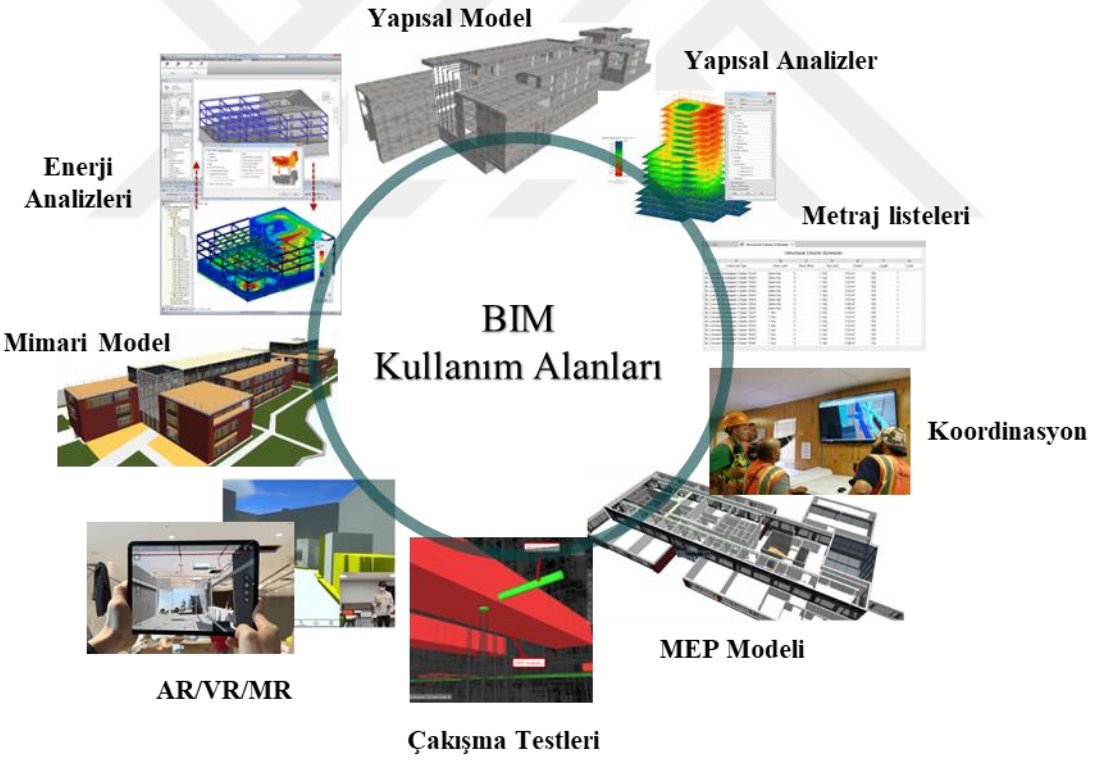
1.8.2. BIM'in Avantajları ve Dezavantajları

BIM, inşaat projelerinde tasarım, uygulama ve yönetim süreçlerine ilişkin bütüncül bir veri yönetimi ve disiplinler arası iş birliği sağlaması sayesinde, geleneksel yöntemlere kıyasla daha yüksek verimlilik potansiyeli barındırmaktadır (Bahadır, 2018). Parametrik modelleme ilkelerine dayanan BIM yazılımları aracılığıyla proje paydaşları, aynı dijital model üzerinde disiplinler arası veri paylaşımı yapabilmekte ve 3B görselleştirmeler sayesinde tasarımları önceden inceleyerek hata ve eksiklikleri erken aşamada tespit edebilmektedir (Czmoch & Pekala, 2014). Bu yaklaşım, projenin konsept ve fizibilite çalışmalarında da kolaylık sağlamaktadır. Ayrıca, tasarım süresince yapılan değişikliklerin diğer dokümanlara anlık olarak yansıması, 2B çizimlerin güncel ve tutarlı biçimde üretilmesine imkân tanımaktadır. Söz konusu iyileştirmeler maliyet ve zaman kayıplarının önlenmesine olanak tanımakta ve müşteri memnuniyetini de artırmaktadır (Vysotskiy vd., 2015).

BIM, yalın inşaat felsefesiyle benzer bir bakış açısını benimseyerek israfı azaltma, hızlı geri bildirim sağlama ve disiplinler arası koordinasyonu güçlendirme gibi hedefleri desteklemekte ve bu sayede süreçlerin değer yaratma kapasitesini üst düzeye çıkarmaktadır (King, 2009). Çakışma tespitlerinin erkenden yapılabilmesi, planlama süreçlerinin daha sağlıklı yürütülmesine katkı sunarken, proje ekibinin iş güvenliği ve risk yönetimi konularında da daha bilinçli kararlar alabilmesini kolaylaştırmaktadır (Azhar vd., 2012). Bu durum, müşterilerle ilişkilerin gelişmesine ve zaman yönetiminin iyileştirilmesine de katkı sağlamaktadır. Öte yandan, BIM sistemlerinde oluşturulan bütünleşik veri tabanının, işletme ve bakım aşamalarında da devredilebilir bilgileri barındırması, tesis yönetim

süreçlerinde şeffaflığın ve etkinliğin artmasına yardımcı olmaktadır (Cao vd., 2015; Aziz vd., 2016).

BIM'in dikkat çeken avantajlarından bir diğeri, dijital bilgi yönetimi yoluyla sürdürülebilirlik odaklı yaklaşımların projelerde daha etkin biçimde uygulanabilmesine imkân tanımasıdır (Ali vd., 2022). Karbon salınımı, atık miktarı ve enerji tüketimi gibi önemli sürdürülebilirlik göstergeleri, BIM uygulamaları sayesinde detaylı biçimde analiz edilebilmekte ve proje sürecinde daha verimli ve çevre dostu ürün seçeneklerini değerlendirme fırsatı sunmaktadır (Volk vd., 2013). Aynı zamanda, ortak veri tabanı üzerinden çalışan proje paydaşları, sürdürülebilirlik odaklı çözümleri birlikte geliştirme ve sonuçlarını ölçme olanağına sahip olarak, bütüncül bir bakış açısı edinmektedir (Kıvırcık, 2016). Şekil 5'te, BIM'in sağladığı faydalar ve farklı kullanım alanlarından bazıları özetlenmektedir.



Şekil 5. BIM kullanım alanları

Diğer taraftan, BIM uygulamalarının yaygınlaşması, çeşitli zorluklar ve dezavantajlar da barındırmaktadır. Yüksek başlangıç maliyetleri, kapsamlı donanım ihtiyacı ve karmaşık eğitim süreçleri, özellikle küçük ve orta ölçekli işletmeler açısından önemli bir bariyer oluşturmaktadır (Tulenheimo, 2015). BIM konusunda yetkin personel

bulma güçlüğü, firmaların teknolojiyi benimseme süreçlerini yavaşlatmaktadır. Ayrıca, mevcut iş akışlarının yeniden yapılandırılması, proje paydaşlarının ortak bir BIM olgunluk seviyesine erişmesi ve örgüt içi koordinasyonun sağlanması karmaşık bir dönüşüm süreci gerektirmektedir (Lindblad & Vass, 2015). Ek olarak, BIM'i yalnızca 3B modelleme aracı olarak algılayan paydaşların teknolojinin sunduğu bütüncül potansiyeli göz ardı etmesi de benimsenme hızını yavaşlatmakta ve sektörde beklentilerin karşılanmasını güçleştirmektedir (Ahmed, 2018).

İnşaat sektörünün parçalı yapısı ve proje bazlı organizasyonlar da BIM entegrasyonunu güçleştirebilmektedir. Bir projede yer alan her paydaşın farklı BIM beceri seviyelerine ve iş yapma yöntemlerine sahip olması, veri bütünlüğünün sağlanmasında engeller oluşturabilmektedir (Tulenheimo, 2015). Kamu kurumları ve özel sektör arasında BIM standartlarına yönelik henüz tam oturmamış kılavuzların olması, projelerin sözleşme aşamasından itibaren "hangi standartlarla ve hangi ayrıntı düzeyiyle/detay seviyesiyle (Level of Development/Detail, LOD) çalışılacağı" sorusunu gündeme getirmekte ve veri bütünlüğü sağlanmasında ek zorluklar yaşanabilmektedir (Froese, 2010; Bargstäd, 2015). Saha uygulamalarında ise sürekli değişen verilerin anlık olarak modele işlenmesi, ilave iş yükü ve koordinasyon gereksinimi doğurarak süreci daha da karmaşık bir hâle getirebilmektedir (Bargstäd, 2015). Tüm bu etmenler, BIM'e geçişin yalnızca teknolojik yatırımlarla sınırlı kalmadığını, aynı zamanda işletmelerin örgüt yapıları, iş akışları ve proje yönetim yaklaşımlarını da yeniden düzenlemeleri gerektiğini göstermektedir (Froese, 2010).

Parametrik modelleme, ortak veri tabanı ve disiplinler arası iş birliği gibi özelliklerin inşaat süreçlerini daha şeffaf, hızlı ve entegre bir yapıya kavuşturmasına karşın, yüksek adaptasyon maliyetleri ve sektördeki parçalı yapı uygulamanın yaygınlaşmasını sınırlayabilmektedir (Kıvırcık, 2016). Bu nedenle, BIM'in sunduğu fırsatlardan yararlanmak isteyen şirketlerin, teknolojik altyapının yanı sıra örgütsel kültür ve iş yapma biçimlerini de kapsamlı biçimde gözden geçirmesi büyük önem taşımaktadır (Tulenheimo, 2015). Paydaşlar arasında net bir anlaşma, ortak standartlar ve sözleşme esaslarının belirlenmesi durumunda, BIM'in inovasyon kapasitesi daha etkili bir biçimde değerlendirilebilmekte ve inşaat sektörünün geleceğini şekillendirecek yenilikçi uygulamalar için sağlam bir temel oluşturulabilmektedir (Eastman vd., 2011).

1.8.3. 4B BIM ve Detay Seviyesi (LOD)

BIM'in çok boyutlu veri yönetimi anlayışı, geleneksel 3B modellemenin ötesinde zaman (4B), maliyet (5B), sürdürülebilirlik (6B) ve işletme-bakım (7B) gibi ek boyutların projeye entegre edilmesine olanak tanımaktadır (Charef vd., 2018). Böylelikle yapım projeleri, tasarımdan işletmeye kadar daha bütüncül bir bakış açısıyla ele alınabilmektedir. Bu çalışma kapsamında ise özellikle 4B modelleme, yani zaman boyutunun modele eklenmesi incelenmektedir.

4B yaklaşımı, 1990'larda kompleks altyapı ve endüstriyel tesis projelerinde kullanılmaya başlanmış ve zaman içinde bina projelerinde de giderek yaygınlaşmıştır (Collier & Fischer, 1995; Eastman vd., 2011). Başlangıçta tasarım evresinde farklı senaryoların değerlendirilmesi ve görselleştirilmesi amacıyla geliştirilen bu yaklaşım, ilerleyen dönemlerde yapım aşamasına da uyarlanmıştır. 4B modelleme, temel olarak 3B yapı elemanlarını proje takvimindeki iş programı aktivitelerine bağlayarak zaman-mekân koordinasyonunu görsel ve etkileşimli bir biçimde sunmayı hedeflemektedir (Sacks vd., 2009). Bu sayede hangi elemanın hangi aşamada inşa edileceği, sahada hangi ekipmanın ne zaman kullanılacağı ve olası çakışmaların nerede ortaya çıkabileceği önceden simüle edilebilmektedir (Eastman vd., 2011; Şenol, 2023). Böylece, iş programının görsel olarak anlaşılması kolaylaşmakta ve farklı paydaşlar arasındaki iletişim güçlenmektedir (Şenol, 2023). Ayrıca, önceden tespit edilemeyen tasarım ve yapım problemleri, 3B model üzerindeki zaman akışı yardımıyla belirlenebilmekte ve inşaatın her aşamasında daha gerçekçi planlama senaryoları geliştirilmesine olanak sağlamaktadır (Sacks, 2009; Olde Scholtenhuis vd., 2016). Böylece, bilgi akışının koordine edilmesi ile takım içi koordinasyon artmakta, sahaya özgü zorluklar (örneğin, ekipman hareket alanlarının çakışması) henüz uygulamaya geçilmeden giderilebilmekte ve kaynak dağılım problemlerine çözüm sunulabilmektedir (Kassem vd., 2015). 4B simülasyonlarının doğru kullanıldığı projelerde maliyet ve zaman aşımlarının önemli ölçüde azaltılabildiği, geleneksel yöntemlere kıyasla yaklaşık %40 oranında daha yüksek verimlilik sağlandığı belirtilmektedir (Candelario-Garrido vd., 2017). Ayrıca, inşaat ilerlemesinin takibinde 4B yönteminin kullanılması durumunda %95,9 oranında doğruluk elde edilebildiği de literatürde ifade edilmektedir (Han & Golparvar, 2015).

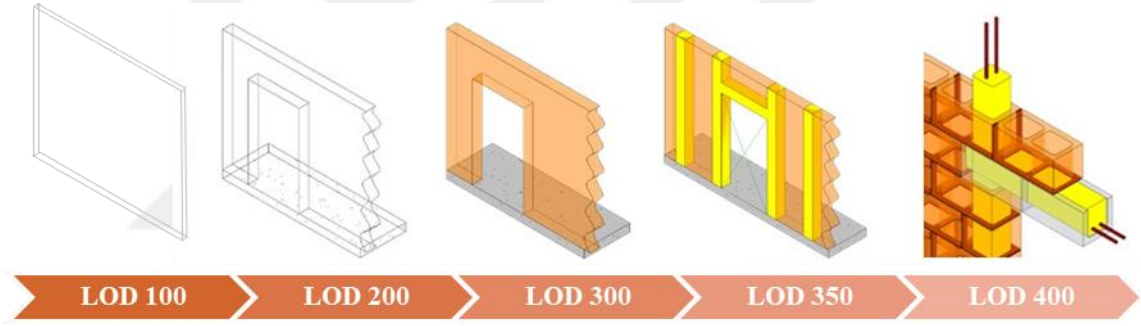
4B modelleme farklı yöntem ve araçlar kullanılarak gerçekleştirilmektedir. Örneğin manuel (CAD tabanlı) yaklaşımlarda, 2B paftalar üzerinde değişik renk ve katmanlarla iş

sıralarını işaretlemek yaygın ama yoğun emek harcanan bir yöntemdir (Eastman vd., 2011). Günümüzde Autodesk Navisworks, Synchro, Vico veya Powerproject gibi 4B simülasyon yazılımları, kapsamlı veri bütünleştirme, otomatik eşleştirme, kaynak yönetimi, risk analizi ve alternatif senaryo karşılaştırma gibi özellikler sunarak hem planlama hem de kontrol aşamalarını önemli ölçüde iyileştirmektedir (Özcan, 2021). Proje takvimleriyle ve BIM modelleriyle bütünleşik biçimde çalışan bu çözümler, farklı zaman dilimlerinde sanal şantiye görselleri üreterek saha lojistiği, vinç kurulum alanları, ekipman yerleşimi ve malzeme depolama sahaları gibi ayrıntıların daha doğru planlanmasına imkân tanımaktadır. Bu sayede, proje paydaşlarına alan-zaman ilişkisi görsel olarak aktarılmakta ve hızlı karar alma süreçlerinde de destek olunmaktadır (Wefki vd., 2024).

4B BIM'in operasyonel verimlilik ve iletişim alanında sunduğu önemli avantajlara rağmen, pek çok çalışma 4B BIM'in sektör genelinde yeterince benimsenmediğini ortaya koymaktadır (Hartmann vd., 2008; Mahalingam vd., 2010; Trebbe vd., 2015). Bu durumun başlıca sebepleri arasında, 4B BIM'in yeni ve köklü bir dönüşüm algısı yaratması, standartların ve modelleme protokollerinin henüz yeterince gelişmemesi, model verilerinin planlama süreçleriyle tam uyumlu olmayışı, 4B teknolojisinin tam olarak nasıl kullanılacağının bilinmemesi ve yüksek uygulama maliyetleri yer almaktadır (Stanley & Thurnell, 2014; Hosseini vd., 2015; Gledson & Greenwood, 2016; Olsen & Taylor, 2017). Buna ek olarak, çoğu proje hâlâ el yöntemiyle planlandığından ve kontrol edildiğinden, 4B BIM'i tüm aşamalara yaygın biçimde entegre etmek yerine yalnızca pilot uygulamalarda kullanma eğilimi olduğu da görülmektedir (Kataoka, 2008; Kroop vd., 2018). Ayrıca, yatırım geri dönüşünün belirsizliği gibi faktörler de inşaat sektöründe 4B BIM adaptasyonunu sınırlandırmaktadır (Ahmed vd., 2014).

3B BIM modelinin LOD değeri, 4B modellemenin doğruluğunu ve kapsamını doğrudan etkilemektedir (Butkovic vd., 2019). LOD, bir BIM model elemanının belirli bir proje aşamasındaki detay ve doğruluk derecesini ifade etmektedir (AIA, 2022). Bir başka deyişle, 3B projelerde kullanılacak nesnelerin istenilen analizleri ve işlevleri sorunsuz şekilde gerçekleştirebilmesi için kullanılacak olan veri düzeyini göstermektedir (Bahadır, 2018). Amerikan Mimarlar Enstitüsü (American Institute of Architects, AIA) ve Amerikan Genel Müteahhitler Birliği (Associated General Contractors of America, AGC) tarafından LOD 100'den LOD 500'e kadar 6 adet LOD seviyesi tanımlanmıştır. LOD 100'de, modeldeki öğeler yalnızca temel geometriye sahip ve tasarım niyetini yansıtacak düzeyde soyutlanmaktadır. LOD 200'ye geçildiğinde, bu öğelerin yaklaşık boyutları, biçimleri ve

konumları belirlenmekte; ancak hâlâ kesin bilgiler yerine yaklaşık değerler kullanılmaktadır. LOD 300, “detaylı tasarım” aşamasını temsil ederek, öğelerin tam boyutlarını, malzemelerini ve konumlarını içermekte; fakat imalat veya montaj ayrıntıları tam olarak belirtememektedir. LOD 350 aşaması, inşaat dokümantasyonuna yönelik bilgi zenginliğini artırarak montaj ve yapı bütünlüğüne ilişkin daha spesifik veriler sunmaktadır. LOD 400 ise “imalat ve montaj” odaklı olarak, modelde tüm bileşenlerin imalat, kurulum ve montaj gereksinimlerini eksiksiz içerecek şekilde ayrıntılandırılmasını sağlamaktadır. Son olarak, LOD 500 düzeyinde model öğeleri, sahadaki gerçek uygulamayı yansıtacak biçimde uygulama sonrası verilerini ve işletme-bakım süreçlerine ilişkin bilgileri barındırmakta ve böylece yapı ömrü boyunca güncel ve kesin bir veri kaynağı oluşturmaktadır. Şekil 6’da bahsedilen LOD seviyeleri bir duvar örneği ile gösterilmektedir.



Şekil 6. LOD seviyeleri (BIMForum, 2024)

Farklı yapım aşamaları ve planlama gereksinimleri için, BIM modelinde kullanılan elemanların LOD seviyelerinin farklı olması gerekebilmektedir (Boton vd., 2015). Örneğin, erken tasarım aşamasında LOD 200, saha uygulamasında LOD 400 kullanımı söz konusu olduğunda, 4B simülasyonların doğruluğu ve detay seviyesi de buna bağlı olarak değişmektedir. Firmanın veya proje ekibinin ana yüklenici konumunda olması hâlinde, tasarım ofisinden gelen modelin LOD seviyesi yeterli değilse, modeli yeniden düzenlemeleri veya eklemeler yapmaları gerekebilir. Böyle bir dönüşüm süreci ise kaynak planlamasını ve zamana bağlı çizelge hazırlıklarını güçleştirebilmekte ve yüklenici firmaların adam-saat maliyetlerini artırabilmektedir (Sigalov & König, 2017). Farklı yapım süreçleri için gerekli olan LOD seviyeleri ve 4B simülasyon ilişkisinin önemi ve uygulama aşamasında yaşanılan sorunlar bilinmesine rağmen, literatürde az çalışılan bir konu olduğu da belirtilmektedir (Butkovic vd., 2019).

ABD GSA tarafından, BIM modelinin farklı LOD seviyeleri ile neler yapabileceğini anlamak için bir kılavuz yayınlanmıştır. Burada 4B BIM modeli ile birlikte; LOD 100'de, genel proje süresi ve ana elemanların temel aşamalandırılmasına dair kabaca bir zaman planlaması sunulabildiği, LOD 200'de büyük faaliyetlerin zaman ölçekli ve sıralı görünümünün netleştiği, LOD 300'de montaj aşamalarını daha ayrıntılı ele alarak sahadaki koordinasyonun arttığı, LOD 400'de üretim ve montaj detaylarını (ör. vinç, manlift vb.) içeren kapsamlı bir 4B simülasyonun oluşturulduğu, LOD 500'de ise işletme ve bakım verilerin eklenmesine odaklandığından, zaman boyutlu planlamanın bu aşamada pratik olarak uygulanmadığı belirtilmektedir (Patel, 2025).

Tüm bu veriler ışığında, 4B modellemenin BIM içerisinde önemli bir yapı taşı konumunda olduğu, ancak sektörün parçalı yapısı, yüksek eğitim maliyetleri, standartların tam oturmamış olması ve veri bütünlüğü eksikliği gibi etkenler nedeniyle henüz yaygın ve sistematik kullanıma tam olarak ulaşamadığı görülmektedir. Yine de 4B modelleme, tasarımdan uygulamaya, sürdürülebilirlikten maliyet kontrolüne kadar pek çok farklı alanda değer yaratma potansiyeline sahip olup özellikle zaman boyutunun devreye alınmasıyla birlikte, projelerin planlama ve kontrol aşamalarını belirgin biçimde iyileştirmektedir. Hem teknolojik altyapının geliştirilmesi hem de kurumsal ve örgütsel dönüşümün sağlanması hâlinde, 4B modellemenin inşaat sektöründe daha yaygın biçimde benimsenmesi ve daha başarılı projelerin hayata geçirilmesi öngörülmektedir.

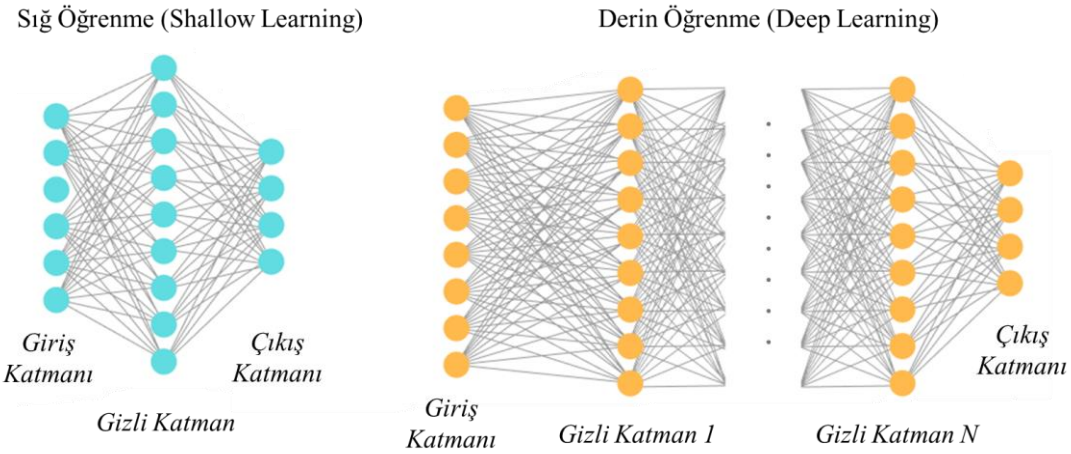
1.9. Makine Öğrenmesi ve Derin Öğrenme ile İlgili Yaklaşımlar

İnşaat sektöründe manuel emeğin yoğunluğu ve dijitalleşme oranının düşük olması; proje gecikmeleri, kaynakların verimsiz kullanımı, kalitesiz uygulamalar ve iş kazaları gibi çeşitli sorunlara yol açmaktadır (Chien vd., 2020). Sektörün büyüme potansiyeli, artan nüfus ve kentleşme baskısı nedeniyle oldukça yüksek olmakla birlikte, üretkenlik düzeyi çoğu zaman arzu edilen seviyeye ulaşamamaktadır (Kapelko & Abbott, 2017). Son on yılda yapılan çalışmalar, AI teknolojilerinin söz konusu sorunların üstesinden gelmek ve inşaat sahalarında daha güvenli, verimli ve sürdürülebilir bir yaklaşım geliştirmek amacıyla giderek daha fazla tercih edildiğini göstermektedir (Mrosla vd., 2019).

AI; bir bilgisayarın öğrenme, karar verme ve insanın düşünme biçimine benzer eylemleri yerine getirme kapasitesi olarak tanımlanmaktadır (Abioye vd., 2021). İnşaat alanında AI, karmaşık projelerin üstesinden gelmek ve yönetim süreçlerini kolaylaştırmak

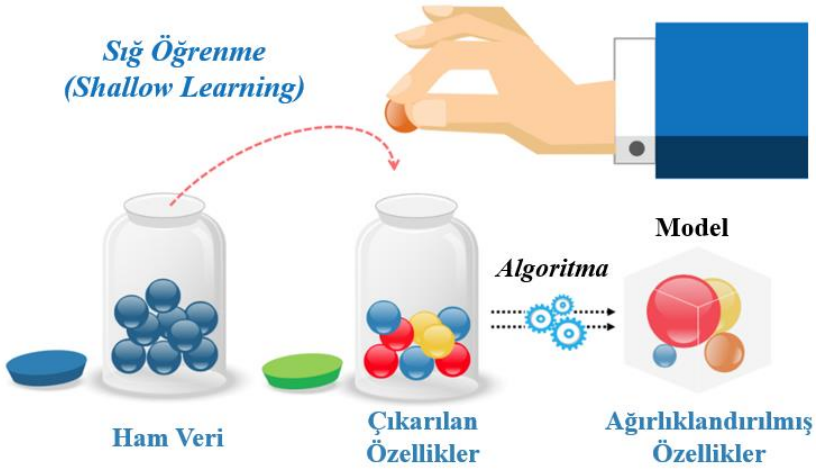
üzere bilgisayar tabanlı sistemlerin ve çeşitli algoritmaların kullanımını ifade etmektedir (Pan & Zhang, 2021). AI uygulamalarının merkezinde yer alan ML, karmaşık problemlerin çözümlenmesinde sunabileceği tahmin ve optimizasyon yetenekleriyle proje yönetiminden maliyet analizine, iş sağlığı ve güvenliğinden enerji verimliliğine kadar birçok alanda kullanılmaktadır (Garcia vd., 2022; Liu & Cao, 2024). Artan hesaplama gücü ve veri miktarı, ML modellerinin daha kapsamlı ve doğru çözümler sunabilmesine olanak tanımakta ve böylece inşaat projelerinin sürdürülebilir ve verimli bir şekilde yürütülmesini desteklemektedir (Datta vd., 2024). Bu sebeple ML yaklaşımları; proje yönetimi, kontrol ve teslimat gibi kritik alanlarda karar destek aracı olarak benimsenmiştir (Mostofi vd., 2025). Ayrıca, dijital ikiz ve BIM gibi yaklaşımların makine öğrenmesiyle bir araya gelmesi, projelerin erken aşamalarında oluşabilecek gecikme veya hataları belirleyerek müdahale etme olanağını artırmakta ve bu sayede proje yönetim süreci daha şeffaf ve izlenebilir hâle gelmektedir (Sacks vd., 2020). Literatürde inşaat izleme, maliyet ve iş programı optimizasyonu gibi konularda ML uygulamalarının potansiyeli giderek artmakla birlikte, bu potansiyelin henüz tam anlamıyla keşfedilmediği de belirtilmektedir (Mostofi, 2024).

ML, verilerdeki örüntüleri istatistiksel ve hesaplamalı yöntemlerle keşfederek bu örüntülerden tahmin veya karar mekanizmaları geliştiren bir yaklaşım olarak tanımlanabilmektedir (Mittal vd., 2023). ML modelleri, inşaat süreçlerinde farklı veri tiplerini analiz ederek potansiyel riskleri öngörmeye, kaynak planlamasını iyileştirmeye ve süreçleri otomatikleştirmeye olanak tanımaktadır (Garcia vd., 2022). ML yaklaşımları, model yapılarına göre “sığ öğrenme (Shallow Learning, SL)” ve “derin öğrenme (Deep Learning, DL)” olarak iki ana kategoriye ayrılmaktadır. SL, tek bir gizli katman içerir ve nispeten daha basit mimarileriyle hızlı çözümler üretebilirken; DL, çok sayıda gizli katmana sahip olup karmaşık verilerin işlenmesinde daha güçlü bir kapasiteye sahiptir (Rahman & Azad, 2021; Xu vd., 2021). Şekil 7’de SL ve DL’nin ağ yapıları basitçe gösterilmektedir. Bu farklı yapılar, inşaat projelerinin doğasına, veri hacmine ve çözülmek istenen problemin türüne bağlı olarak ML yöntem seçimini etkilemektedir.



Şekil 7. Sığ ve derin öğrenme ağ yapıları

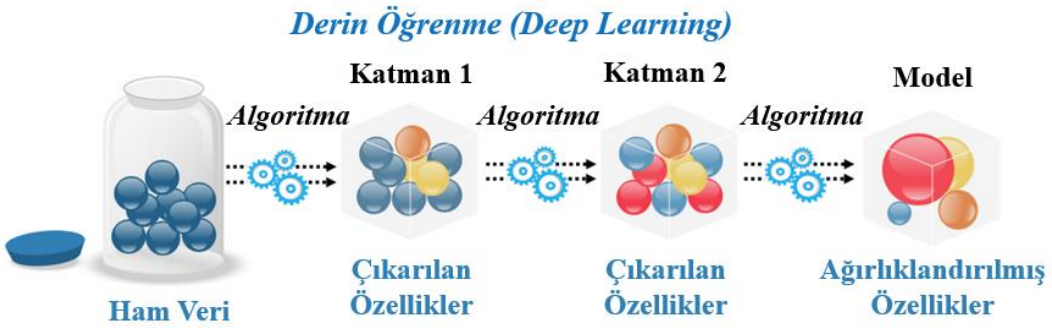
SL, sınırlı katman derinliği ile nispeten düşük karmaşıklığa sahip modelleri içeren bir ML yaklaşımı olarak öne çıkmaktadır (Mittal vd., 2023). Lojistik regresyon, destek vektör makineleri veya karar ağaçları gibi geleneksel ML yöntemleri SL kapsamına girmektedir (Xu vd., 2021). Parametre sayısı görece olarak az olup, tipik olarak el ile özellik mühendisliği (manual feature engineering) aşamasına dayalıdır (Jafari vd., 2024). SL modelleri; hızlı eğitim, daha düşük hesaplama maliyeti ve çıktıların yoruma elverişli olması nedeniyle özellikle küçük veya orta ölçekli veri kümelerinde tercih edilmektedir (Pasupa, & Sunhem, 2016). Bununla birlikte, karmaşık ve çok boyutlu verilerle karşılaşıldığında sığ yapılar, sınırlı temsil gücü nedeniyle DL yöntemlerine kıyasla yeterli performansı sağlayamayabilir (Xu vd., 2021). Şekil 8'de SL yaklaşımının temel modelleme süreci gösterilmektedir.



Şekil 8. Sığ öğrenme (Mahmoud, 2024)

Şekil 8'de gösterildiği gibi, ilk aşamada ham veri toplanarak, modellenmeye doğrudan uygun olmayan veriler anlamlı bir biçimde dönüştürülür veya el ile seçilen özelliklere (öz niteliklere) ayrıştırılır. Bu adım, gereksiz bilgilerin elenmesi ve çözümleme açısından önemli parametrelerin belirlenmesi bakımından kritik öneme sahiptir. Sonraki aşamada, çıkarılan özellikler seçilen ML algoritmasına beslenerek model eğitimi gerçekleştirilir. Bu süreçte modelin parametreleri verilerin içindeki örüntüleri anlamlandıracak şekilde ayarlanır. Eğitimin tamamlanmasının ardından model, elde edilen ağırlıklar ve önem dereceleri üzerinden yeni veriler üzerinde hızlı ve tutarlı tahminler veya kararlar üretmeye hazır hâle gelmektedir.

DL, çok katmanlı sinir ağları üzerinden daha karmaşık ve doğrusal olmayan örüntüleri yakalamayı hedefleyen bir ML alt disiplini olarak tanımlanmaktadır (Rahman & Azad, 2021). Evrişimsel sinir ağları (Convolutional Neural Networks, CNN), RNN ve üretici çekişmeli ağlar (Generative Adversarial Networks, GAN) gibi farklı mimariler, çok katmanlı yapılarda her aşamada giderek daha soyut seviyede özellikler çıkararak görüntü tanıma, doğal dil işleme ve konuşma tanıma gibi alanlarda üstün başarı sağlamaktadır (Xu vd., 2021). Bu modellerin otomatik özellik çıkarma kabiliyeti, geniş veri kümelerinde insan müdahalesini asgariye indirerek yüksek performanslı sonuçlar üretmelerine imkân tanımaktadır (Mittal vd., 2023). Ancak, eğitimin maliyetli oluşu, geniş etiketli veri gereksinimi ve çıktıların anlaşılma düzeyinin düşük olması, derin öğrenmenin belirgin sınırlılıkları arasında yer almaktadır (Pasupa, & Sunhem, 2016). Şekil 9'da DL yaklaşımının temel modelleme süreci gösterilmektedir.



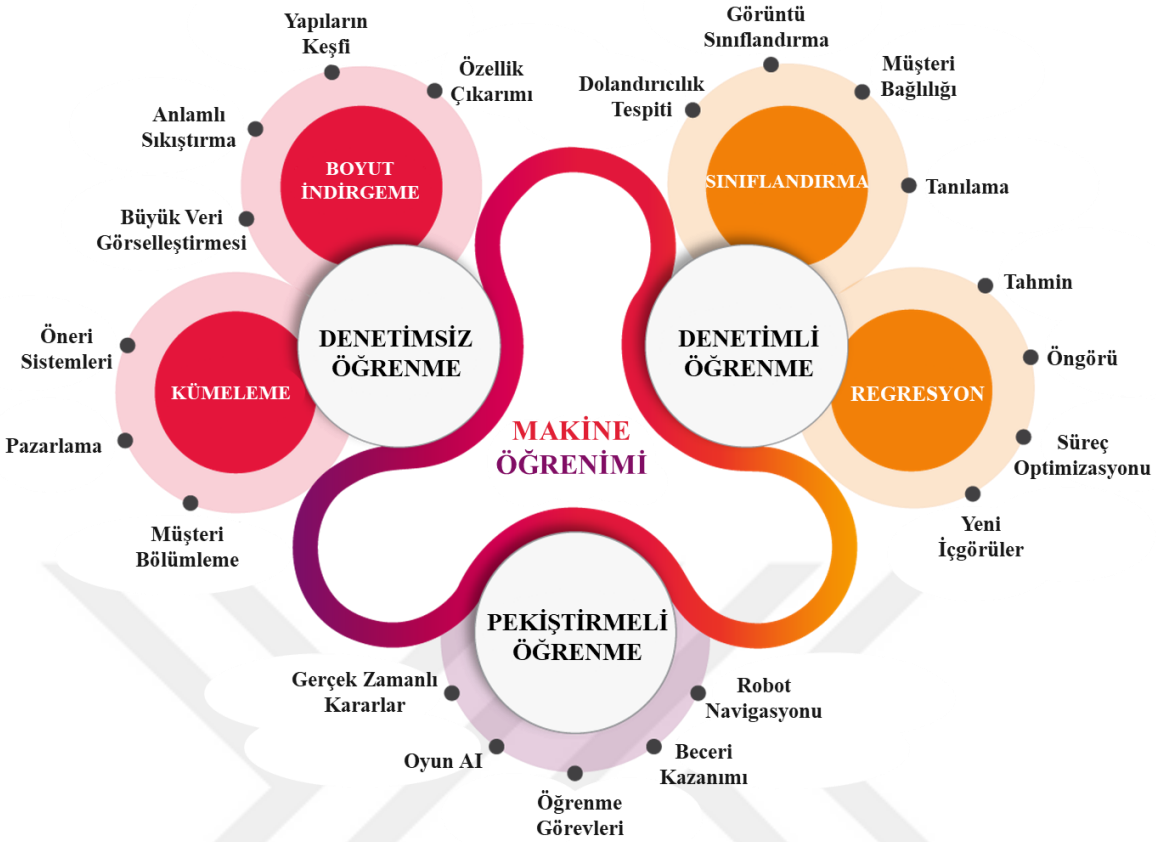
Şekil 9. Derin öğrenme (Mahmoud, 2024)

Şekil 9, çok katmanlı bir derin öğrenme mimarisinde ham verinin katmanlar boyunca adım adım işlenerek giderek daha soyut özellik temsillerine dönüştürülmesini anlatmaktadır. İlk aşamada "ham veri", modelin birinci katmanına girmekte ve temel veya

düşük seviyeli özellikler çıkarılmaktadır. Bu özellikler, ikinci katmana aktarılmakta ve bu kez önceki katmanda elde edilen bilgiler birleştirilerek daha karmaşık veya daha soyut düzeyde temsiller oluşturulmaktadır. Aynı mantık, çok sayıda katman içeren derin ağ boyunca sürer ve her katmanda, verinin önceki katmanlardan aktarılan temsili, giderek üst düzey kavramlara veya örüntülere dönüştürülür. Katmanlardan geçerek rafine edilen bu veriler "model" aşamasına ulaşır ve bu noktada öğrenilen ağırlıklar (hangi özelliğin ne kadar önemli olduğu) kullanılarak tahmin, sınıflandırma veya karar verme işlemleri gerçekleştirilir. Bu süreç, her ek katmanla birlikte özelliklerin daha yüksek soyutlama düzeylerine ulaşmasını ve özellikle karmaşık veri türlerinde daha etkili sonuçlar elde edilmesini sağlamaktadır.

Projelerde sığ veya derin öğrenme tercihinin, yalnızca model büyüklüğü yerine veri setinin boyutu, problem karmaşıklığı ve mevcut hesaplama imkânları gibi etkenler göz önüne alınarak yapılması gerekmektedir. Dolayısıyla, hangi yaklaşımın seçileceği, projenin hedeflerine ve kısıtlarına en uygun yöntemi belirlemek amacıyla çok yönlü bir değerlendirmenin sonucunda netleştirilmelidir (Mazuera-Rozo vd., 2021).

ML, veri etiketlerinin mevcudiyeti ve öğrenme hedefleri doğrultusunda genellikle denetimli öğrenme (supervised learning), denetimsiz öğrenme (unsupervised learning), yarı denetimli öğrenme (semi-supervised learning) ve pekiştirmeli öğrenme (reinforcement learning) olmak üzere dört temel kategoriye ayrılmaktadır (Alkhaleel, 2024; Xu & Saleh, 2021). Şekil 10'da bahsedilen ML türleri ile bu kategoriler altında yer alan alt yöntemlerin şematik bir gösterimi verilmektedir.



Şekil 10. ML türleri ve alt yöntemler (URL-3, 2024)

Denetimli öğrenme, verinin önceden etiketlenmiş olduğu ortamlarda model geliştirmeye odaklanan bir ML yaklaşımıdır (Mittal vd., 2023). Bu yöntemde model, etiket bilgisi sayesinde tahmin veya sınıflandırma görevlerinde oldukça başarılı sonuçlar üretebilmektedir (Jiang vd., 2020). Literatürde, sağlık hizmetlerinden (Gu vd., 2023) finansal uygulamalara (Lei vd., 2022), görüntü ve ses işleme alanlarından (Tripathi vd., 2021; Aljuaid & Anwar, 2022) doğal dil işleme (Nagarhalli vd., 2021), üretim ve endüstriyel süreçler (Wuest vd., 2014) ile genetik (Abdulqader vd., 2020) ve çevre bilimlerine (Tahmasebi vd., 2020) kadar geniş bir çerçevede denetimli öğrenmenin kullanıldığı görülmektedir. Denetimli öğrenme, veri işleme şekline göre sınıflandırma ve regresyon olmak üzere iki ana alt kategoriye ayrılır (Nasteski, 2017). Sınıflandırma, veri örneklerini önceden tanımlanmış kategorilere ayırmaya yönelir (Kadhim, 2019) ve yapay sinir ağları (Artificial Neural Network, ANN), K-en yakın komşu (K-Nearest Neighbors), destek vektör makineleri (Support Vector Machines, SVM), Naive Bayes, karar ağaçları (Decision tree) gibi algoritmalarla gerçekleştirilir (Bhavsar & Ganatra, 2012; Choudhary & Gianey, 2017; Crisci vd., 2012; Sharpe vd., 2019; Arabameri vd., 2021). Regresyon ise

bağımsız değişkenlerle bağımlı değişkenler arasındaki ilişkiyi modelleyerek tahmin etmeye odaklanır (Xu vd., 2022; Jiang vd., 2020) ve bu amaçla doğrusal (Linear Regression), polinom (Polynomial Regression), rastgele orman (Random Forest Regression) veya destek vektör (Support Vector Regression) gibi yöntemler yaygın biçimde kullanılmaktadır (Bonaccorso, 2018; Doan & Kalita, 2015).

Denetimsiz öğrenme, etiketlenmemiş veri kümelerindeki gizli örüntüleri ve ilişkileri keşfetmeyi hedefleyen bir ML yaklaşımıdır (Schmarje vd., 2021). Bu yöntem, verinin doğal yapılarını ortaya çıkarmak amacıyla insan gözetimi olmaksızın çalışmakta ve sosyal medya pazarlaması (Sánchez-Hernández vd., 2013), tıbbi uygulamalar (Eckhardt vd., 2023), finans (Hoang & Wiegratz, 2023), siber güvenlik (Alom & Taha, 2017) ve görüntü tanıma (Wang vd., 2016) gibi geniş bir yelpazede kullanılmaktadır. Denetimsiz öğrenme, kümeleme ve boyut indirgeme olarak iki ana kategoriye ayrılmaktadır (Althuwaynee vd., 2021). Kümeleme yaklaşımı, veri noktalarını benzerlik temelli gruplara ayırarak ve birliktelik kuralları çıkartarak pazarlama, müşteri segmentasyonu, görüntü tanıma, satış stratejileri veya öneri sistemleri gibi alanlarda önemli içgörüler sunmaktadır (Mahadevkar vd., 2022). Bu yaklaşımda K-ortalama, Hiyerarşik kümeleme, Apriori ve FP-Growth algoritmaları yaygın olarak kullanılmaktadır (Althuwaynee vd., 2021). Boyut indirgeme, yüksek boyutlu verileri yönetilebilir formata dönüştürerek hem görselleştirmeyi kolaylaştırır hem de model eğitim süresini azaltır (Pekşen, 2024). Bu bağlamda PCA (Principal Component Analysis) ve t-SNE (t-Distributed Stochastic Neighbor Embedding) gibi teknikler sıklıkla tercih edilmektedir. Bu çeşitlilik, denetimsiz öğrenmenin etiketlenmemiş verilerden anlamlı bilgiler çıkarmak ve karar mekanizmalarını güçlendirmek için kritik bir rol oynadığını göstermektedir (Anowar vd., 2021).

Yarı denetimli öğrenme, kısıtlı etiketli veriyle birlikte çok daha geniş etiketsiz veri gruplarını kullanarak modelin performansını artırmayı amaçlayan bir yaklaşım olarak öne çıkmaktadır (Forestier & Wemmert, 2016). Bu yöntem, denetimli ve denetimsiz tekniklerin avantajlarını birleştirerek özellikle görüntü sınıflandırma ve doğal dil işleme gibi alanlarda etkin şekilde uygulanmaktadır (Hagberg vd., 2022). Yarı denetimli öğrenmenin bu esnek yapısı, etiketlenmiş veri miktarının sınırlı olduğu durumlarda dahi başarılı modeller geliştirilebilmesine imkân tanımaktadır (Van Engelen & Hoos, 2020).

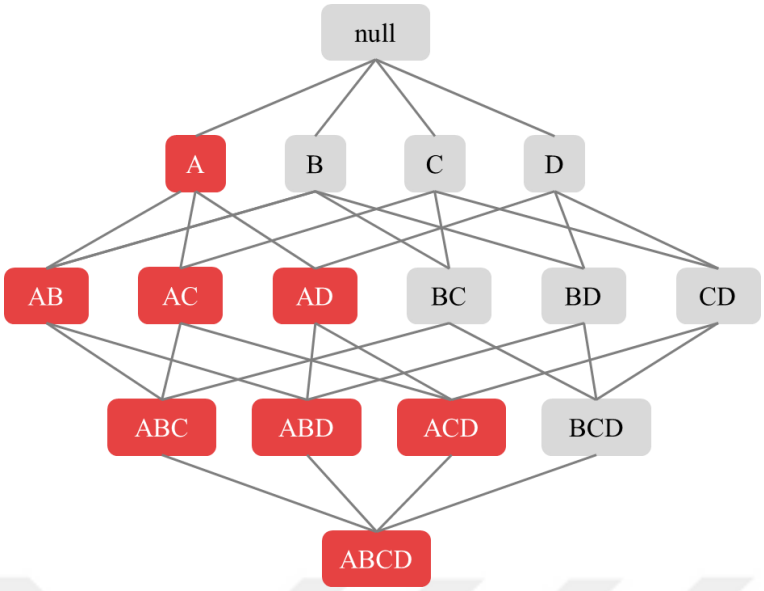
Pekiştirmeli öğrenme, davranışçılık kuramından esinlenen ve bir sistemin, içinde bulunduğu ortamda en yüksek ödülü elde edebilmek için hangi eylemleri gerçekleştirmesi gerektiğini ortaya koymaya odaklanan bir ML yaklaşımıdır (Karakuş, 2023). Bu yaklaşım,

çevrimiçi çok oyunculu oyunlardan (Xin vd., 2022) robot eğitimine (Ibarz vd., 2021) ve otonom sürüş uygulamalarına (Kiran vd., 2021; Shan vd., 2020) kadar geniş bir yelpazede kullanılmaktadır. Ayrıca, üretim ve operasyon yönetimi süreçlerinde karar verme mekanizmalarını optimize etmek amacıyla da sıkça tercih edilmektedir (Kayhan & Yildiz, 2023).

Her bir ML türü farklı avantaj ve dezavantajlara sahiptir. Denetimli öğrenme yüksek doğruluklu tahminler sunabilmekle birlikte etiketlenmiş veri ihtiyacı nedeniyle maliyetli olabilirken, denetimsiz öğrenmede etiketlenmemiş veriler üzerinde çalışabilme esnekliği bulunmakla beraber, elde edilen sonuçların yorumlanması zaman alabilmektedir (Alloghani vd., 2020). Yarı denetimli öğrenme, etiketli verinin az olduğu senaryolarda büyük fırsatlar sunarken, pekiştirmeli öğrenme uzun ve maliyetli eğitim döngüleri gerektirmektedir (Pekşen, 2024). Bu farklı yöntemler arasındaki seçim, sorunun türüne, veri yapısına ve kullanılabilecek veri miktarına göre belirlenmektedir (Alloghani vd., 2020). Dolayısıyla, inşaat sektörü için en uygun ML yaklaşımının seçimi, proje gereksinimlerini ve veri mevcudiyetini dikkate alan çok boyutlu bir değerlendirme süreci gerektirmektedir. Sonraki bölümlerde bu tez çalışmasında kullanılan ML teknikleri açıklanmaktadır.

1.9.1. Birliktelik Kural Madenciliği (ARM)

ARM, büyük veri kümelerindeki öğeler arasında gizli kalıpları ve ilişki kurallarını keşfetmeye odaklanan bir veri madenciliği tekniği olarak tanımlanmaktadır (Kamsu-Foguem vd., 2013; Liu vd., 2020). Bu yaklaşım, çoğunlukla denetimsiz ML kapsamına girmekte olup analizde eşzamanlılığı ve koşullu olasılıkları dikkate alarak veri ögeleri arasındaki bağlantıları "X → Y" biçiminde kural setleri hâlinde sunmaktadır (Mou vd., 2020). ARM süreçlerinde en yaygın kullanılan yöntemlerden biri olan Apriori algoritması, başlangıçta en sık rastlanan öğe kümelerini belirlemekte, ardından öncül (X) ve sonuç (Y) ilişkilerini tanımlayarak bu kuralların destek (support) ve güven (confidence) değerlerine göre önem sıralamasını yapmaktadır (Pearl, 2009). Tek öğelerle başlayarak, kademeli olarak daha büyük öğe kümelerini araştırır ve gerekli destek seviyesinin altında kalanları öğeleri kaldırır. Şekil 11'de Apriori algoritmasının çalışma prensibi 4 öğe ile örnek olarak gösterilmektedir.



Şekil 11. Apriori algoritması

Şekil 11'de gösterildiği gibi, en üstte boş bir öğe kümesiyle başlanıp, önce tek öğeler (A, B, C, D) oluşturulur ve bu aşamada yeterli sıklığa sahip olmayan öğeler elenir. Ardından geriye kalan tek öğeler farklı kombinasyonlar hâlinde birleştirilerek iki öğeli kümeler (örneğin AB, AC, AD vb.) oluşturulur. Eğer bu iki öğeli kümelerden bazıları belirlenmiş "destek" ölçütünün altındaysa, o kümeler de elenir. Daha sonra elenmeden kalan çiftlerden, üç öğeli kümeler (örneğin ABC, ABD vb.) türetilir ve aynı eleme prensibi uygulanır. Son olarak yine yeterli sıklığı koruyabilen üçlüler, daha büyük öğe kümelerine genişletilir. Bu şekilde, katmanlar halinde ilerleyen Apriori algoritması, hem sık öğe kümelerini hem de onlardan türetilen olası kuralları (X → Y) çıkarmayı sağlar. Şekilde kırmızıyla vurgulanan düğümler, yeterli sıklığı sağlamış ve sonraki aşamaya taşınmaya uygun bulunan öğe kümelerini göstermektedir.

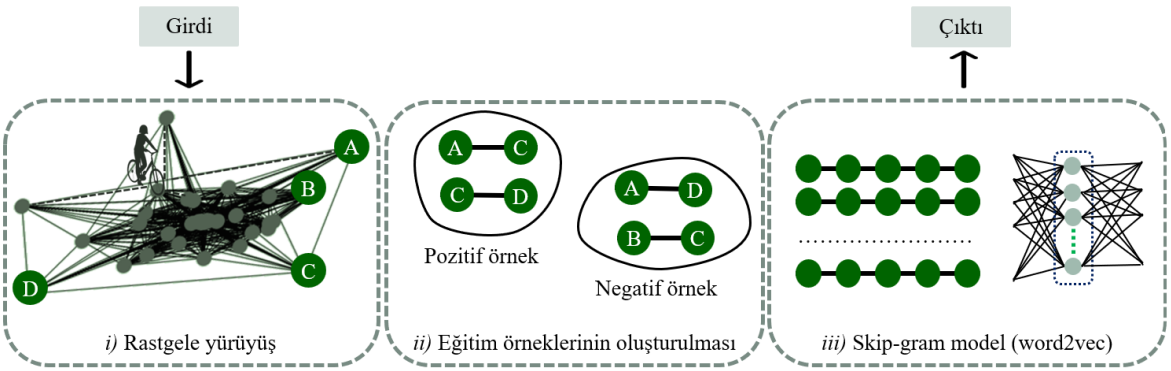
Literatür incelendiğinde, inşaatta karşılaşılan kusurların nedenlerini ve bu kusurların birbirleriyle olan ilişkilerini belirlemeyi amaçlayan çok sayıda çalışma yer almaktadır (Mostofi vd., 2023a; Ayhan vd., 2020; Lee vd., 2019). Bu çalışmalarda, ARM yaklaşımının, bir kusurun meydana gelmesi durumunda hangi kusurun sonraki aşamada ortaya çıkma olasılığının yüksek olduğunu öngörme kapasitesine sahip olduğu belirtilmektedir (Cheng vd., 2015; Lin & Fan, 2018). Böylece ARM özellikle inşaat projelerinde, kusur önleme ve kalite yönetimi stratejilerinin geliştirilmesi için önemli bir karar destek mekanizması sağlamaktadır (Fan, 2020). Bu çalışmaların yanı sıra, sismik fay tespitinde (Liu vd., 2020), inşaat kazaları arasındaki korelasyonların tespit edilmesinde

(Verma vd., 2014; Ayhan vd., 2020), işyeri tehlikelerinin belirlenmesinde (Wang vd. 2018), asfalt çatlama göstergelerinin tespitinde (Dong vd. 2018) ve güvenli olmayan işçi davranışlarının zaman dağılımlarını incelemede (Liao vd., 2019) ARM tabanlı yöntemlere başvurulduğu görülmektedir.

1.9.2. Node2vec

Node2vec, ağ verilerinden anlamlı düğüm temsilleri (embedding) çıkarmayı amaçlayan ve yarı-denetimli öğrenme yaklaşımı tabanlı, verimli ve ölçeklenebilir bir yöntem olarak açıklanmaktadır (Grover & Leskovec, 2016; Alişan, 2024). Bu yaklaşım, her bir düğümü düşük boyutlu bir vektör uzayında temsil ederek, ağ yapısına dayalı görevlerin (örneğin, düğüm sınıflandırma, topluluk tespiti ya da bağlantı tahmini) daha etkin bir şekilde gerçekleştirilmesini hedeflemektedir (Kaczmarek, 2023; Rahman & Azad, 2021; Ahmed, 2023). Node2vec'in geliştirilme aşamasında, doğal dil işleme alanından esinlenilmiş olup, kelime temsillerini öğrenen word2vec yaklaşımının mantığı ağ verilerine uyarlanmıştır ve düğümler arasındaki ilişkilerin benzerlik prensipleri çerçevesinde modellenmesi sağlanmaktadır (Alaca, 2023).

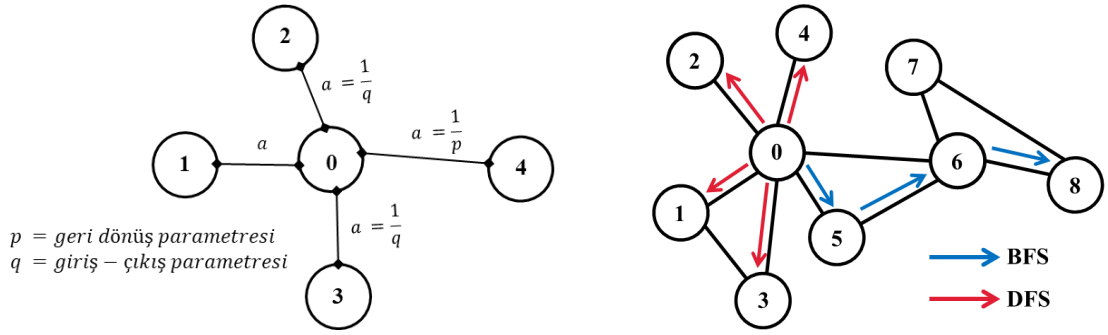
Şekil 12'de gösterildiği üzere, node2vec algoritması genellikle üç temel adımdan oluşmaktadır: (i) ön işleme aşamasında p ve q parametreleri dikkate alınarak her düğüm için belirli sayıda rastgele yürüyüş gerçekleştirilmesi, (ii) bu yürüyüşlerden pozitif ve negatif eğitim örneklerinin oluşturulması ve (iii) optimizasyon aşamasında, elde edilen diziler üzerinde skip-gram veya benzeri bir model ile düğüm temsillerinin öğrenilmesi (Mostofi vd., 2023a; Song vd., 2023).



Şekil 12. Node2vec algoritması

Node2vec, rastgele yürüyüş yöntemini skip-gram modeli ile bütünleştirerek, her düğümün komşu düğümleriyle olan etkileşimini istatistiksel olarak modelleyerek bir temsil öğrenmektedir (Palumbo vd., 2017; Goswell, 2020). Rastgele yürüyüşün taraflı (biased random walk) biçimde uygulanması, modelin hem yerel hem de küresel ağ özelliklerini daha etkili biçimde yakalamasına olanak tanımaktadır (Kaczmarek, 2023). Söz konusu taraflı yürüyüş stratejisi, p ve q olarak adlandırılan iki hiper parametre ile kontrol edilmektedir. Bu parametrelerden p, yürüyüşün önceden ziyaret edilen düğüme geri dönme olasılığını yönetirken; q ise başlangıç düğümünden uzaklaşma eğilimini belirlemektedir (Song vd., 2023).

Node2vec, ağlarda gözlemlenen farklı bağlantı yapılarını daha kapsamlı yakalayabilmek amacıyla genişlik öncelikli arama (Breadth-First Search, BFS) ve derinlik öncelikli arama (Depth First Search, DFS) yöntemlerini de kullanmaktadır (Kaczmarek, 2023). Bu süreçte, BFS düğümlerin kaynak düğüme yakın bölgelerini tarayarak yerel kümelenmeleri ortaya çıkarırken; DFS giderek uzaklaşan düğümleri araştırarak küresel yapıya ilişkin bilgiler sunmaktadır (Rahman & Azad, 2021). Dolayısıyla, p parametresi, düşük değerlerde BFS benzeri, yüksek değerlerde ise DFS benzeri davranış göstermektedir. Benzer biçimde, q parametresi için q>1 durumunda yürüyüş daha çok yerel komşuluğa odaklanırken, q<1 durumunda ağın daha uzak kısımları keşfedilmektedir (Grover & Leskovec 2016; Grohe, 2020; Salamat vd., 2020). Sonuç olarak, p ve q parametreleri doğru şekilde ayarlandığında, yerel kümelenmeler ile uzak düğümler arasındaki yapısal ilişkiler eş zamanlı olarak yakalanabilmektedir. Şekil 13'te taraflı rastgele yürüyüş ile BFS ve DFS yöntemlerinin çalışma ilkeleri gösterilmektedir.



(a) Taraflı rastgele yürüyüş (b) BFS ve DFS araştırma stratejileri

Şekil 13. (a) Taraflı rastgele yürüyüş ve (b) BFS ve DFS yöntemlerinin çalışma ilkeleri

Bu taraflı rastgele yürüyüşlerden elde edilen düğüm dizileri, word2vec yaklaşımının skip-gram modelinin hedef ve bağlam mantığına benzer şekilde işlenmektedir (Ahmed, 2023). Bu çerçevede, her bir düğüm için tanımlanan vektör temsili, ilgili düğümün komşularını (rastgele yürüyüşlerden üretilen dizideki yakın düğümleri) olabildiğince doğru tahmin edecek şekilde optimize edilmektedir.

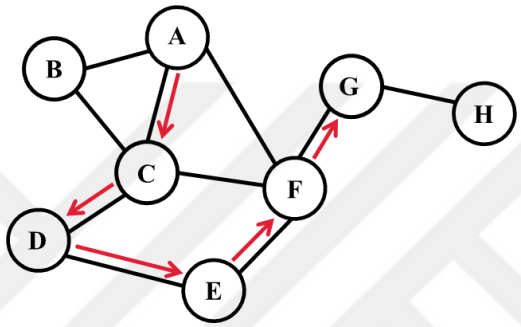
İnşaat mühendisliği alanında, özellikle kök neden analizi (root cause analysis) gibi kusur veya risk etkenlerini ortaya koymayı hedefleyen problemlerde node2vec yaklaşımından yararlanıldığı görülmektedir (Mostofi vd., 2023a). Bu yöntem, farklı kusurların birbiriyle ilişkisini, tekrarlayan hataların arka planını ve gelecekteki bağlantı eğilimlerini tahmin edebilecek bir çerçeve sunarak karar destek mekanizmalarına önemli katkılar sağlamaktadır (Grover & Leskovec, 2016). Parametre ayarlarının uygulama alanına özgü şekilde dikkatlice yapılması gerekmesine karşın, rastgele yürüyüş ve skip-gram modelinin birleşimiyle yerel ve küresel ağ yapısına aynı anda odaklanabilme becerisi, node2vec'i geniş ölçekli grafik veri kümeleri üzerinde gerçekleştirilen çok yönlü analizlerde oldukça değerli bir araç hâline getirmektedir (You vd., 2021).

1.9.3. DeepWalk

DeepWalk, node2vec'e benzer şekilde, word2vec'in rastgele yürüyüş ve yöntemlerini kullanarak bir grafikteki düğümler için temsiller oluşturan denetimsiz bir algoritma olarak tanımlanmaktadır (Perozzi vd., 2014; Kaczmarek, 2023). Bu yöntemin temelini, algoritmanın ağ üzerindeki her bir düğümden başlatarak ürettiği rastgele yürüyüşler oluşturmaktadır (Bandyopadhyay vd., 2018). Söz konusu yürüyüşler, skip-gram modeli aracılığıyla işlenmekte ve düğümler, kelimelerden oluşan cümleler gibi ele alınmaktadır (Xu vd., 2019). Bu bağlamda, algoritma word2vec yaklaşımını kullanarak oluşturulan yürüyüşlerdeki komşuluk bilgisine dayanarak düğümler için vektör temsillerini öğrenmektedir. Node2vec ile karşılaştırıldığında, DeepWalk'un en önemli farkı, rastgele yürüyüşlerin üretilmesi esnasında yerel ve küresel ağ yapısının keşfini kontrol etme imkânı tanımamasıdır (Song vd., 2023). Bu sebeple, DeepWalk genellikle düğümlerin yerel bağlamına daha yoğun bir biçimde odaklanma eğilimindedir (Kaczmarek, 2023).

DeepWalk, esas itibarıyla rastgele yürüyüş, eğitim örneklerinin oluşturulması ve güncelleme prosedürü olmak üzere üç bölümden oluşmaktadır (Soni vd., 2019). Rastgele yürüyüşler, düğümler arasındaki benzerliklerin ve olası topluluk üyeliklerinin tahminine

olanak sağlamaktadır (Bozorgi vd., 2025). Şekil 14'te gösterilen tarafsız rastgele yürüyüş sürecinde, algoritma ilk olarak bir düğümden başlamakta ve ardından komşu düğümlerden birini seçmektedir. Bu süreç, belirlenen adım sayısı boyunca tekrarlanmaktadır. Bu işlemin sonucu olarak ziyaret edilen düğüm dizilimi elde edilmekte ve bu dizilim, pozitif örneklerin oluşturulması için kullanılmaktadır. Daha sonra, Skip-gram algoritması yardımıyla yapılan güncellemeler, elde edilen veri üzerinde uygulanmaktadır (Chen vd., 2022). Bu yaklaşım, rastgele yürüyüşleri kullanmayan yöntemlere kıyasla daha üstün bir performans sergilemektedir (Bozorgi vd., 2025).



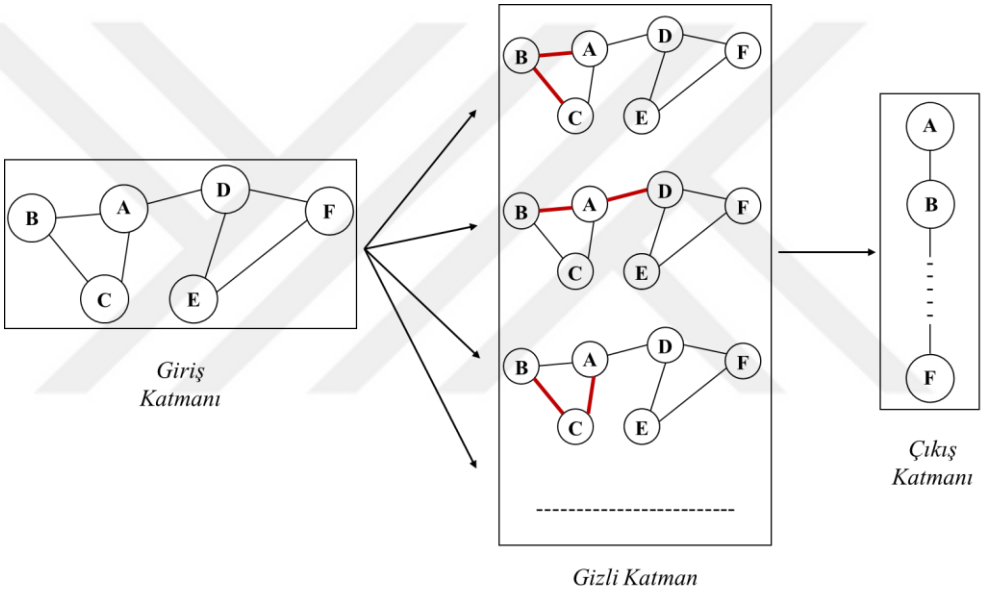
Şekil 14. Tarafsız rastgele yürüyüş

Bir grafiğin dolaşım biçimi, elde edilen düğüm yerleştirmeleri üzerinde belirleyici bir etkiye sahiptir. DeepWalk, node2vec'in aksine tarafsız rastgele yürüyüş yaklaşımını kullanmaktadır. Tarafsız rastgele yürüyüşte, her komşu düğümün ziyareti sırasında seçilme olasılığı eşit kabul edilmektedir (Rahman & Azad, 2021). Bu yöntem, grafiğin yapısıyla ilgili ayrıntılı bilgilerin elde edilmesine imkan tanımakta ve tüm düğümlerin eşit olarak değerlendirilmesi prensibine dayanmaktadır. Böylece, düğümler arasındaki hem yerel hem de küresel bağlantıların analizine imkan sağlanır (Bozorgi vd., 2025). Her iki yaklaşımın da araştırma hedeflerine bağlı olarak çeşitli avantaj ve dezavantajları bulunmaktadır. Tarafsız yürüyüş, grafik yapısının kapsamlı bir genel görünümünü sunarken; taraflı yürüyüş, belirli grafik özelliklerini daha derinlemesine inceleme fırsatı sağlamaktadır (Kaczmarek, 2023).

1.9.4. Grafik Evrişimli Ağlar (GCN)

Grafik tabanlı verilerden anlamlı temsiller çıkarmayı amaçlayan GCN, Kipf & Welling (2016) tarafından ölçeklenebilir yarı denetimli bir yaklaşım olarak kabul

edilmektedir. GCN, özetle, geleneksel evrişim katmanlarının grafik yapılı verilere uyarlanmasıyla geliştirilmiştir (Kipf & Welling, 2016). Bu yöntem, düğümlerin kendi öznitelikleriyle komşu düğümlerden gelen bilgileri çok katmanlı bir yapı içinde yinelemeli şekilde birleştirir (Chen vd., 2021). Bu çok katmanlı evrişim mekanizması, her bir katmanda düğüm özelliklerini ve komşuluk ilişkilerini normalleştirilmiş bitişiklik matrisini kullanarak günceller (Shaygan vd., 2022). Matematiksel ifade olarak, l-inci katmandan (l+1)-inci katmana geçiş A matrisinin normalleştirilmesi ve katman ağırlıklarının uygulanmasıyla gerçekleşmektedir (Kipf & Welling, 2016). GCN prosedürü Şekil 15'te gösterilmektedir.



Şekil 15. GCN prosedürü

Şekil 15'te ilk olarak grafiğin bitişiklik yapısı ve düğüm öznitelikleri modele girdi olarak alınır ve ardından her bir katmanda düğümün kendi öznitelikleri ile doğrudan komşu düğümlerinden toplanan bilgiler belirli bir normalizasyon ve ağırlıklandırma işlemiyle birleştirilir. Böylece, (l+1)-inci katmanda, l-inci katmandan gelen düğüm temsilleri normalleştirilmiş bitişiklik matrisi yardımıyla güncellenir ve yeni katman ağırlık matrisi uygulanarak daha soyut ve anlamlı düğüm öznitelikleri elde edilir. Çok katmanlı bu işlem adım adım tekrarlandığında, hem düğümlerin kendi özelliklerini hem de komşuluk yapısını hesaba katarak grafiğin geneline dair daha zengin bir temsil inşa edilmiş olunur (Shaygan vd., 2022).

Alandaki güçlü ve popüler yöntemlerden biri olarak tanımlanan GCN'nin (Cui vd., 2024), çok sayıda pratik problemdeki ilişki ağlarını modelleme kapasitesi sayesinde inşaat yönetimi, üretim planlama ve güvenlik risk tespiti gibi konularda kapsamlı biçimde ele alındığı görülmektedir (Mostofi vd., 2023b; Mostofi vd., 2022; Mostofi & Toğan, 2024). Bu çerçevede GCN; düğümlerin ve onların komşularının özelliklerini katmanlar boyunca yeniden işler, toplar ve temsil uzayında güçlü bir öznitelik öğrenme yeteneği sergiler. Böylelikle, grafik yapılı bilgilerin modellemesinde yüksek performanslı ve açıklanabilir tahminleme imkânı sunar (Mostofi & Toğan, 2024).

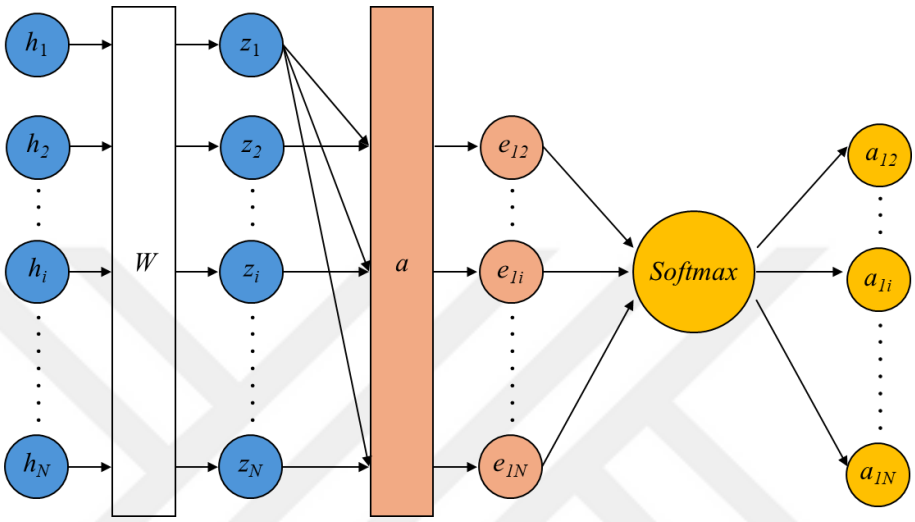
Birçok çalışma, GCN yönteminin sosyal öneri sistemlerinde (Wu vd., 2018), bilimsel atıf ağlarında (Abu-El-Haija vd., 2019), sosyal ağ analizlerinde (Chen vd., 2020) ve inşaat süreçlerindeki maliyet tahmininde de (Mostofi vd., 2024) etkili olduğunu ortaya koymuştur. Ayrıca, karar verme mekanizmalarına katkı sağlama (Mostofi vd., 2023b) ve inşaat güvenliği tahmini (Pan vd., 2022) gibi pratik uygulamalarda da başarılı sonuçlar elde edildiği bildirilmektedir.

1.9.5. Grafik Dikkat Ağları (GAT)

GAT, GCN ile birlikte yaygın olarak kullanılan grafik sinir ağıdır (Graph Neural Networks, GNN) (Khemani vd., 2024). GAT, grafik verilerinin işlenmesinde düğümler arasındaki bağımlılıkları değişken önem dereceleriyle modelleyebilmek amacıyla geliştirilmiş bir DL yaklaşımıdır (Veličković vd., 2018). Klasik GNN'den farklı olarak GAT, düğümler arasındaki etkileşimleri sabit komşuluk ilişkileri üzerinden değil, öğrenilebilir dikkat katsayıları üzerinden değerlendirir (Vrahatis vd., 2024). Böylelikle, her bir düğümün komşularına olan ilgisi modele dinamik bir biçimde kazandırılmış olur. Bu mekanizma, grafik üzerinde zamanla veya bağlama göre farklılaşan ilişkileri yakalamada önemli bir esneklik sağlar (Mostofi vd., 2025).

GAT modelinde ilk olarak, düğüm ve bağlantı bilgileri kullanılarak girdi ağı işlenmekte ve böylece düğümler arası bağlantılar tanımlanarak grafik veri seti düzenlenmektedir. Ardından, girdi katmanının oluşturulmasıyla birlikte her düğüme ait öznitelikler ve düğümler arasındaki bağlantı kriterleri tanımlanır. Bu katman, temel veriyi modelin geri kalan aşamalarında kullanılabilecek şekilde yapılandırır. Bir sonraki adım, Şekil 16'da oluşturulma süreci gösterilen dikkat mekanizmasının kurulmasıdır (Wang vd., 2023). Bu aşamada, düğüm özellikleri komşu düğümlerle birlikte değerlendirerek önem

derecelerini öğrenen ağırlık parametreleri (W) burada başlatılır. Her düğüm için tanımlanan dikkat puanları, ilgili düğümün çevresindeki diğer düğümlerle olan bağına dair bir önem seviyesini yansıtır (Vrahatis vd., 2024). Bu puanlar, modelde her özelliğin ne kadar etkili olacağını belirlediğinden genellikle normalleştirme süreçleriyle stabilize edilir (Wang vd., 2023).



Şekil 16. GAT dikkat mekanizması süreci

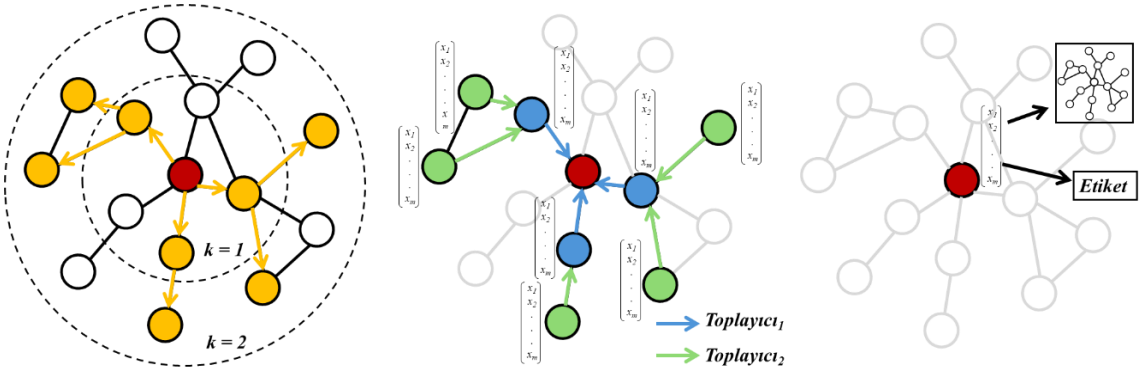
GAT mimarisinde sıklıkla çok başlı (multi-head) dikkat katmanları kullanılır. Bu sayede farklı konumlarda yer alan birçok temsil alt alanından elde edilen veri, çoklu perspektiflerden eşzamanlı olarak incelenerek zenginleştirilmiş temsil vektörleri elde edilir (Fu vd., 2020). Böylece, girdi verisinin çeşitli yönlerinin yakalanması sağlanmaktadır. Birden fazla dikkat mekanizmasının çıktıları birleştirildikten sonra, modelde bulunan doğrusal olmayan ilişkileri yakalamak için düzeltilmiş doğrusal birim (Rectified Linear Unit, ReLU) gibi fonksiyonlar kullanılan bir özellik dönüşüm aşaması gelmektedir (Sun vd., 2023). Bu adım, verideki doğrusal olmayan ilişkileri yakalamayı kolaylaştırarak modelin ifade gücünü artırır. Ayrıca, en kritik özelliklere odaklanmak amacıyla boyut azaltma işlemi uygulanmakta ve bu sayede modelin verimliliği artarken, hesaplama karışıklığı da azalmaktadır (Mostofi, 2024). Daha sonra, farklı dikkat başlıklarından gelen çıktılar, birleştirme aşamasında ağırlıklandırma ve toplama işlemlerinin uygulanmasıyla tek bir bütüncül temsil haline getirilir. Buradaki birleşim, her dikkat başlığından öğrenilen bilgileri dengeleyerek daha sağlam ve tutarlı bir özellik vektörü oluşturur. Ardından, model çıktı katmanı tasarımı aşamasıyla toplanan özellikleri daha fazla işleyerek nihai çıktı üretir

(Mostofi, 2024). Bu yapı, GAT'yi heterojen ve zaman içerisinde değişebilen grafik verileri üzerinde oldukça başarılı kılmakta; farklı görevler, veri türleri ve ölçeklerde işlevsel çözümler sunmaktadır (Mostofi vd., 2025).

1.9.6. GraphSAGE

GraphSAGE, geniş ölçekli yapılandırılmış grafik verilerini etkin bir şekilde işleme ve temsil öğrenimini verimli hale getirmek amacıyla geliştirilen bir GNN modeli olarak öne çıkmaktadır (Ay, 2023). Yerel komşuluk verisini örnekleyip bir araya getiren GraphSAGE, büyük miktarda düğüme sahip grafiklerde temsil öğrenimini mümkün kılan dayanıklı bir çerçeve sunmaktadır (Song vd., 2023). Uyarlanabilir bir örnekleme stratejisi ve grafiğin yapısal ve özellik detaylarını yakalayabilen bir toplama mekanizması kullanarak, geleneksel GCN'lerin kısıtlarının aşılmasını sağlamaktadır (Hamilton vd., 2017).

GraphSAGE'in temel prensibi, bir düğümün k-atlama (k-hop) komşuluk alanından bilgi örnekleyerek ve toplayarak düğüm temsilleri oluşturmaya dayanmaktadır (Miao vd., 2022). Burada k, hedef düğüm ile örneklenen düğümler arasındaki adım sayısını ifade etmektedir (Kaşıkçı, 2024). Sabit komşuluk yapılarına odaklanan GCN'lerin aksine, GraphSAGE farklı komşu boyutlarının ve yapılandırmalarının uyarlanabilir biçimde örneklenmesine olanak tanıyarak hem yerel hem de küresel ölçekteki grafik bilgilerinin etkin bir şekilde öğrenilmesini sağlamaktadır (Kumaş, 2023). Model hem hedef düğümün hem de rastgele seçilen komşuların özelliklerini alan ve bunlardan sabit uzunlukta bir vektör çıktısı üreten, eğitilebilir bir toplayıcı fonksiyonundan (trainable aggregator function) yararlanır (Hamilton vd., 2017). Bu toplayıcı fonksiyon; ortalama toplama, maksimum havuzlama ya da LSTM tabanlı toplama gibi çeşitli mekanizmalarla farklı grafik desenlerinin ve özelliklerinin yakalanmasına imkân tanımaktadır (Rahman & Azad, 2021). Şekil 17'de GraphSAGE algoritması gösterilmektedir.



Şekil 17. GraphSAGE algoritması

Şekil 17'de gösterildiği gibi GraphSAGE'in çalışma süreci, komşuluk örneklemesi, özellik bilgilerinin toplanması, graf tahmini ve etiketleme olmak üzere üç ana aşamaya ayrılabilir (Şahin, 2023).

- Komşuluk örneklemesi: GraphSAGE modelinin ilk adımı olarak, giriş grafiğinde her düğüm komşu düğümlerden örnekleme yapar ve böylece komşuluk kümesi oluşturulur. Bu süreç, grafikteki her düğüm için komşu düğümlerin bir alt kümesini seçmeyi amaçlamaktadır. Bu yerel komşuluk örneklemesi sayesinde GraphSAGE, etkili temsil öğrenimi açısından kritik öneme sahip olan yakın düğümlerden gelen bilgileri yakalayabilmektedir.
- Özellik bilgilerinin toplanması: Bu aşama, her arama derinliğinde kullanılacak toplama işlevlerinin öğrenilmesini merkez alır. Örneklenmiş komşu düğümlerin belirlenmesinin ardından, GraphSAGE bu düğümlerin özelliklerini birleştirmek için ilgili toplama yöntemlerini uygular. Böylece model, farklı komşu düğümlerden gelen bilgiyi bütünleştirerek her düğümün temsiline daha geniş bir bakış açısı kazandırır.
- Graf Tahmini ve Etiketleme: Bu aşamada, önceki adımlarda elde edilen sinir ağı temsilleri kullanılarak bir tahmin veya etiketleme görevi gerçekleştirilir. Bu görev, düğüm sınıflandırması ya da graf yapısının tanımlanması gibi farklı amaçları içerebilir. Modelin asıl öğrenme işlemi de bu aşamada yürütülmektedir. Bu üç aşama, denetimli bir yaklaşım çerçevesinde graf içinde yer alan her düğüm için sırayla tekrarlanır.

1.10. Tez Kapsamında Kullanılan Yazılımlar ve Teknolojik Araçlar

1.10.1. Autodesk Revit

Autodesk Revit, AEC sektöründe yaygın biçimde kullanılan, mimar, mühendis ve tasarımcılara entegre bir çalışma ortamı sağlayan BIM esaslı bir tasarım ve proje yönetimi yazılımıdır (Ozcan Deniz, 2018). Programın Autodesk firması tarafından 2002 yılında satın alınması, yazılımın daha geniş kullanıcı kitlesine ulaşmasına ve BIM alanında hızla gelişip yaygınlaşmasına katkı sağlamıştır (Yeşilyurt, 2017; Alshamali, 2020; Canpolat, 2023). Revit adı, "revise instantly" (anında değişiklik/ güncelleme) ifadesinden türetilmiştir ve temel olarak parametrik modelleme yaklaşımıyla gerçekleşen anlık değişiklikler proje genelinde otomatik şekilde güncellenebilmektedir (Topal, 2019). Bu özelliği sayesinde, bir yapı elemanında yapılan değişiklik, kesitlerden metraj listelerine kadar tüm görünümlere eşzamanlı yansıtılabilmektedir (Altınok, 2020).

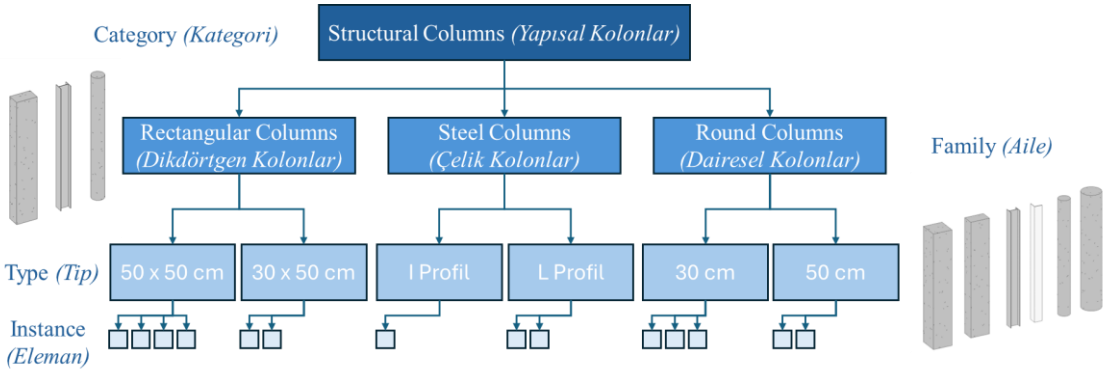
Revit'in en önemli avantajlarından biri, farklı disiplinlere (mimarlık, inşaat, elektrik, mekanik vb.) ait iş akışlarını tek bir çatı altında toplayarak koordinasyon sorunlarını azaltmasıdır (Sampaio vd., 2023). Özellikle çoklu kullanıcı desteği sayesinde aynı proje üzerinde birden fazla uzmanın eş zamanlı çalışabilmesi, takım içi iletişim ve iş birliğini geliştirmektedir (Alshamali, 2020). Bunun yanı sıra Revit, 2B projelerden farklı olarak 3B tasarım ve modelleme imkânı sunmaktadır. Böylece plan, kesit, görünüş ve metraj listeleri gibi veriler, tek bir modelden otomatik şekilde türetilebilir. Bu yaklaşım, tasarımcıyı tekrarlı çizim ve hesaplamalardan büyük ölçüde kurtarırken geri dönüşlerin fazla olduğu projelerde zaman kazandırır (Topal, 2019).

Revit, tasarımın yanı sıra proje yönetimi, metraj ve koordinasyon konularında da önemli imkânlar sunmaktadır. Metraj listelerini hazırlama, görselleştirme ve veri tabanında proje bileşenlerini saklama özellikleri, projenin bütüncül olarak ele alınmasına yardımcı olmaktadır (Canpolat, 2023). Böylece hem mimarlık hem de mühendislik disiplinlerine ait projeler tek bir model üzerinden yönetilebilir ve hata payı en aza indirilebilir. Ayrıca program, 4B ve 5B modellemeye uyumlu çalışabilmekte; ancak bazı durumlarda ek yazılımlara da ihtiyaç duyulmaktadır (Gülerses, 2018). Autodesk Navisworks gibi tamamlayıcı programlarla sorunsuz çalışabilmesi, çakışma tespiti ve yapım süreçlerinin daha sağlıklı bir biçimde planlanmasına olanak tanımaktadır (Canpolat, 2023). Kısacası,

tek başına da kullanılabilen Revit, BIM ekosistemi içinde veri paylaşımı, koordinasyon ve doğru metraj çıktıları elde etme konularında kilit bir rol üstlenmektedir.

Modelleme sürecinde, farklı formatlarda kaydedilmiş projeler (.dwg, .dxf, .ifc vb.) Revit'e aktarılabildiği gibi, Revit modelleri de çeşitli uzantılarda dış ortama aktarılabilmektedir (Raiman, 2022). Böylelikle tasarımcılar, farklı yazılımlarla veri alışverişini kesintisiz biçimde sürdürebilirler. Geniş eğitim materyallerinin yanı sıra çevrimiçi kütüphaneler üzerinden çok sayıda hazır nesneye ulaşabilmek, Revit'in dünya çapında tercih edilmesindeki faktörlerden biri olmaktadır (Alshamali, 2020). Bu kütüphane desteği ve parametrik modelleme yapısı sayesinde proje bileşenleri gerçeğe daha yakın görsellerle sunulabilir ve tasarım süreci hızlandırılabilir (Yeşilyurt, 2017).

Revit yazılımında, yapı elemanlarının sınıflandırılması ve yönetiminde Category (Kategori), Family (Aile), Type (Tip) ve Instance (Eleman / Somut örnek) hiyerarşisi büyük önem taşımaktadır (Aubin, 2020). En üst düzeyde yer alan kategori; duvar, pencere, kapı, kolon, kiriş ya da döşeme gibi temel yapı bileşenlerini sınıflandıran ve Revit tarafından önceden tanımlanmış bir çerçeveyi ifade etmektedir. Bu kategoriler, kullanıcılar tarafından eklenip silinememekte, dolayısıyla sistemin tutarlı ve standart bir yapı sunmasına katkı sağlamaktadır. Kategorilerin altında tanımlanan aile (family), benzer fiziksel ve işlevsel özellikleri taşıyan öğelerden oluşmaktadır. Örneğin, "Kapılar (Doors)" kategorisi altında farklı kapı özelliklerini barındıran bir aile bulunabilir. Bu aile içinde, boyut ve malzeme gibi parametreler bakımından farklılık gösteren birden fazla tip (type) tanımlanabilmektedir. Dolayısıyla aynı ailenin iki farklı tipi, boyut ve malzeme gibi değişkenler açısından birbirinden ayrılmakta, ancak temel işlevsellik bakımından benzerlik taşımaktadır. Bir tipin projeye yerleştirilen somut örnekleri ise "instance" olarak adlandırılmaktadır. Örneğin, aynı kapı tipinin bir projede farklı konumlarda ve yönlerde birden fazla kez kullanılması, her bir yerleşimin ayrı bir "instance" olarak değerlendirilmesine imkân vermektedir. Şekil 18'de gösterilen bu hiyerarşik yapı, Revit modelinde yer alan tüm bileşenlerin düzenli ve verimli biçimde yönetilmesini kolaylaştırmakta ve bu sayede tasarım, dokümantasyon ve koordinasyon süreçlerinde bütüncül bir kontrol sağlamaktadır.



Şekil 18. Revit eleman hiyerarşisi

Revit projelerinde yapı elemanlarıyla ilgili verileri saklamak ve göstermek amacıyla çeşitli parametre tipleri kullanılmaktadır. Bu parametreler, yazılımın verileri organize etmesi ve farklı disiplinlerce paylaşılan bilgilere erişimi kolaylaştırması bakımından kritik öneme sahiptir (URL-4, 2024). Başlıca Revit parametre tipleri; Sistem Parametreleri (System Parameters), Proje Parametreleri (Project Parameters), Aile Parametreleri (Family Parameters), Paylaşılan Parametreler (Shared Parameters) ve Küresel Parametreler (Global Parameters) şeklinde sıralanabilir (Politi, 2018). Şekil 19'da parametre tipleri ve kullanım açısından en önemli üç özellik gösterilmektedir.

Parametre Tipi	Kullanım Yeri	Etiketleme	Listeleme / Çizelgeleme
Sistem (System)	*Aile*, *Proje*	✓	✓
Proje (Project)	*Proje*	✗	✓
Aile (Family)	*Aile*	✗	✗
Paylaşılan (Shared)	*Aile*, *Proje*	✓	✓
Küresel (Global)	*Proje*	✗	✗

Şekil 19. Revit parametre tipleri

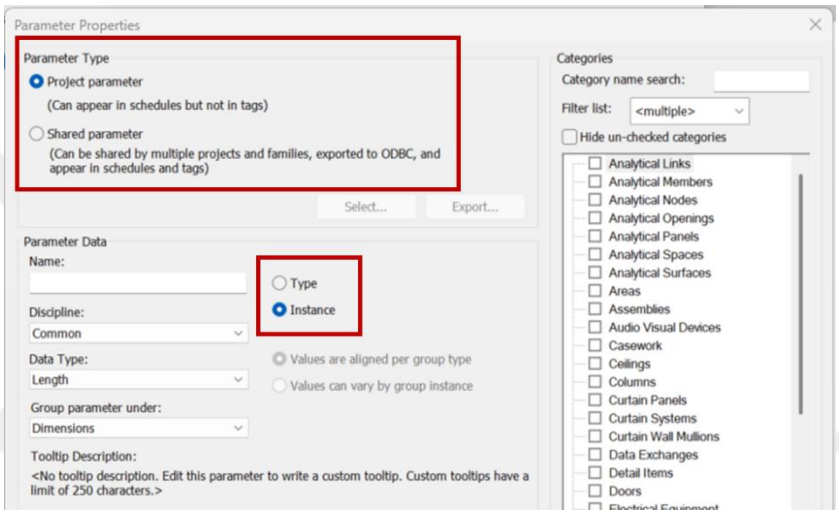
1. Sistem Parametreleri (System Parameters): Revit'in dahili parametreleri olarak tanımlanan sistem parametreleri; kullanıcılar tarafından oluşturulamayan, değiştirilemeyen ve silinemeyen parametrelerdir (URL-4, 2024). Ancak bu

parametrelerin değerleri değişebilmekte ve diğer parametre türleri gibi etiketlerde, listelerde ve paftalarda kullanılabilmektedir. Örneğin, bir eleman seçildiğinde "Properties" penceresi altında görülen eleman kimlik (identity data) parametreleri, faz/aşama (phasing) parametreleri, m, cm gibi proje birimleri (project units), projenin kuzey yönü (project north) gibi parametrelerdir.

2. Proje Parametreleri (Project Parameters): Proje parametreleri, kullanıcı tarafından tanımlanır ve proje ya da proje elemanları hakkında ilave verileri saklamak, paylaşmak ve görüntülemek için kullanılırlar (URL-4, 2024). Standart Revit parametrelerinde yer almayan veya özel ihtiyaçlar doğrultusunda eklenmek istenen bilgiler bu kategoride tanımlanabilir. Proje parametreleri, Revit'in paylaşılan parametre (shared parameter) özelliği yardımıyla birden fazla projede de kullanılabilirler. Ayrıca, listelerde ve paftalarda görünmektedirler. Böylece aynı bilgilerin farklı ekipler ve projeler arasında tutarlı bir şekilde paylaşılması sağlanır.
3. Aile Parametreleri (Family Parameters): Aile parametreleri, belirli bir aile (family) tipine özgü olan, kullanıcı tanımlı parametrelerdir. Bu parametreler sayesinde aile içerisindeki çeşitli alt bileşenlerin davranışı ve görünümü daha hassas biçimde kontrol edilebilir (URL-4, 2024).
4. Paylaşılan Parametreler (Shared Parameters): Paylaşılan parametreler, bir veya birden fazla proje ve aile arasında ortak kullanımı mümkün kılan özel parametrelerdir (URL-4, 2024). Tanımlanan paylaşılan parametreler, ayrı bir metin dosyasında saklanır ve başka projelere de kolaylıkla entegre edilebilir. Proje parametrelerinin aksine etiketlerde de kullanılabilirler. Paylaşılan parametreler, kurumsal standartların projeler arasında tutarlı uygulanması ve verilerin merkezi olarak yönetilmesi açısından büyük kolaylık sağlamaktadır.
5. Küresel Parametreler (Global Parameters): Küresel parametreler, proje genelinde ortak kullanılacak boyut, açı, mesafe gibi değerleri tek bir noktadan yönetmeyi sağlarlar (URL-4, 2024). Bu parametreler, farklı model bileşenleri arasında ilişki kurar ve bir bileşende yapılan değişikliğin ilgili diğer bileşenlere otomatik olarak yayılmasına olanak tanır.

Bu parametre türleri, Revit içerisinde tam kapsamlı bir BIM yaklaşımını mümkün kılarak, verilerin doğru saklanması, paylaşılması ve yönetilmesini destekler. Böylece hem tasarım hem de uygulama süreçlerinde daha etkin koordinasyon ve esneklik sağlanır.

Bu parametreler modeldeki elemanlara uygulanması açısından “Instance Parameter” ve “Type Parameter” olmak üzere iki türde oluşturulmaktadır. Parametre tanımlanırken söz konusu parametre “instance parameter” olarak belirlenirse, parametre değerinde yapılacak bir değişiklik yalnızca ilgili elemanın değerini etkilemektedir. Buna karşın, aynı parametrenin “type parameter” olarak tanımlanması durumunda, o tipe (type) ait tüm elemanlarda söz konusu değişiklik otomatik olarak uygulanmaktadır. Şekil 20’de Revit’te proje/paylaşılan parametre (project/shared parameters) oluşturulma aşamasında “type” ve “instance” parametre tipleri seçimi gösterilmektedir.



Şekil 20. Type ve instance parametre seçimi

Yukarıda verilen bilgiler ışığında tez kapsamında; 3B modellerin oluşturulması, modeldeki elemanlara parametreler tanımlanması, bu parametrelere yeni bilgiler yüklenmesi, diğer kullanılan yazılımlarla ilişkiler kurulması ve iş programı elde edilebilmesi için Revit yazılımı BIM aracı olarak kullanılmıştır. Tüm bu işlemler için Revit 2024 versiyonu kullanılmıştır.

1.10.2. Autodesk Navisworks Manage

Autodesk Navisworks; ilk olarak İngiltere merkezli Sheffield firması tarafından geliştirilen ve 2007 yılında Autodesk tarafından satın alınan, ağırlıklı olarak analiz ve koordinasyon odaklı bir yazılımdır (Yeşilyurt, 2017). Tasarım süreci yerine, farklı disiplinlerden elde edilen 3B modelleri entegre etme, proje planlamasını destekleme,

çakışma tespiti yapma ve 4B/5B modelleme gibi işlevlerle ön plana çıkmaktadır (Özcan, 2021). Freedom, Simulate ve Manage olmak üzere üç ayrı modüle sahip olan yazılım, kullanıcılarına farklı seviyelerde özellikler ve kontrol imkânları sunmaktadır (URL-5, 2024).

Navisworks, çok çeşitli 3B veri formatını (dwg, IFC, 3ds, SketchUp vb.) destekleyerek farklı modelleme yazılımlarında oluşturulmuş verilerin tek bir ortamda birleştirilmesini kolaylaştırmaktadır. Bu sayede, yapısal, mimari veya mekanik gibi farklı disiplinlere ait modeller aynı sahnede bir araya getirilmekte ve çakışmalar hızlı bir şekilde görüntülenmektedir (Topal, 2019). Ayrıca Primavera, MS Project ve Excel gibi planlama dosyalarına bağlanabilme özelliği sayesinde; 4B veya 5B modellemeler yoluyla zaman, maliyet ve kaynak yönetimi süreçleri de yazılım içerisine entegre edilebilmektedir (Canpolat, 2023).

Program, tasarım hatalarını erken aşamada belirleyerek projedeki farklı taraflar arasında daha etkili bir iş birliği sağlamayı hedeflemektedir (Huang & Bhat, 2017). Özellikle Manage modülü; gelişmiş çakışma kontrolü, koordinasyon, yapım simülasyonu, proje analizleri ve birleştirilmiş model inceleme gibi ileri seviye işlevleriyle dikkat çekmektedir (Özcan, 2021). Bunun yanı sıra gerçeğe yakın animasyon, gerçek zamanlı gezinme ve fotoğraf gerçekçiliği düzeyinde görselleştirme olanaklarıyla proje sunumlarını daha anlaşılır ve etkileyici hâle getirmektedir (Alshamali, 2020).

Bu bilgiler ışığında, birçok yazılım ile entegre olarak çalışabilmesi ve 4B modelleme aşamalarında yaygın kullanımı nedeniyle tez çalışması kapsamında Autodesk Navisworks Manage yazılımı ve 2024 versiyonu kullanılmıştır.

1.10.3. Görsel Programlama ve Dynamo

Görsel programlama, kullanıcıların kod yazma deneyimini kolaylaştırmak amacıyla düğüm, blok veya diyagram gibi grafiksel öğelerle mantıksal akış oluşturmalarını sağlayan bir yöntem olarak tanımlanmaktadır (Yavan, 2023). Bu yaklaşım, geleneksel metin tabanlı programlamaya kıyasla daha az sözdizimi bilgisi gerektirmesi nedeniyle özellikle yeni başlayanlar için daha uygun bir seçenek haline gelmektedir. Metin tabanlı programlama ise esnekliği ve karmaşık projelerdeki kontrol kolaylığıyla öne çıkmakta, dolayısıyla her iki yöntemin de proje kapsamına göre kendine özgü avantajları bulunmaktadır (Noone & Mooney, 2018).

Autodesk Revit ve Maya gibi çeşitli yazılımlarla bütünleşebilen ya da tek başına "Sandbox" modunda da çalışabilen Dynamo, bu görsel programlama yaklaşımını ileri boyuta taşıyarak parametrik modelleme, veri analizi ve otomasyona yönelik kapsamlı bir platform sunmaktadır. C# temelli bir altyapıya sahip olan Dynamo, sürükle-bırak yöntemiyle düğümlerin (node) görsel düzenlemesine imkân tanırken, Python düğümü (Python script) eklenmesini de destekleyerek kullanıcıların karmaşık fonksiyonlar veya özel algoritmalar geliştirmesine olanak tanımaktadır (Altınok, 2020).

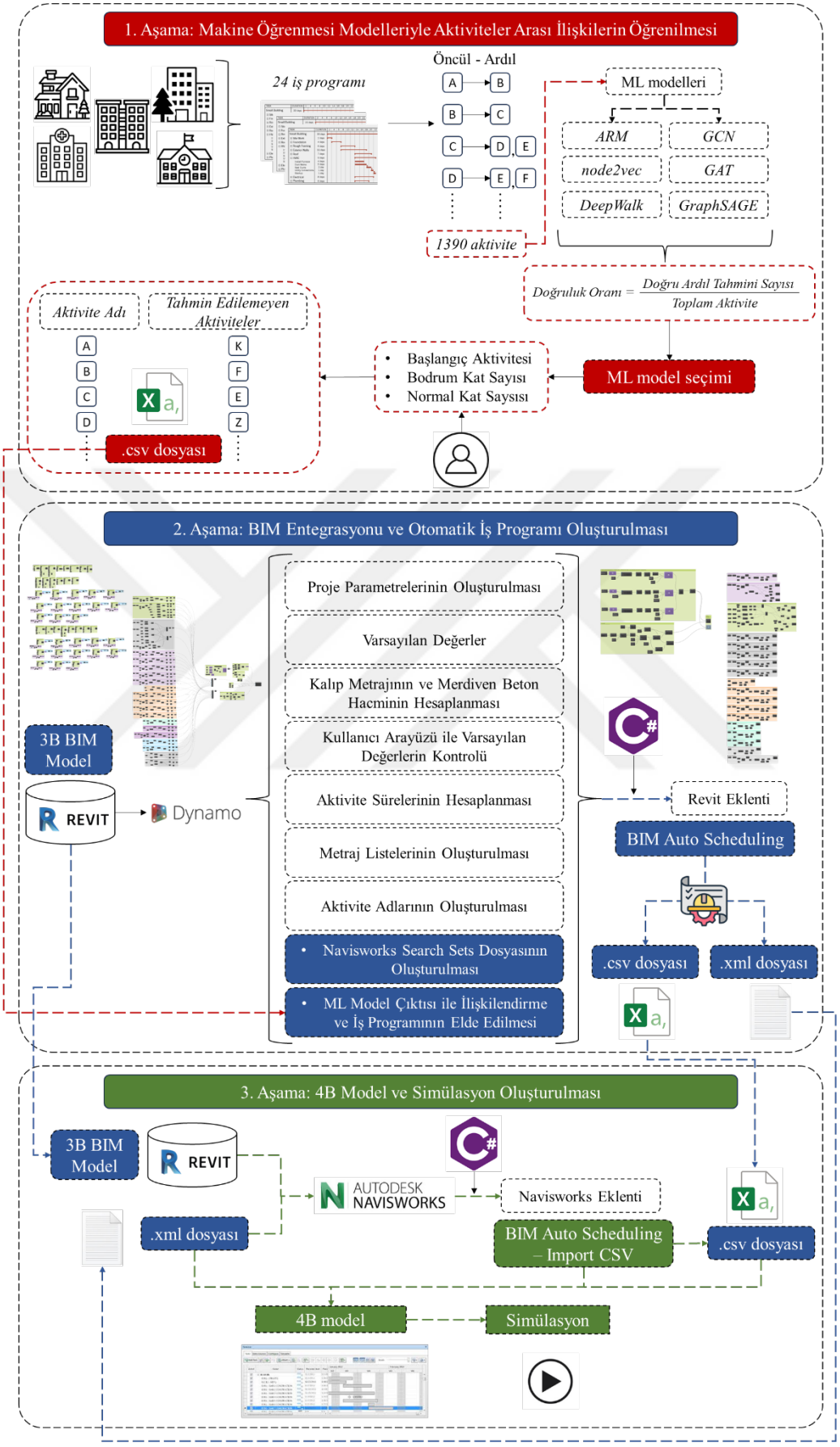
AEC sektöründe, çeşitli araştırmalar Dynamo'nun BIM süreçlerinde dokümantasyon, koordinasyon ve analiz gibi iş akışlarını hızlandırdığını ortaya koymaktadır (URL-6, 2024). Aynı zamanda, tekrar eden görevlerin otomasyonu, cephe tasarımından enerji analizine kadar geniş bir yelpazede pratik uygulamalara zemin hazırladığı da vurgulanmaktadır (Çepni, 2021). Böylelikle hem akademik hem de endüstriyel alanda kullanıcılara esnek, ölçeklenebilir ve kullanıcı dostu bir çözüm sunan Dynamo, BIM tabanlı projelerde veri yönetimini ve tasarım süreçlerini önemli ölçüde kolaylaştırmaktadır (URL-6, 2024).

Dynamo, parametrik tasarım ve BIM ekosistemindeki otomasyon ihtiyacını karşılamak amacıyla sürekli gelişen bir araç seti sunmaktadır. Bu bütüncül yaklaşım, kullanıcıların ister görsel ister metinsel kodlama tekniklerini tercih ederek karmaşık projelerde verimli ve hatasız süreçler tasarlamasına imkân tanımaktadır. Bu amaçla tez kapsamında Revit'e eklenti oluşturulması ve bu eklenti içerisinde tezin sonraki aşamalarında bahsedilen kodlamaların yapılması amacıyla Dynamo programı ve 2.19 sürümü kullanılmıştır.

2. YAPILAN ÇALIŞMALAR

2.1. Araştırma Yöntemi

Bu doktora tez çalışmasında, yapım projelerinde BIM ile ML yöntemlerinin bütünleşik ve otomasyon odaklı bir çerçevede kullanılarak iş programı hazırlanması ve 4B simülasyonun elde edilmesi amaçlanmaktadır. Bu doğrultuda çalışmada geliştirilen çerçeve, geleneksel iş programı oluşturma sürecinde gözlenen zaman kaybı, manuel müdahale zorunluluğu ve insan hatası risklerini azaltmayı hedefleyen bir yaklaşım sunmaktadır. Araştırma yöntemi, 1.4. Tezin Amacı ve Kapsamı başlıklı bölümde verilen tez çalışması kapsamında belirlenen gereksinim ve hedefler doğrultusunda düzenlenmiş olup 1.6. Araştırma Yöntem Özeti başlıklı bölümde kısaca tanıtılmıştır. Şekil 21, bu kısa tanıtımı detaylandırarak tez çalışmasında geliştirilen araştırma yöntemini göstermektedir. Şekil 21'den de görülebileceği üzere araştırma yöntemi temelde ML modelleri ile aktiviteler arası ilişkilerin öğrenilmesi, BIM entegrasyonu ve otomatik iş programı oluşturulması, 4B model ve simülasyon oluşturulması adımları olmak üzere üç ana adımı içermektedir. Sonraki bölümlerde, çalışmada sunulan araştırma yönteminin bu aşamaları ve bu aşamaların birbirleriyle olan etkileşimi sunulmaktadır.



Şekil 21. Araştırma yöntemi akış diyagramı

1. Aşama: Makine Öğrenmesi Modelleriyle Aktiviteler Arası İlişkilerin Öğrenilmesi: Bu aşamada, farklı ölçek ve türdeki (villa, konut, kamu binası, hastane, okul vb.) toplam 24 inşaat projesinden derlenen iş programları kullanılarak, aktiviteler arası ilişkilerin ML teknikleriyle ortaya konması amaçlanmıştır. Söz konusu iş programları, aktivitelerin öncül-ardıl ilişkileri ve Bitiş-Başlangıç (Finish-to-Start, FS) mantığı korunacak şekilde düzenlenmiş ve ML süreçlerine uygun tekdüze bir isimlendirme yöntemi uygulanmıştır. Örneğin "Level -1-Grobeton İmalatı, Level -1-Temel Kalıp İmalatı" biçiminde etiketlenen aktiviteler sayesinde benzer adlandırmalara sahip faaliyetler aynı kategoride işlenebilmiştir. Böylece 24 projeden toplam 1390 aktivite toplanmış olup, her bir aktiviteye ilişkin kat (seviye/level), aktivite adı, ardıl aktivite ve proje numarası bilgileri, ML modellerine girdi olarak sunulmuştur.

Araştırmada, ARM, node2vec, DeepWalk, GCN, GAT ve GraphSAGE olmak üzere altı farklı ML modeli kullanılmıştır. Her model performansını etkileyen hiper parametreleri için belirli sayıda parametre değerleri kombinasyonu denenmiş ve oluşturulan eğitim-test seti üzerinde en iyi sonuç veren parametre değerleri saptanmıştır. Model değerlendirmesinde, proje ölçeği ve aktivite sayısı bakımından çeşitlilik sağlamak amacıyla, en düşük, orta ve en yüksek aktivite yoğunluğuna sahip projelerden rastgele seçilmiş ikişer projeden oluşan farklı eğitim-test setleri oluşturulmuştur. Modellerin performansları, "test setinde doğru ardıl tahmini yapılan aktivitelerin, toplam test aktivitelerine oranı" olarak tanımlanan Doğruluk Oranı metriğiyle karşılaştırılmıştır.

Bu değerlendirmeler neticesinde, en yüksek doğruluğa sahip ML modeli belirlenmiş ve test projelerindeki aktiviteler ile ardıl bilgileri, .csv uzantılı bir dosyada dışarıya aktarılmıştır. Aktarım sırasında, Python ile gerçekleştirilen kodlama sayesinde, BIM modelinin kat sayısı ve seviyesine ilişkin "Başlangıç Aktivitesi", "Bodrum Kat Sayısı" ve "Normal Kat Sayısı" bilgileri dikkate alınarak aktivitelerin sıralaması düzenlenmiştir. Ayrıca, ML modellerinin ardılını tahmin edemediği aktivitelerin de gözden kaçmaması ve daha kapsamlı bir iş programı oluşturulması amacıyla "Tahmin Edilemeyen Aktiviteler" başlığı altında aynı .csv dosyasında sunulmuştur. Kullanıcı, ihtiyaç duyduğu takdirde bu dosya üzerinde aktivite sıralamasına yönelik değişiklikleri yapabilmekte; ardından nihai .csv dosyası, Revit yazılımına eklenen bir arayüz yardımıyla BIM modelindeki elemanlarla eşleştirilebilmektedir. Bu sayede, projeye özel parametrelerin dikkate alındığı ve ML tabanlı tahminlerle desteklenen bütünleşik bir iş programı taslağı elde etmek mümkün hâle gelmektedir.

2. Aşama: BIM Entegrasyonu ve Otomatik İş Programı Oluşturulması: Bu aşamada, Revit ortamına eklenen özel bir eklenti sayesinde, 1. aşamadan elde edilen .csv uzantılı dosyanın doğrudan otomatik iş programına dönüştürülmesi amaçlanmıştır. Bu kapsamda öncelikle Dynamo üzerinden kodlamalar geliştirilmiştir. Söz konusu kodlar, yapı elemanlarına yeni parametre eklenmesi, adam-saat değerlerinin girilmesi, kalıp metrajının çıkarılması, kullanıcı arayüzü aracılığıyla varsayılan değerlerin görüntülenmesi ve değiştirilebilmesi, aktivite sürelerinin hesaplanması, metraj listelerinin oluşturulması, aktivite isimlerinin düzenlenmesi, Navisworks Search Sets oluşturulması ve ML modelinden elde edilen aktivite sıralaması ile 3B BIM modelindeki aktivitelerin eşleştirilmesi gibi temel işlemleri yerine getirmektedir. Bu amaçla toplam dokuz adet Dynamo kodu oluşturulmuştur. Daha sonra kullanıcı açısından bu kodların tek tek çalıştırılmaması ve işlemlerin doğru sıralamada yürütülmesi amacıyla Visual Studio ortamında C# programla dili ile Revit içerisine "BIM Auto Scheduling" adıyla bir eklenti ve söz konusu eklentiye ilişkin yönergelerin yer aldığı "About" düğmesi oluşturulmuştur.

Eklentinin çalıştırılmasıyla birlikte, oluşturulan dokuz Dynamo kodu sırasıyla otomatik olarak çalışmaktadır. Bu süreçte kullanıcıdan iki kez girdi talep eden arayüz pencereleri açılmaktadır. İlk pencere, yukarıda belirtilen varsayılan değerlerin onaylanması veya gerektiğinde güncellenmesi için tasarlanmıştır. İkinci arayüz ise iş programı başlangıç tarihi ile 1. aşamadan gelen .csv dosyasının seçilmesini sağlamaktadır. Kodların çalışması tamamlandığında, yazılım nihai .csv uzantılı dosyada iş programını ve Navisworks için gerekli olan .xml uzantılı "Search Set" dosyasını otomatik olarak dışarıya aktarmaktadır. Bu sayede, proje planlamasına ilişkin en güncel veriler hem iş programı hem de 3B model görünümüyle kolayca ilişkilendirilebilecek şekilde ilgili dosyalarda sunulmuş olmaktadır.

3. Aşama: 4B model ve simülasyon oluşturulması: Bu aşamada, Navisworks ortamına iş programının aktarılması ve modellenen yapı elemanlarıyla eşleştirilmesi süreçlerinin otomatikleştirilmesi hedeflenmiştir. Model ve aktivite sayıları arttıkça, Navisworks'te aktivitelerin elemanlarla manuel olarak ilişkilendirilmesi zaman alan ve hata riski yüksek bir işlem hâline gelmektedir. Bu gereksinimden yola çıkılarak, Visual Studio ortamında C# programlama diliyle "BIM Auto Scheduling - Import CSV" adlı bir eklenti geliştirilmiştir.

Geliştirilen eklenti sayesinde, 2. aşamada üretilen .csv uzantılı nihai iş programı, herhangi bir ön düzenlemeye gerek duyulmaksızın Navisworks yazılımına aktarılabilmekte ve aynı zamanda .xml dosyası aracılığıyla içe alınan "Search Sets" veri kümesiyle aktivite-

eleman eşleştirmesi otomatik olarak gerçekleştirilmektedir. Bu işlem sonucunda, 4B model oluşturmak için gerekli tüm adımlar hızla tamamlanabilmekte ve "Simulate" komutuyla inşaat sürecine ilişkin görsel bir simülasyon elde edilebilmektedir. Böylelikle hem zaman hem de iş gücü tasarrufu sağlanmakta, aynı zamanda aktivite ve model verilerinin yüksek doğrulukla senkronize edilmesi mümkün kılınmaktadır.

2.2. Makine Öğrenmesi Modelleriyle Aktiviteler Arası İlişkilerin Öğrenilmesi

Tez çalışmasının bu bölümünde, 24 farklı inşaat projesinden elde edilen ve FS ilişkisi korunarak standart bir adlandırma yaklaşımıyla düzenlenen 1390 aktivite verisi üzerinde uygulanan ML modellerine odaklanılmaktadır.

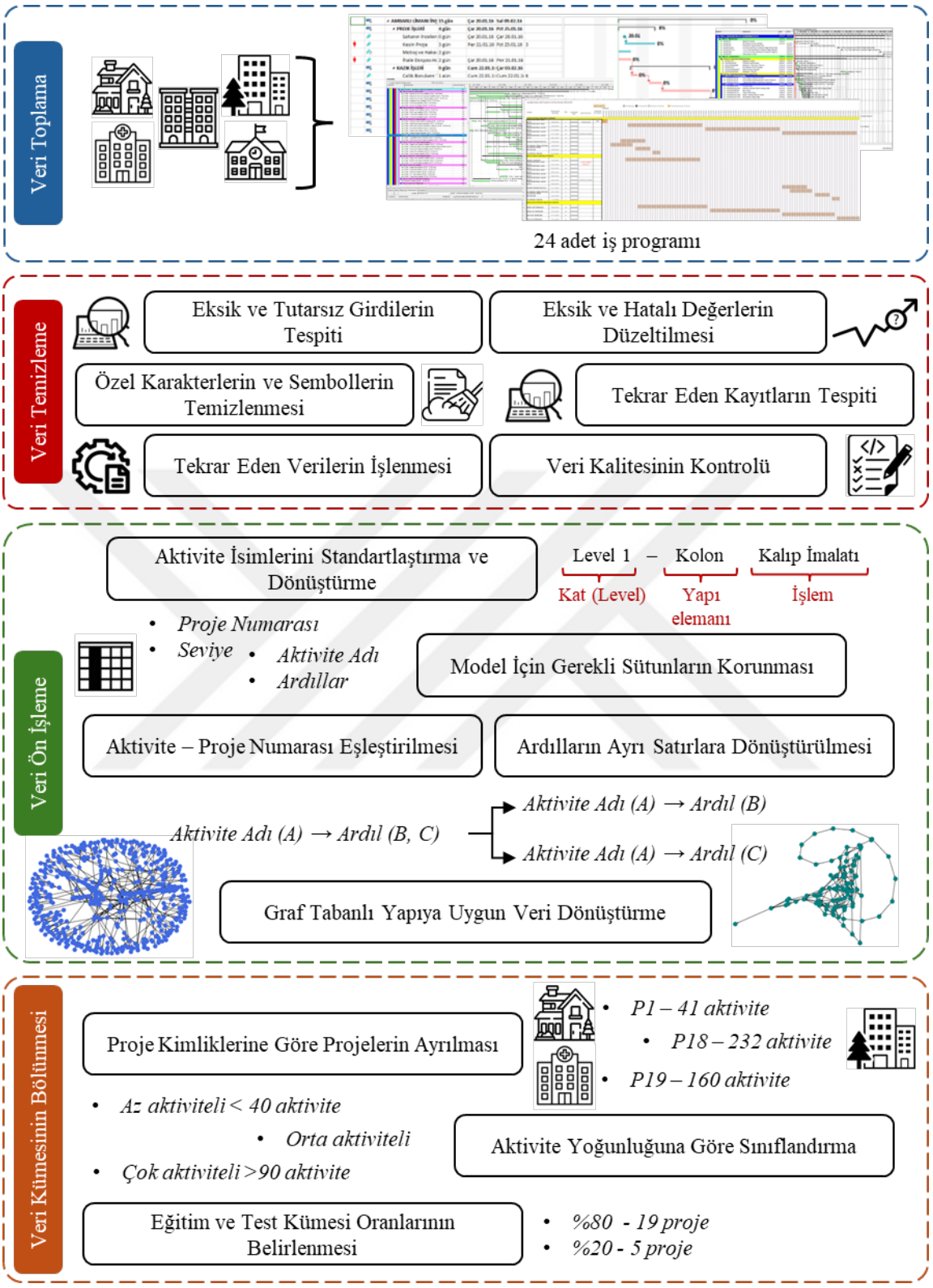
Önerilen yaklaşım, küçük ve orta ölçekli projelerdeki aktivite sıralaması problemine yenilikçi çözümler sunmayı hedeflemektedir. Geleneksel derin sinir ağı yaklaşımları (Deep Neural Network, DNN) (örneğin, RNN, LSTM vb.) büyük veri setlerine ihtiyaç duyduklarından, az sayıda aktivite içeren projelerdeki ilişkisel ve sıralı bağıntıları yakalamakta yetersiz kalabilmektedir (Dou vd., 2023). Bu nedenle çalışma kapsamında, GNN modelleri kullanılarak aktivite ağlarındaki ilişkisel yapı yakalanmaya çalışılmıştır. GNN modelleri, özellikle GCN, GAT ve GraphSAGE, faaliyet ağının yapısını katman bazlı yayılım ve komşu toplama mekanizmaları ile doğrudan kullanmaktadır (Chen vd., 2021). Buna karşılık, node2vec ve DeepWalk gibi ML modelleri, önyargılı rastgele yürüyüşler ve "skip-gram" optimizasyonu aracılığıyla gizli temsiller öğrenerek farklı bir yaklaşım benimsemektedir (Song vd., 2023). Aktiviteler arasındaki gizli benzerlikleri yakalama kapasitesinden dolayı node2vec ve DeepWalk modelleri, küçük veri kümeleri içeren durumlarda GNN modellerine kıyasla cazip bir alternatif haline gelebilmektedir (Rahman & Azad, 2021). Ayrıca, düşük veri yoğunluğuna rağmen önemli ilişkilerinin yakalanmasında güçlü bir alternatif sunan ARM yöntemi de bu tür çalışmalarda değerlendirilmiştir (Mostofi vd., 2023a; Lee vd., 2019). Dolayısıyla, çalışma kapsamında GCN, GAT, GraphSAGE, node2vec, DeepWalk ve ARM olmak üzere altı farklı ML modeli yaklaşımı benimsenmektedir.

Farklı hiper parametre değerleriyle eğitilen ve test edilen ML modelleri arasından en yüksek doğruluğa sahip model seçilmiş, ardından ilgili aktiviteler ve ardıl ilişkileri .csv formatında dışa aktarılmıştır. Kullanıcının belirttiği başlangıç aktivitesi ve kat sayısına

göre düzenlenen bu sıralama, “Tahmin Edilemeyen Aktiviteler” bilgisiyle birlikte BIM yazılımıyla entegre bir planlama taslağı oluşturmak için kullanılmıştır.

2.2.1. Veri Toplama ve Hazırlama

Bu bölümde, ham inşaat proje kayıtlarının ileri düzey analitik yöntemlerle işlenebilmesi için gerekli veri toplama, dönüştürme ve hazırlama süreci ele alınmaktadır. Şekil 22’de, 24 farklı inşaat projesinden (villa, konut, iş yeri, hastane, okul vb.) elde edilen iş programlarının toplanmasından başlayarak, verilerin temizlenmesine, normalizasyonuna, yeniden yapılandırılmasına, veri kümesinin eğitim ve test alt kümelerine ayrılmasına kadar izlenen sistematik yaklaşım özetlenmektedir.



Şekil 22. Veri toplama ve hazırlama aşamaları

1. Veri Toplama: Türkiye'deki 24 projeye ait Primavera P6, MS Project, PDF ve Microsoft Excel gibi farklı formatlarda düzenlenmiş iş programları toplanmıştır. Bu

programlarda yer alan aktivite isimleri, öncül–ardıl ilişkileri, WBS kodları, adam-saat değerleri ve aktivite süreleri gibi bilgiler başlangıçta yapılandırılmamış veya yarı yapılandırılmış biçimdedir. Projelerin genel özelliklerini ve iş programı kapsamlarını gösteren özet bilgiler Tablo 1'de sunulmaktadır.

Tablo 1. İş programlarının elde edildiği projelere ait genel bilgiler

Proje Numarası	Bina Türü	Kat Sayısı	Bina Toplam Kat Alanı (m^2)	İş Programı Dosya Türü
P1	Villa-1	Bodrum + Zemin	160	MS Project
P2	Villa-2		155	
P3	Villa-3		140	
P4	Villa-4	Bodrum + Zemin + 1 Kat	180	PDF
P5	Villa-5		160	
P6	Villa-6		120	
P7	Villa-7		115	
P8	Villa-8		145	
P9	Villa-9		140	
P10	Villa-10	Zemin + 1 Kat	160	PDF
P11	Villa-11		155	
P12	Villa-12		150	
P13	Villa-13		110	
P14	Villa-14		120	
P15	Villa-15	Zemin	150	MS Project
P16	Villa-16		130	
P17	Villa-17		110	
P18	Konut ve İş Yeri	Zemin + 13 Kat	-	Excel & PDF
P19	Hastane - A Blok	3 Bodrum + Zemin + 5 Kat	-	Primavera P6
P20	Kamu Binası (Yalnızca Betonarme)	2 Bodrum + Zemin + 7 Kat	-	Excel & PDF
P21	Hastane	4 Bodrum + Zemin + 8 Kat	-	Excel
P22	Konut	Zemin + 4 Kat	-	PDF
P23	Okul (Yalnızca Betonarme)	Zemin + 5 Kat	-	Excel
P24	Okul (Yalnızca Betonarme)	Zemin + 3 Kat	-	Excel

Veri toplama aşaması, çeşitli proje ölçekleri ve türlerine (villa, konut, kamu binası, hastane, okul vb.) ait bilgilerin derlenmesini sağlayarak, ML modellerinin geniş bir yelpazede genellenebilir olmasına katkı sunmuştur. Ancak format farklılıkları, verilerde tutarsızlıklar ve eksik alanlar bulunması nedeniyle sonraki aşamalarda kapsamlı bir

dönüşüm ve temizleme süreci gerekmiştir. Şekil 23'te, toplanan iş programlarından 624 aktiviteli bir konut binasının Excel'de hazırlanan iş programının kısa bir bölümü gösterilmektedir.

BÜTÇE KODU	Disiplin	Aktivite ID	İMALAT AÇIKLAMASI	ETAP	BLOK	BİRİM	İnşaat Alanı	Adam.Saat	İmalat Süresi Tek Ekibe Göre	Ekip Sayısı	İmalat Süresi	İmalat Başlangıç Tarihi	İmalat Bitiş Tarihi
BVP-B94-D			BEYKOZ VİLLA PROJESİ İNŞAAT İŞLERİ								986,00	01.06.2021	12.02.2024
BVP-B94-D-TPI			TOPRAK İŞLERİ									01.06.2021	17.12.2022
BVP-B94-D-TPI-KAI			KAZI - DOLGU İŞLERİ					153.930				01.06.2021	04.12.2022
BVP-B94-D-TPI-IKS			İKSA İŞLERİ					49.893				11.06.2021	14.12.2022
BVP-B94-D-TPI-YSY			YOL VE SANAT YAPILARI					71.387				11.06.2021	17.12.2022
BVP-B94-D-ALT			ALTYAPI İŞLERİ					210.569				11.06.2021	17.12.2022
BVP-B94-D-IDV			İSTİNAT DUVARLARI					553.310				11.06.2021	17.12.2022
BVP-B94-D-ORI			ORMAN İŞLERİ					17.496				01.06.2021	17.12.2022
BVP-B94-D-KB			BETONARME & YAPISAL ÇELİK İŞLERİ									01.02.2022	06.09.2023
			A BLOK BETONARME İMALATLARI					295.320				01.02.2022	04.12.2022
	Kaba İnşaat	B94-KB-BE-1-A	A Blok Betonarme İmalatları - 1.Etap İşleri	1.Etap İşleri	A BLOK	m2	59.778 m²	11.581	42	2	21	01.02.2022	22.02.2022
	Kaba İnşaat	B94-KB-BE-2-A	A Blok Betonarme İmalatları - 2.Etap İşleri	2.Etap İşleri	A BLOK	m2	30.999 m²	23.162	84	2	42	01.04.2022	13.05.2022
	Kaba İnşaat	B94-KB-BE-3-A	A Blok Betonarme İmalatları - 3.Etap İşleri	3.Etap İşleri	A BLOK	m2	14.113 m²	86.659	315	3	105	01.05.2022	14.08.2022
	Kaba İnşaat	B94-KB-BE-4-A	A Blok Betonarme İmalatları - 4.Etap İşleri	4.Etap İşleri	A BLOK	m2	65.025 m²	46.325	168	2	84	13.05.2022	05.08.2022
	Kaba İnşaat	B94-KB-BE-5-A	A Blok Betonarme İmalatları - 5.Etap İşleri	5.Etap İşleri	A BLOK	m2	84.327 m²	34.743	126	2	63	05.08.2022	07.10.2022
	Kaba İnşaat	B94-KB-BE-6-A	A Blok Betonarme İmalatları - 6.Etap İşleri	6.Etap İşleri	A BLOK	m2	45.393 m²	-					
	Kaba İnşaat	B94-KB-BE-7-A	A Blok Betonarme İmalatları - 7.Etap İşleri	7.Etap İşleri	A BLOK	m2	15.054 m²	92.649	336	3	112	14.08.2022	04.12.2022
	Kaba İnşaat	B94-KB-BE-8-A	A Blok Betonarme İmalatları - 8.Etap İşleri	8.Etap İşleri	A BLOK	m2	42.367 m²	-					
			B BLOK BETONARME İMALATLARI					1.200.816				01.02.2022	06.09.2023
			A BLOK YAPISAL ÇELİK İŞLERİ					15.922				22.02.2022	20.12.2022
			B BLOK YAPISAL ÇELİK İŞLERİ					97.202				15.03.2022	01.08.2023
BVP-B94-D-IZO			İZOLASYON İŞLERİ									03.02.2022	28.10.2023
			A BLOK TEMEL İZOLASYON İŞLERİ					1.535				03.02.2022	01.09.2022
			B BLOK TEMEL İZOLASYON İŞLERİ					6.135				06.02.2022	07.01.2023
			A BLOK PERDE İZOLASYON İŞLERİ					11.400				22.02.2022	21.01.2023
			B BLOK PERDE İZOLASYON İŞLERİ					21.792				09.07.2022	28.10.2023
			A BLOK BİNAİÇİ İZOLASYON İŞLERİ					1.044				22.02.2022	20.12.2022
			B BLOK BİNAİÇİ İZOLASYON İŞLERİ					3.289				09.07.2022	02.10.2023
BVP-B94-D-IKB			İLERİ KABA İŞLERİ									22.02.2022	17.06.2023
			A BLOK GAZBETON DUVAR İŞLERİ					53.056				22.02.2022	24.11.2022
	İleri Kaba İşleri	B94-IKB-GD-1-A	A Blok Gazbeton Duvar İşleri - 1.Etap İşleri	1.Etap İşleri	A BLOK	m2	59.778 m²	2.081	30	1	30	22.02.2022	24.03.2022
	İleri Kaba İşleri	B94-IKB-GD-2-A	A Blok Gazbeton Duvar İşleri - 2.Etap İşleri	2.Etap İşleri	A BLOK	m2	30.999 m²	4.161	60	1	60	22.04.2022	21.06.2022
	İleri Kaba İşleri	B94-IKB-GD-3-A	A Blok Gazbeton Duvar İşleri - 3.Etap İşleri	3.Etap İşleri	A BLOK	m2	14.113 m²	15.605	225	3	75	22.05.2022	05.08.2022
	İleri Kaba İşleri	B94-IKB-GD-4-A	A Blok Gazbeton Duvar İşleri - 4.Etap İşleri	4.Etap İşleri	A BLOK	m2	65.025 m²	8.322	120	1	120	03.06.2022	01.10.2022
	İleri Kaba İşleri	B94-IKB-GD-5-A	A Blok Gazbeton Duvar İşleri - 5.Etap İşleri	5.Etap İşleri	A BLOK	m2	84.327 m²	6.242	90	1	90	26.08.2022	24.11.2022
	İleri Kaba İşleri	B94-IKB-GD-6-A	A Blok Gazbeton Duvar İşleri - 6.Etap İşleri	6.Etap İşleri	A BLOK	m2	45.393 m²	-					
	İleri Kaba İşleri	B94-IKB-GD-7-A	A Blok Gazbeton Duvar İşleri - 7.Etap İşleri	7.Etap İşleri	A BLOK	m2	15.054 m²	16.645	240	3	80	04.09.2022	23.11.2022
	İleri Kaba İşleri	B94-IKB-GD-8-A	A Blok Gazbeton Duvar İşleri - 8.Etap İşleri	8.Etap İşleri	A BLOK	m2	42.367 m²	-					
			B BLOK GAZBETON DUVAR İŞLERİ					269.326				15.03.2022	28.05.2023
			A BLOK ALÇIPAN DUVAR İŞLERİ					17.454				22.02.2022	24.11.2022
			B BLOK ALÇIPAN DUVAR İŞLERİ					40.503				15.03.2022	28.05.2023
BVP-B94-D-IKB-SVI			SIVA İŞLERİ										
			A BLOK SIVA İŞLERİ					74.069				27.02.2022	29.11.2022
			B BLOK SIVA İŞLERİ					141.756				04.04.2022	17.06.2023

Şekil 23. Konut binası iş programı ekran görüntüsü

2. Veri Temizleme: Ham veri, farklı biçimlerde ve kalitede olduğundan, ML yöntemleriyle uyumlu hâle getirmek için sistematik bir temizleme sürecine tabi tutulmuştur. Bu aşamada, özellikle "Ardıllar" (Successors) sütununda yer alan eksik ve tutarsız girdiler düzeltilmiştir. Eksik değerler ya da tire (-) ve boşluk karakterleri gibi geçici ifadelerin yerine "İnşaat Bitişi" aktivitesi koyularak modelde nihai aktiviteleri temsil etmesi sağlanmıştır.

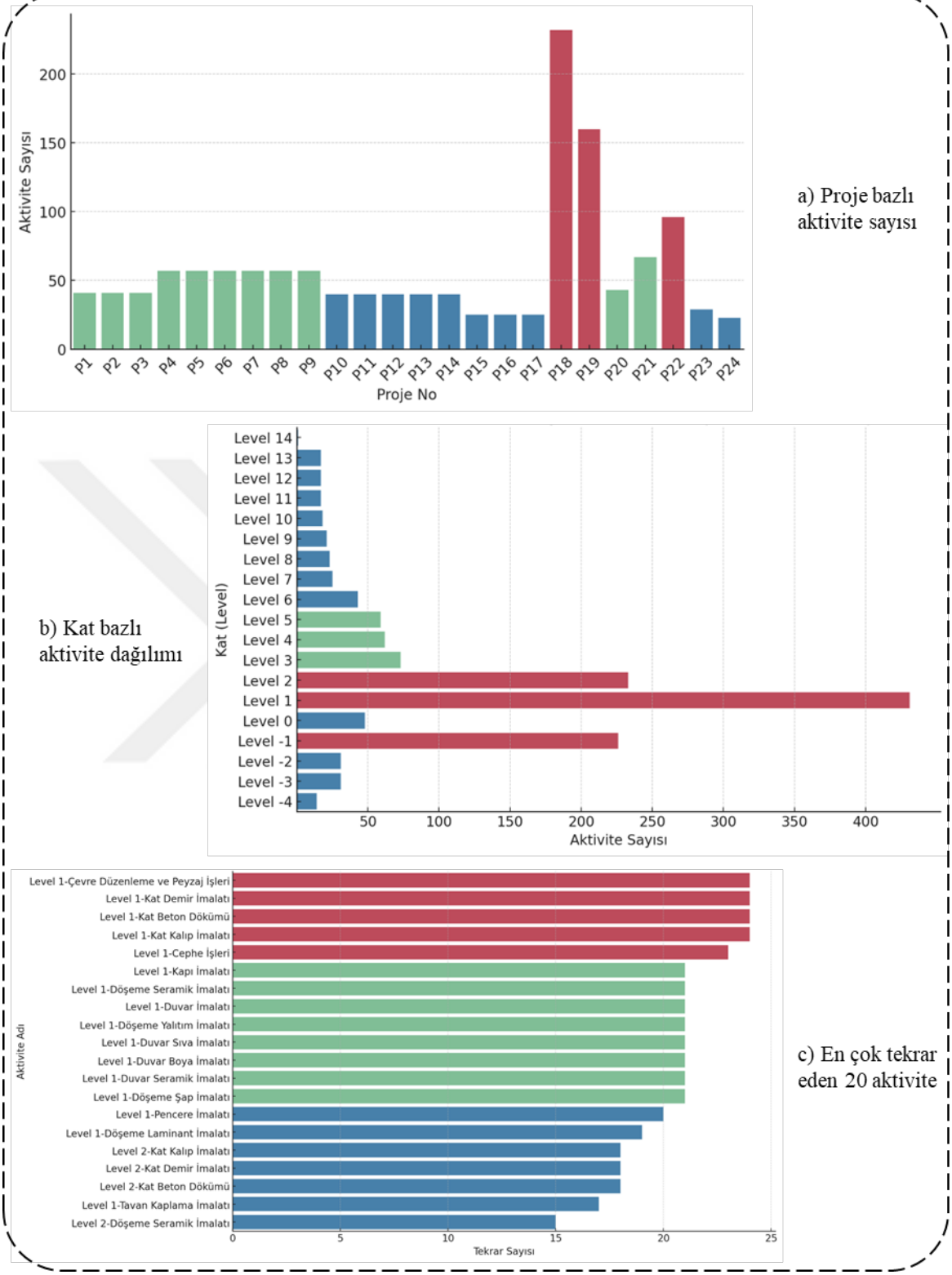
Ayrıca metinsel uyumsuzluklar, özel karakterler ve semboller, analiz sırasında hataya neden olmamaları için ayıklanmış veya dönüştürülmüştür. Çakışan proje numaraları, aynı aktivite için birden fazla kayıt gibi tekrarlar tespit edildiğinde veri setinden çıkarılmış veya birleştirilmiştir. Bu işlem, aktivite isimlerinin ve proje kimliklerinin tutarlı bir veri yapısı içinde yer almasını sağlamış, ilerleyen aşamalarda yürütülecek modelleme süreçlerinde veri kalitesini artırmıştır.

3. Veri Ön İşleme: Veri temizleme aşaması tamamlandıktan sonra, analitik modellemeye uygun bir veri yapısı oluşturmak amacıyla "normalizasyon" ve "yeniden yapılandırma" gibi ön işleme adımları uygulanmıştır. Bu çerçevede, özellikle farklı projelerde aynı anlama gelen ancak farklı adlandırılan aktiviteler, bütünlük ve karşılaştırılabilirlik sağlamak üzere ortak bir terminolojiye dönüştürülmüştür. Böylelikle,

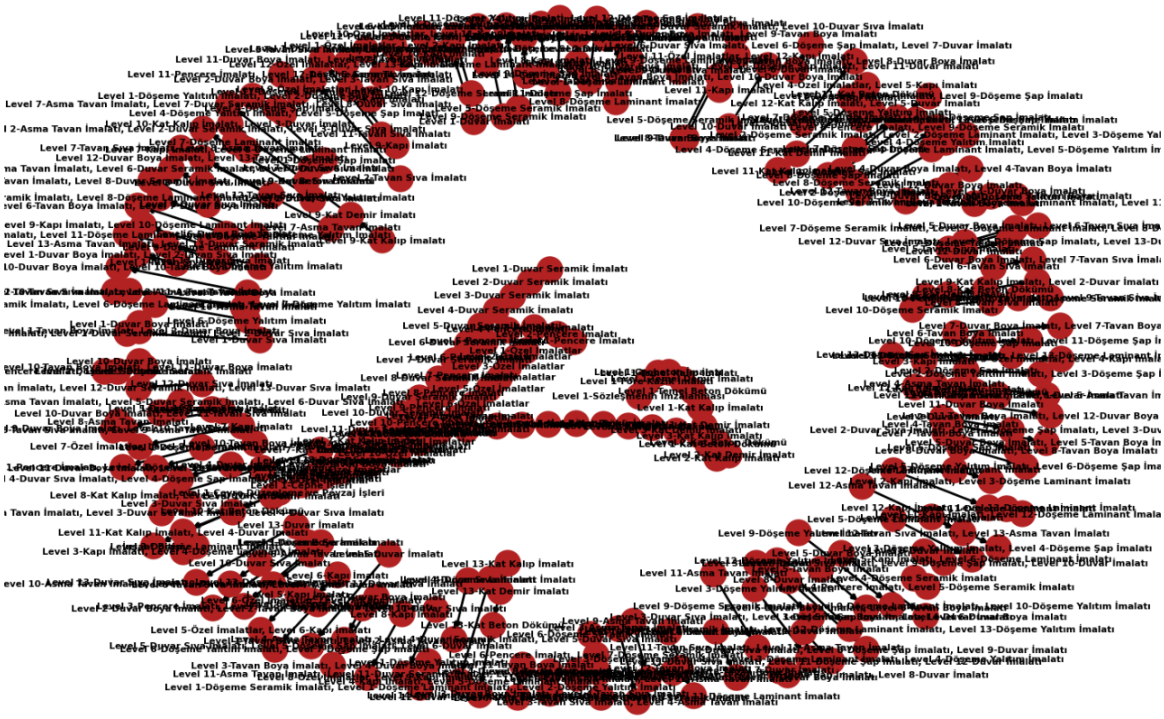
ML modellerinde parametre veya etiket düzeyinde gerçekleştirilen analizlerin tutarlılığı ve karşılaştırılabilirliği artırılmıştır.

Veri dönüşümü ve normalizasyonu kapsamında, inşaat projelerinin iş programlarında yer almasına rağmen analizde kullanılmayacak olan gereksiz veya tekrarlı bilgiler ayıklanmıştır. Buna karşılık, "Aktivite Adı" (Activity Name), "Proje Numarası" (Project Number), "Seviye" (Level) ve "Ardıllar" (Successors) gibi kritik sütunların bütünlüğü ve mantıksal tutarlılığı özel olarak kontrol edilmiştir. Her proje numarası, geçerli aktivitelerle eşleştirilmiş ve çelişkili kayıtlar veri setinden çıkartılmış ya da birleştirilmiştir. Bu sayede, başlangıçta farklı formatlarda ve değişken kalitede elde edilen veriler, yalnızca modellemede gerekli sütunları koruyacak biçimde standardize edilmiş ve 24 projeden toplanan toplam 1390 aktivitenin yer aldığı tutarlı bir veri kümesi oluşturulmuştur.

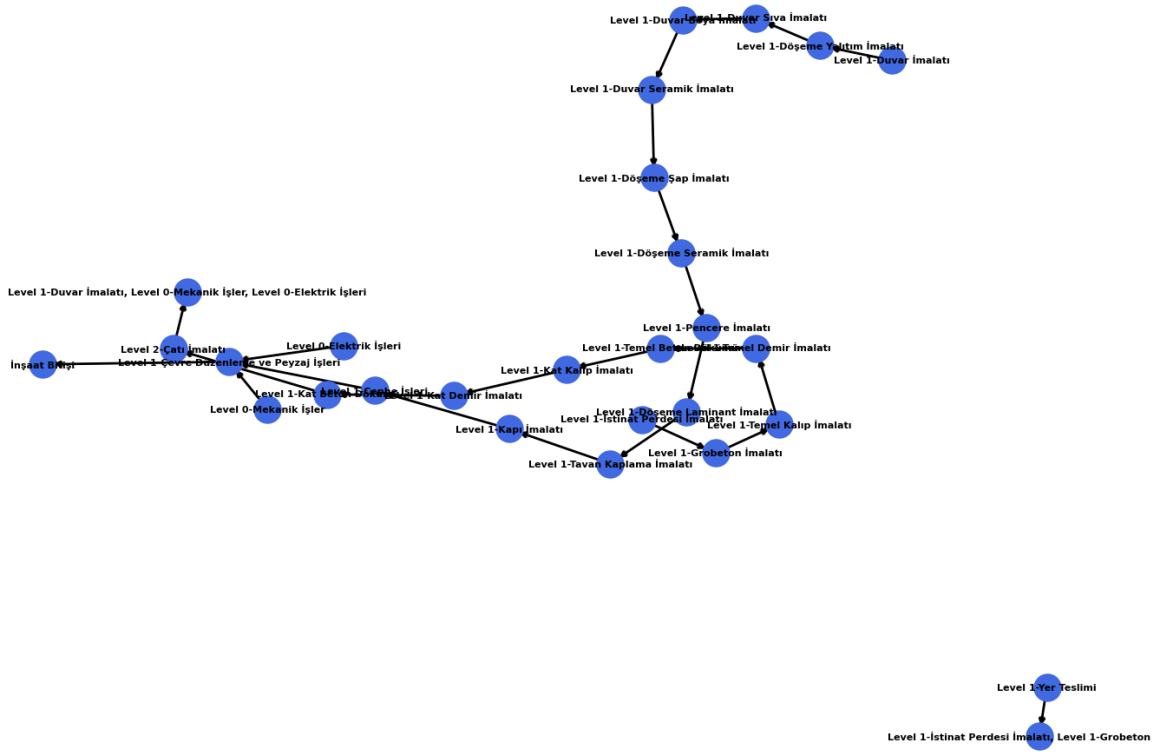
Ayrıca, iş programlarında yer alan aktivitelerin isimlendirilmesi hem BIM modelleriyle hem de 24 farklı iş programı arasında tutarlılık sağlamaya yönelik bir şablon üzerinden yürütülmüştür. Bu amaçla, "Seviye" (örneğin, "Level 1", "Level 2"), "Eleman Adı" (örneğin, "Grobeton", "Seramik", "Temel", "Kiriş" vb.) ve ilgili "İşlem" (örneğin, "İmalatı", "Kalıp İmalatı", "Beton Dökümü" vb.) bileşenleri kullanılarak sistematik bir adlandırma şeması oluşturulmuştur. Örneğin, "Level -1-Grobeton İmalatı" veya "Level -1-Temel Kalıp İmalatı" gibi tanımlamalar, proje genelinde düzenli bir terminoloji sağlanmış; böylece, kat seviyesi, yapı elemanı ve ilgili faaliyetin açık biçimde ifade edilmesi mümkün hâle gelmiştir. Söz konusu yaklaşım, projeler arası karşılaştırma yapılabilmesini ve ML sürecinde veri tutarlılığını artırmayı hedefleyen bütüncül bir yöntem sunmuştur. Şekil 24'te yukarıdaki bölümlerde değinilen çalışmalar sonucu elde edilmiş olan veri grubu gösterilmektedir. Bu kapsamda, Şekil 24(a) proje bazlı toplam aktivite sayılarını, (b) kat bazlı aktivite dağılımını ve (c) en çok tekrar eden aktiviteleri göstermektedir. Şekil 25'te ise, proje bazlı aktivite geçiş ağları sunulmaktadır. Grafikte, her projenin aktiviteleri düğümler olarak yer alırken, aktiviteler arasındaki geçişler yönlü kenarlarla gösterilmiştir. En yoğun aktivite geçişlerine sahip proje P18 ile en az aktivite içeren projelerden bir tanesi olan P16 gösterilerek, projeler arasındaki iş hacim farklılıkları vurgulanmakta ve aktivite yoğunluğunu anlama açısından sırası ile Şekil 25(a) ve (b)'de ilgili projelere yönelik görselleştirmeler sunulmaktadır.



Şekil 24. (a) Proje bazlı aktivite sayıları, (b) kat bazlı aktivite dağılımı ve (c) en çok tekrar eden aktiviteler



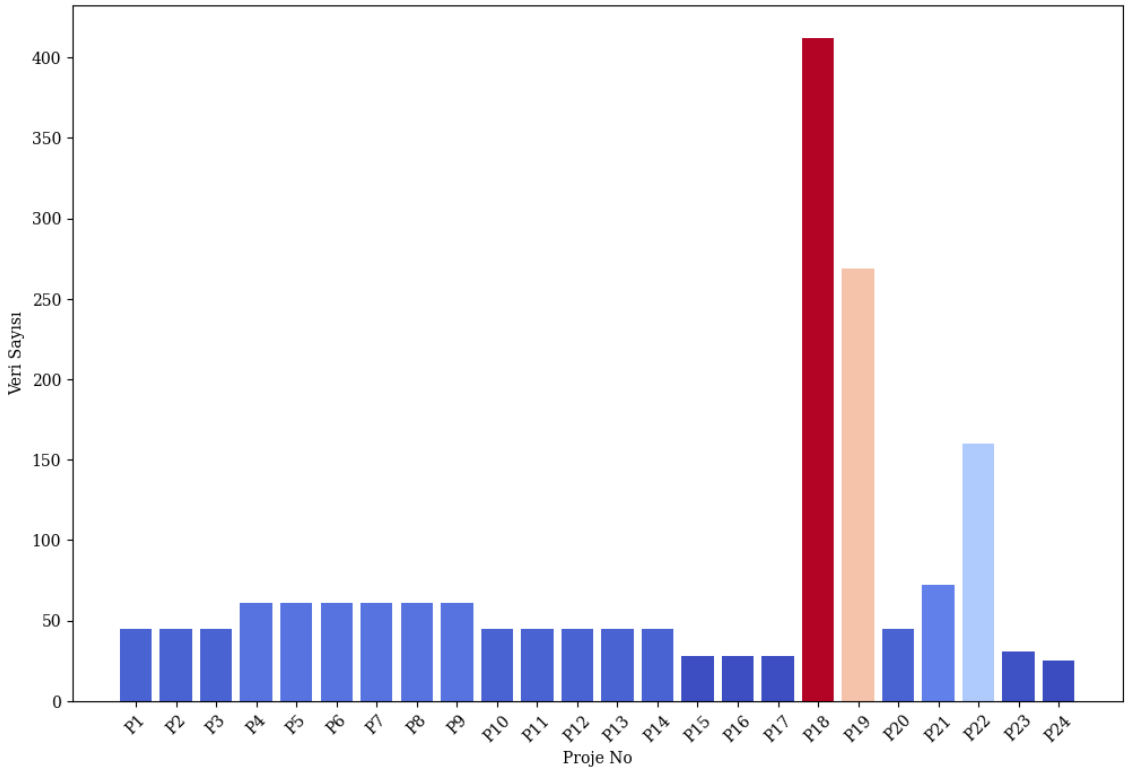
a) P18 projesine ait aktivite geçiş ağı



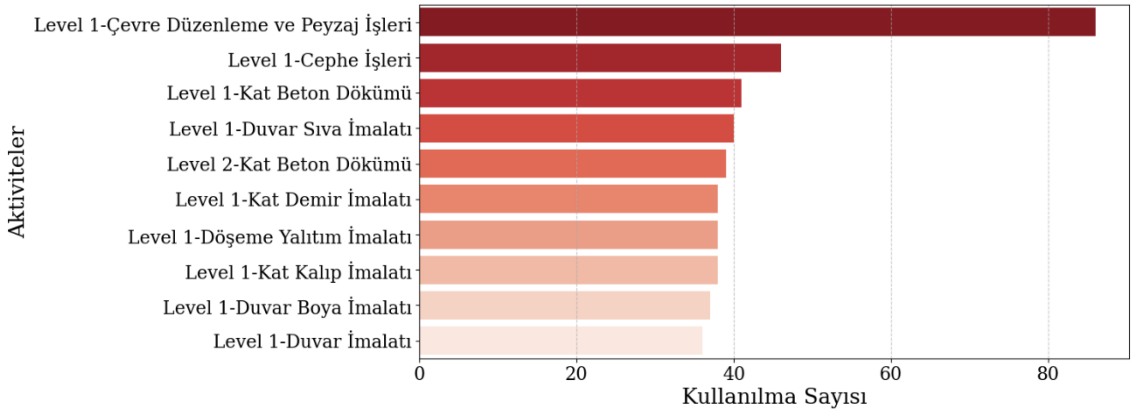
b) P16 projesine ait aktivite geçiş ağı

Şekil 25. (a) P18 ve (b) P16 projelerine ait aktivite geçiş ağları

"Ardıllar" sütununda birden fazla aktivite adı bulunduğu durumlarda (Aktivite Adı (A) → Ardıl (B, C)), her bir ardıl kaydı ayrı bir satıra dönüştürülerek "Aktivite Adı (A) → Ardıl (B)" ve "Aktivite Adı (A) → Ardıl (C)" biçiminde işlenmiştir. Bu sayede, bir faaliyetin birden fazla ardılı olması hâlinde her ardıl için yeni bir satır üretilmiş ve ağ tabanlı modellerde sıklıkla başvurulan kenar listesi (edge list) ve ilişki listesi (adjacency list) yapılarının oluşturulması kolaylaştırılmıştır. Böylelikle, her "aktivite-ardıl" ilişkisi bağımsız bir işlem (transaction) olarak kaydedilmiş ve ağ temelli modelleme tekniklerinde kullanım için gerekli sadelikte, ancak işlevsel bir veri şeması elde edilmiştir. Bu işlemler sonrasında veri kümesindeki toplam veri sayısı 1824'e ulaşmıştır ve her satır, tek bir "aktivite-ardıl" çiftini temsil edecek şekilde düzenlenmiştir. Şekil 26'da düzenlenmiş veri üzerinde her projedeki yeni veri (aktivite) sayıları, Şekil 27'de ise aktivite ve ardıl sütunlarında en fazla bulunan aktiviteler gösterilmektedir.



Şekil 26. Veri ön işleme sonrası proje bazlı veri sayısı



Şekil 27. Aktivite adı ve ardıl sütunlarında en fazla bulunan aktiviteler

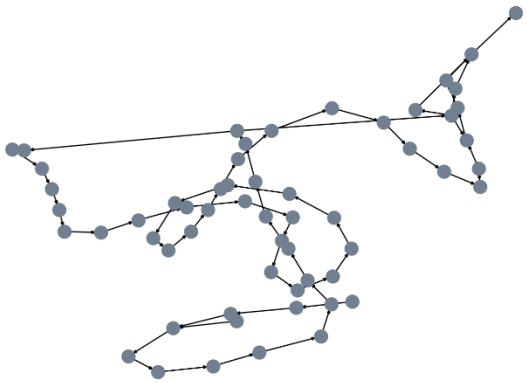
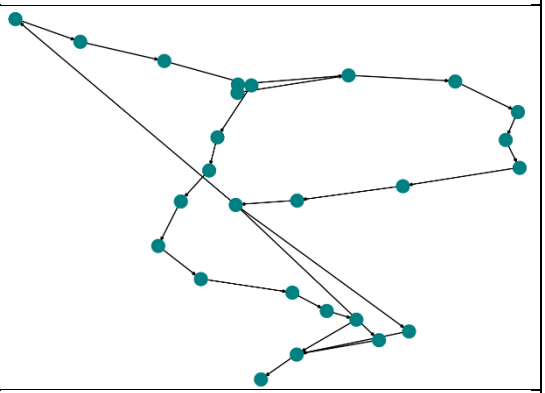
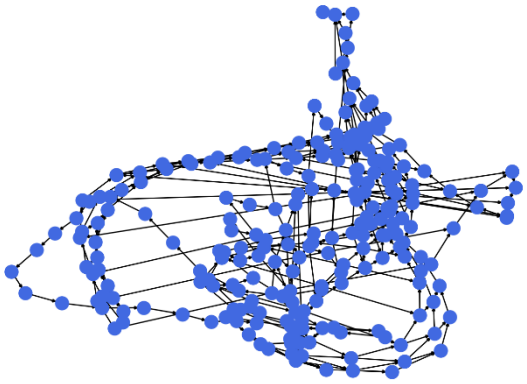
Grafik tabanlı öğrenme yöntemlerine geçiş için bir sonraki önemli adım, ağın komşuluk matrisinin (adjacency matrix) oluşturulmasıdır. Komşuluk matrisi, benzersiz düğümlerin sayısı ile eşit boyutlara sahip bir kare matristir ve bu matriste bir düğümden başka bir düğüme yönlendirilmiş bir kenar mevcutsa ilgili giriş "1", aksi takdirde "0" olarak tanımlanmaktadır (Kaczmarek, 2023). GNN modellerinde yalnızca doğrudan komşu bilgisi değil, düğümlerin kendi iç özellikleri de dikkate alındığından kimlik matrisi (identity matrix) eklenerek öz-bağlantılar (self-loops) oluşturulmuştur. Öz-bağlantılar sayesinde her düğüm, evrişimsel (convolutional) toplama işlemi sırasında kendi özelliklerini de modele dâhil edebilmektedir (Lampert & Scholtes, 2023).

Devamında, artırılmış komşuluk matrisinin her satırının toplamı alınarak derece matrisi (degree matrix) elde edilmiş ve ardından yüksek dereceye sahip düğümlerin orantısız şekilde ağırlık kazanmasını önlemek amacıyla bir normalizasyon uygulanmıştır. Normalleştirilmiş komşuluk matrisi, PyTorch tensörü biçimine dönüştürülerek GNN modeline girdi sağlayabilecek seviyeye getirilmiştir. Düğüm özellikleri ise, düğüm sayısına eşit boyutta bir kimlik matrisi (identity matrix) olarak başlatılmıştır. Bu yöntem, her düğümün tekil bir birim vektörle (one-hot encoding) temsil edilmesini sağlamakta ve ağ üzerinden daha ileri düzeyde gömme (embedding) öğrenimi için temel bir başlangıç noktası oluşturmaktadır.

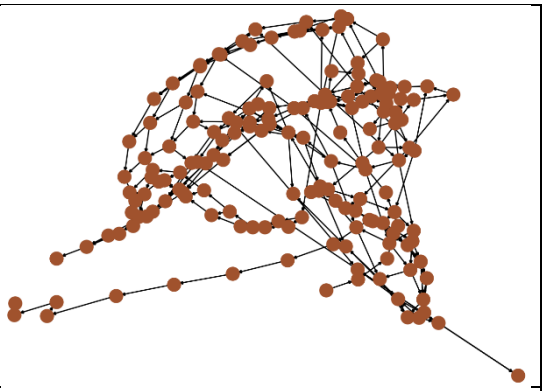
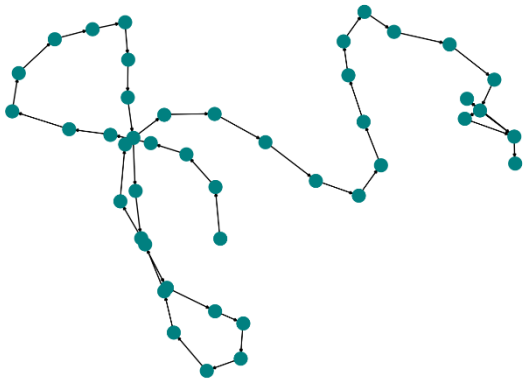
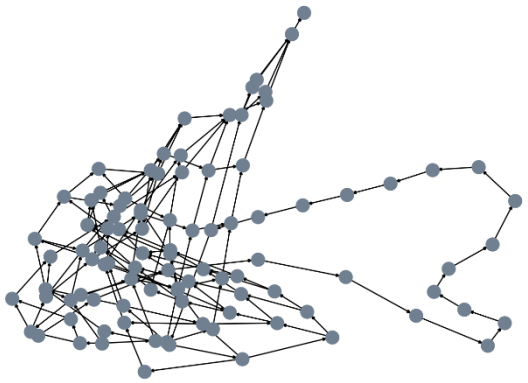
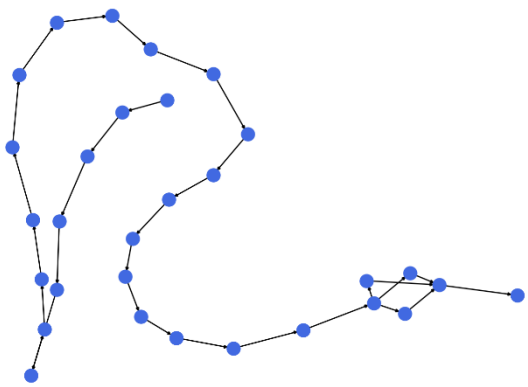
Bu yaklaşım, özellikle grafik tabanlı analizlerle entegre çalışan ML modellerinde (örneğin, GCN, GAT) ardıl tahmini, ağ merkeziyet ölçümleri veya ARM gibi yöntemleri başarılı bir şekilde uygulamaya imkân vermektedir. Tablo 2'de, söz konusu veri hazırlığı sonucunda elde edilen projelerin grafik görünümü örneklenmekte ve bu sayede sıralı (sequential) ve ağ temelli modelleme çalışmalarında elde edilebilecek bilgilerin görsel bir

özeti sunulmaktadır. Böylece, başlangıçta farklı biçim ve kalitede bulunan ham veriler, analitik modellerin doğruluğunu ve genellenebilirliğini artıracak bir yaklaşımla yeniden düzenlenmiştir. Proje yönetimi ve planlama alanında veriye dayalı karar alma süreçlerini güçlendiren bu kapsamlı ön işleme çalışması, ARM, ağ analizi ve tahmine dayalı yöntemler gibi tekniklerin etkin biçimde uygulanabilmesi için gerekli veri temelini sağlamıştır. Süreç boyunca yürütülen veri ön işleme çalışmalarıyla, veri kalitesi arttırılmış ve ML modellerinin performansı iyileştirilmiştir.

Tablo 2. Veri ön işleme sonrası projelere ait bilgiler ve verilerin graf görünümleri

Proje Numarası	**Bina Türü**	**Kat Sayısı**	**Veri Sayısı**	**Graf Görünümü**
P9	Villa-9	Bodrum + Zemin + 1 Kat	61	
P16	Villa-16	Zemin	28	
P18	Konut ve İş Yeri	Zemin + 13 Kat	412	

Tablo 2'nin devamı

P19	Hastane - A Blok	3 Bodrum + Zemin + 5 Kat	269	
P20	Kamu Binası (Yalnızca Betonarme)	2 Bodrum + Zemin + 7 Kat	45	
P22	Konut	Zemin + 4 Kat	160	
P23	Okul (Yalnızca Betonarme)	Zemin + 5 Kat	31	

4. Veri Kümesinin Bölünmesi: Veri setinin eğitim ve test alt kümelerine ayrılması, oluşturulacak modellerin genellenebilirliğini ve performansını güvenilir biçimde değerlendirmek açısından kritik bir adımdır. Bu süreçte, özellikle proje tabanlı verilerin

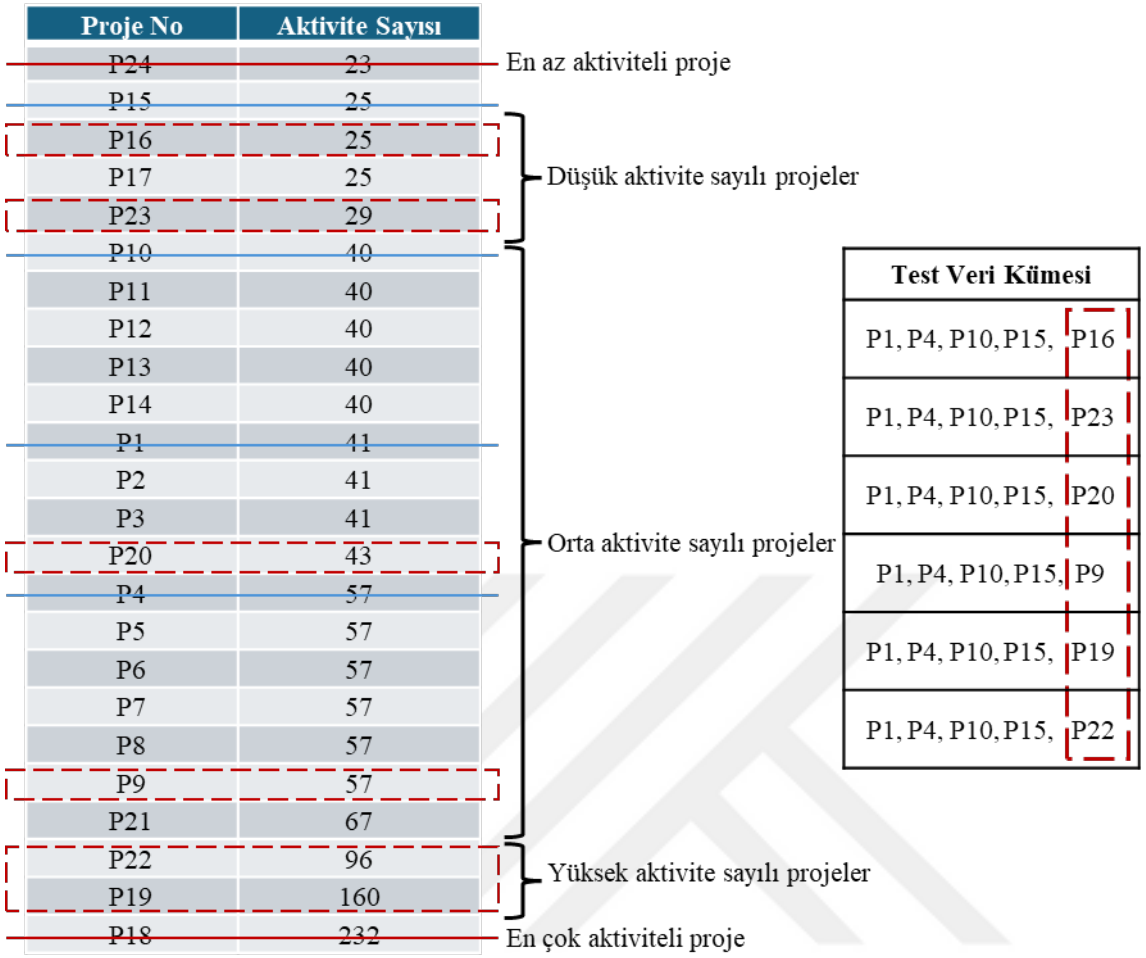
(örneğin, kat sayısı, aktivite yoğunluğu) modele yansıtılmasında dikkatli bir strateji izlenmiş ve proje özellikleri esas alınarak eğitim ve test kümeleri ayrımı yapılmıştır. Toplam 24 projenin 19'u eğitim (%80), geriye kalan 5'i ise test (%20) kümesine tahsis edilmiştir.

ML modellerinin çıktısı olarak, her bir projenin verileri üzerinde gerçekleştirilecek aktivite sıralaması test projelerinden elde edilecektir. Dolayısıyla, test projelerinin, 24 projeden oluşan tüm veriyi temsil edecek şekilde oluşturulmasına özen gösterilmiştir. Öte yandan, en yüksek ve en düşük aktiviteye sahip projeler, ML modellerinin eğitimi açısından daha büyük önem taşıdığı kabul edilerek test kümesinden çıkarılmış ve doğrudan eğitim kümesinde konumlandırılmıştır.

Toplanan projelerin boyutları ve çalışma kapsamı dikkate alındığında, ML model çıktılarının daha çok küçük ve orta ölçekli inşaat projeleri için kullanılabileceği düşünülmektedir. Dolayısıyla, test kümesinin bunu sağlayabilecek nitelikte olması gerekmektedir. Kat sayısı ve aktivite düzeyi orta veya nispeten küçük olan projelerin test setinde bulunması gerektiği düşünüldüğünden, P1, P4, P10 ve P15 numaralı projeler test kümesi için seçilmiştir. Son test projesi ise, farklı aktivite yoğunluk kategorilerine (en düşük, orta ve en yüksek) sahip projeler arasından rastgele belirlenmiştir. Böylece, farklı yoğunluk seviyeleri göz önüne alınarak oluşturulan test kümesi, modelin çeşitli ölçek ve karmaşıklıktaki projelerdeki performansını ölçmeye elverişli hâle getirilmiştir.

Ayrıca, modellerin ve projelerin genellenebilirliği daha kapsamlı biçimde test edebilmek ve çıktılarda aktivite sayısı bakımından çeşitlilik sağlamak amacıyla, her yoğunluk kategorisinden ikişer proje seçmek suretiyle toplam altı farklı eğitim-test bölünmesi (senaryosu) kurgulanmıştır. Bu uygulamayla, farklı kombinasyonlardaki eğitim ve test kümeleri üzerinde modeller eğitilerek, model çıktılarıyla modellerin doğruluğu test edilmiştir.

Bu strateji, proje bazında veri sızıntısı riskini asgari düzeye indirirken, modellerin tamamen yeni ve farklı ölçeklerdeki projelerde de doğru tahmin ve sıralama yapabilmesini sağlamayı hedeflemiştir. Seçilen projelerin dağılımı ve farklı senaryolar altındaki eğitim-test ayrımları, Şekil 28'de ayrıntılı biçimde gösterilmektedir.



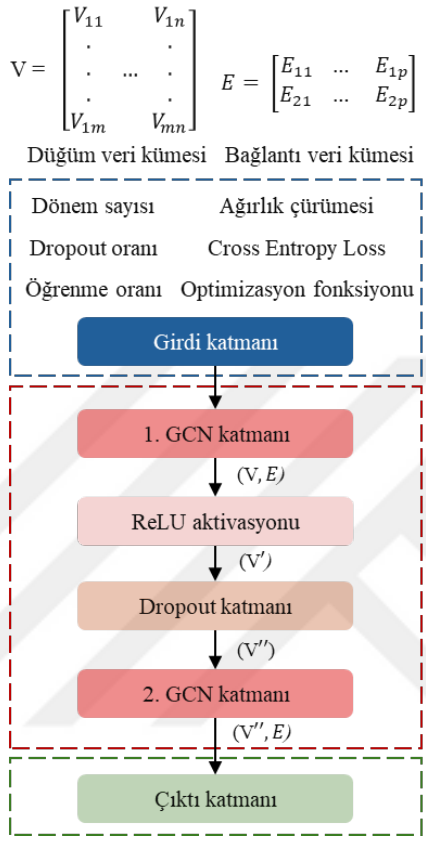
Şekil 28. Eğitim ve test kümelerinin belirlenme aşamaları

2.2.2. GCN Modelinin Yapılandırılması

GCN modelinin tasarımı, Kipf & Welling (2016) tarafından önerilen çalışmadan esinlenerek inşaat aktiviteleri alanına uyarlanmıştır. Bir GCN mimarisi, bir bağlantı tahmin çerçevesi içinde uygulanır ve değerlendirilir. GCN modeli, düğüm ve kenar tanımlamaları yoluyla bir grafik inşa ederek ham faaliyet verilerini işler ve evrişimsel işlemleri yönlendiren bir komşuluk matrisi hazırlar.

GCN modeli, Şekil 29'da gösterildiği gibi iki ardışık grafik evrişim katmanı ile yapılandırılmıştır. İlk katman, komşu düğüm özelliklerinin normalleştirilmiş komşuluk matrisi ($\hat{A}$) ile çarpımı yoluyla birleştirilmesinin ardından, girdi özellikleri üzerinde öğrenilebilir ağırlık matrisi (learnable weight matrix) kullanarak doğrusal bir dönüşüm gerçekleştirir. Bu dönüşüm, ReLU aktivasyon fonksiyonu ile takip edilerek modele doğrusal olmayan özellikler kazandırılır. İkinci katman ise gizli temsilleri daha ileri

düzeyde dönüştürerek son düğüm yerleştirmelerini (node embeddings) üretir. Bu yerleştirmeler hem orijinal düğüm özelliklerini hem de aktivite ağının yapısal ilişkilerini içeren öğrenilmiş temsilleri yansıtır.



Şekil 29. GCN model mimarisi

Graf ve düğüm özelliği temsilleri oluşturulduktan sonra, iki düğüm (aktivite) arasındaki kenar varlığı, ikili sınıflandırma problemi olarak ele alınarak bağlantı (link) tahmini görevi tanımlanır. Pozitif örnekler, eğitim verisinden çıkarılan gerçek kenarlara karşılık gelirken; negatif örnekler, eğitim grafiğinde bağlantısı olmayan düğüm çiftlerinin rastgele örneklenmesiyle oluşturulmuştur. Veri dengesinin sağlanması amacıyla, negatif kenar sayısı pozitif kenar sayısıyla eşit olacak şekilde belirlenmiştir. Pozitif ve negatif kenar indisleri (yani kaynak ve hedef düğüm kimlikleri) birleştirilerek, her birine karşılık gelen ikili etiketler (pozitif kenarlar için 1, negatif kenarlar için 0) atanmıştır.

GCN modeli, ikili çapraz - entropy kaybı (binary cross-entropy loss) ile standart denetimli öğrenme yaklaşımı kullanılarak eğitilmiştir. Her bir eğitim döneminde (epoch), model, normalleştirilmiş komşuluk matrisi ve düğüm özelliklerini işleyerek düğüm yerleştirmelerini hesaplamaktadır. Her bir eğitim kenarı için, ilgili kaynak ve hedef düğüm

yerleştirmeleri çıkarılarak, iki düğüm arasındaki bağlantının olasılığını temsil eden bir skor değeri üretmek amacıyla nokta çarpımı (dot product) hesaplanmıştır. Modelin parametreleri, 0.01 öğrenme oranı ile Adam optimizasyon algoritması kullanılarak güncellenmiştir. Eğitim süreci 500 dönem/iterasyon boyunca tekrarlanmış, kayıp değeri periyodik olarak kaydedilerek modelin yakınsama durumu izlenmiştir. Bu eğitim döngüsü, benzer faaliyetlerin (grafik yapısına göre) yerleştirme vektörlerinin, nokta çarpımı aracılığıyla var olan bağlantılar için yüksek benzerlik skorları üretmesini sağlayacak şekilde evrimleşmesini güvence altına almıştır.

Elde edilen öğrenilmiş yerleştirmeler, bağlantı tahmini görevini değerlendirme aşamasında kullanılmıştır. Her bir test faaliyeti (test verisinden çıkarılmış ve benzer ön işleme adımlarından geçirilmiş), tahmin fonksiyonu tarafından değerlendirilmiştir. Belirli bir kaynak faaliyetin yerleştirilmesi ile tüm aday ardılların yerleştirmeleri arasındaki nokta çarpımı benzerlikleri hesaplanarak, en olası ardıllar belirlenmiştir. Bu çalışmada, en olası üç ardıl tahmin edilmiştir. Tahmin edilen ardıllar, test veri kümesindeki gerçek ardıllarla karşılaştırılarak modelin performansı değerlendirilmiştir.

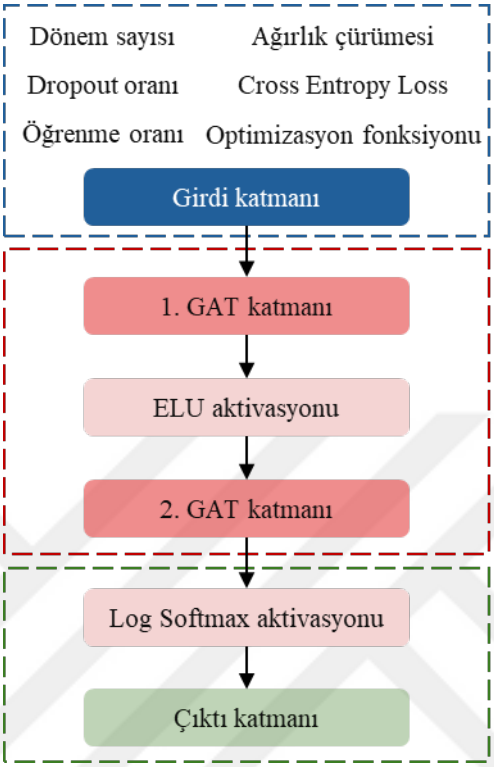
GCN modelinin en uygun mimarisini belirlemek amacıyla, farklı gizli katman boyutları (hidden sizes) (16, 32, 64) ve yerleştirme boyutları (embedding dimensions) (16, 32, 64) ile modelin tahmin performansı incelenmiştir. Eğitim süreci boyunca 500 iterasyon (number of epochs) ve 0.01 öğrenme oranı (learning rate) kullanılmıştır. Önerilen GCN modelinin Python kodu detayları Ek 1'de verilmektedir.

2.2.3. GAT Modelinin Yapılandırılması

Yapılan çalışmalar, GCN modeline dikkat mekanizmasının eklenmesinin, modelin tahmin doğruluğunu artırabileceğini ortaya koymuştur (Mostofi & Toğan, 2023). Bu doğrultuda, bu çalışmada grafik tabanlı yaklaşım genişletilerek, inşaat faaliyet sıralaması problemine yönelik GAT modeli uygulanmış ve önceki GCN modeliyle karşılaştırılmıştır.

Her iki model ortak bir veri ön işleme ve eğitim çerçevesini paylaşmasına rağmen, GAT modeli, statik normalleştirilmiş komşuluk matrisi yerine ikili (binary) komşuluk matrisi kullanmakta ve dikkat mekanizmasını entegre ederek komşu düğümlerin önem derecelerini dinamik olarak belirlemektedir. Bu farklılık nedeniyle, GAT modelinin öğrenme dinamikleri, GCN modelinden farklılaşmakta ve sabit derece tabanlı ağırlıklar yerine, modelin dikkat katsayıları aracılığıyla belirli komşulara odaklanmasını

sağlamaktadır (Rahman & Azad, 2021). Şekil 30'da GAT modelinin mimarisi detaylandırılmaktadır.



Şekil 30. GAT model mimarisi

GAT modeli, iki dikkat katmanından (attention layers) oluşan bir mimari ile tanımlanmıştır (Mostofi, 2024). İlk katman, giriş düğüm özelliklerine doğrusal bir dönüşüm uygular. Bu dönüşüm, başlangıçta rastgele başlatılan ve eğitim sürecinde öğrenilen bir ağırlık matrisi kullanılarak gerçekleştirilir. Daha sonra, model, dönüşüme uğramış özelliklerden çiftler halinde kombinasyonlar oluşturarak dikkat mekanizması girdisini üretir. Temel olarak, grafikteki her olası düğüm çifti için, ilgili özellik temsilleri birleştirilerek dikkat mekanizmasına giriş olarak sunulacak bir vektör oluşturulur.

Öğrenilebilir bir parametre vektörü, her düğüm çifti için bir dikkat skoru (attention score) hesaplamak üzere nokta çarpımı uygular ve bu işlem sonucunda ham dikkat katsayılarından oluşan bir matris elde edilir (Zheng vd., 2022). Modelin ikili komşuluk matrisi (binary adjacency matrix) tarafından belirlenen düğüm komşuluk ilişkilerine bağlı olarak, bağlantısı bulunmayan düğümlerin nihai temsile katkıda bulunmaması sağlanır. Bunun için, bağlantısı olmayan düğümler arasındaki dikkat katsayıları, -9e15 gibi büyük

bir negatif değerle maskelenir (Lin vd., 2021). Ardından, softmax fonksiyonu uygulanarak her satır boyunca dikkat katsayıları normalize edilir.

Dikkat ağırlıkları hesaplandıktan sonra, komşu düğümlerin dönüşüme uğramış özellikleri bu ağırlıklarla çarpılarak düğümün yerel komşuluk bilgisi toplanır (Ma vd., 2020). Katmanlar arasında doğrusal olmayan özelliklerin modellenmesini sağlamak amacıyla üstel doğrusal birim (Exponential Linear Unit, ELU) aktivasyon fonksiyonu uygulanmıştır. İkinci GAT katmanı, birinci katmanda elde edilen toplanmış özellikleri daha da iyileştirerek nihai düğüm yerleştirmelerini (node embeddings) üretir (Zangari vd., 2021). Bu yerleştirmeler hem faaliyetlerin kendilerine ait özelliklerini hem de çevresindeki ağın etkisini yansıtan öğrenilmiş temsiller sağlar. Bu süreç, dikkat mekanizması tarafından belirlenen komşuların etkisini dikkate alarak gerçekleştirilir.

GCN modelinde olduğu gibi, bağlantı tahmini görevi, eğitim verisinden elde edilen pozitif kenarlar ve bağlantısı olmayan düğüm çiftlerinden rastgele örneklenerek oluşturulan negatif kenarlarla yapılandırılmıştır. Pozitif ve negatif örnekler birleştirilerek, her biri için ikili etiket (pozitif kenarlar için 1, negatif kenarlar için 0) atanmıştır.

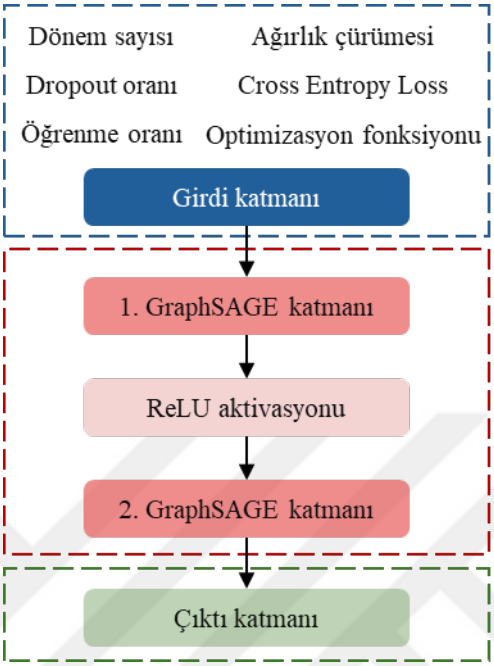
Modelin performansını optimize etmek için, GAT modelinin farklı gizli katman boyutları (16, 32, 64) ve yerleştirme boyutları (16, 32, 64) ile eğitilmesi sağlanmıştır. Modelin parametreleri, 0.01 öğrenme oranı ile Adam optimizasyon algoritması kullanılarak güncellenmiş ve eğitim süreci 500 dönem boyunca sürdürülerek modelin performansı analiz edilmiştir. Önerilen GAT modelinin Python kodu detayları Ek 2'de verilmektedir.

2.2.4. GraphSAGE Modelinin Yapılandırılması

GraphSAGE modeli, görülmemiş düğümlere veya değişen grafik yapılarına genelleme yapabilme kapasitesi sayesinde özellikle avantajlı bir indüktif öğrenme mekanizması sunmaktadır (Arya vd., 2020). GCN ve GAT modelleri, düğüm komşularının özelliklerini sabit bir normalizasyon veya dikkat tabanlı ağırlıklandırma ile işlerken, GraphSAGE, komşuluk bilgilerini nasıl birleştireceğini doğrudan öğrenmektedir (Rahman & Azad, 2021).

GraphSAGE modeli, komşu düğüm özelliklerinin toplanması, düğüm özelliklerinin birleştirilmesi, dönüştürme ve aktivasyon işlemi olmak üzere üç aşamalı bir ileri besleme (forward pass) süreci uygulamaktadır (Ay, 2023; Şahin 2023). Bu süreç, ardışık olarak

uygulanan iki GraphSAGE katmanı ile gerçekleştirilmiştir. Şekil 31'de çalışmada kullanılan GraphSAGE mimarisi gösterilmektedir.



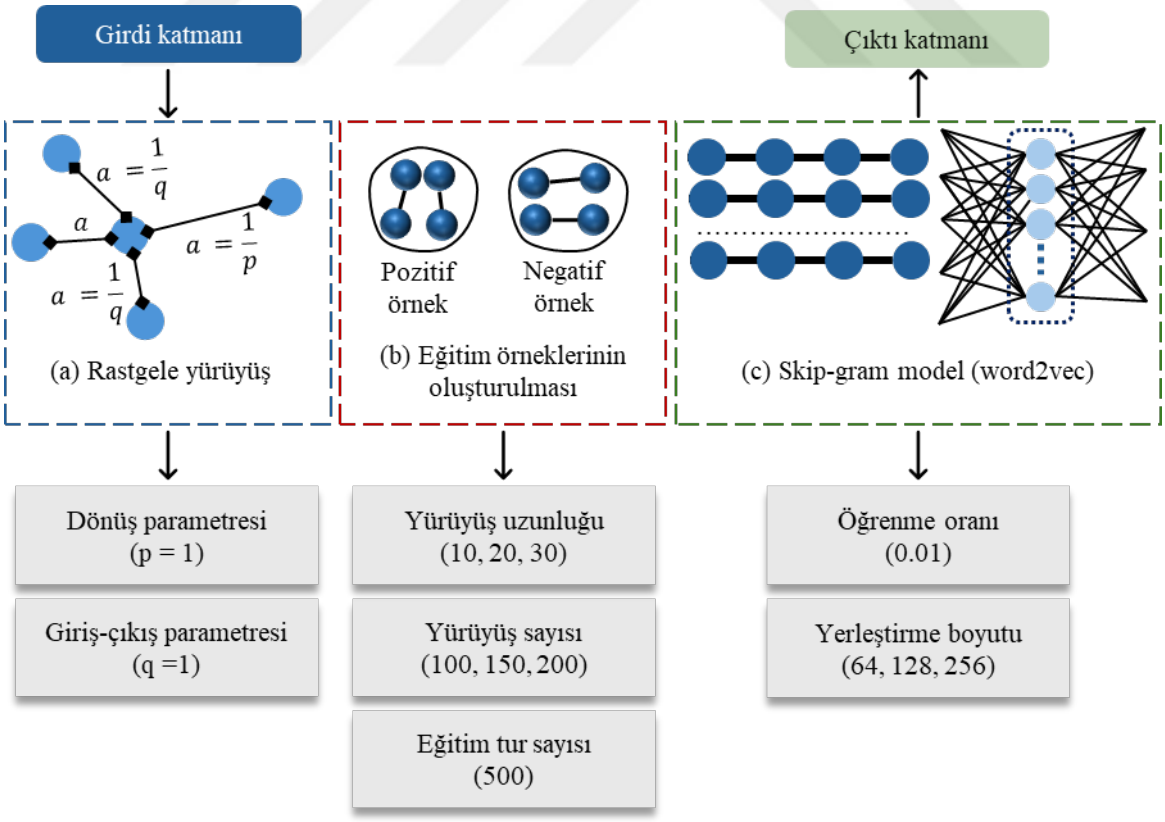
Şekil 31. GraphSAGE model mimarisi

İlk katmanda komşu düğüm özelliklerini birleştirmek için komşuluk matrisi ile düğüm özellik matrisi arasında matris çarpımı gerçekleştirilmektedir. Toplanan komşu düğüm özellikleri, ilgili düğümün kendi özellikleriyle birleştirilerek özellik boyutu iki katına çıkarılmaktadır. Bu birleştirilmiş vektör, bir doğrusal dönüşüm ve ReLU aktivasyon fonksiyonundan geçirilerek işlenmektedir. İkinci GraphSAGE katmanı, bu işlemi tekrarlayarak düğüm temsillerini daha da geliştirmekte ve nihai yerleştirmeleri üretmektedir. Bu tür bir öğrenilebilir toplama fonksiyonu, GraphSAGE'in eğitim sırasında görmediği düğümlere genelleme yapmasını sağlamakta ve modeli özellikle indüktif öğrenme görevleri için uygun hale getirmektedir (Rahman & Azad, 2021).

GCN ve GAT modellerinde olduğu gibi, GraphSAGE modeli de Adam optimizasyon algoritması ile 0.01 öğrenme oranı kullanılarak eğitilmiştir. Eğitim süreci 500 iterasyon boyunca sürdürülmüş ve modelin performansı detaylı olarak incelenmiştir. Gizli katman boyutları (16, 32, 64) ve yerleştirme boyutlarının (16, 32, 64) model verimliliği üzerindeki etkisi kapsamlı bir şekilde değerlendirilmiştir. Önerilen GraphSAGE modelinin Python kodu detayları Ek 3'te verilmektedir.

2.2.5. Node2vec Modelinin Yapılandırılması

GNN modellerine benzer şekilde, node2vec ardılları bağımsız kayıtlar hâline dönüştürmüş ve her satırın tek bir aktivite-ardıl çiftini temsil ettiği bir veri kümesi oluşturmuştur. Bu yapılandırılmış veri, düğümlerin benzersiz aktiviteleri ve yönlendirilmiş kenarların ardıl ilişkilerini ifade ettiği yönlendirilmiş bir graf oluşturmak için kullanılmaktadır. Node2vec modeli, belirli boyutlardaki yerleştirme vektörleri, her bir rastgele yürüyüşte (random walk) yer alacak düğüm sayısını belirleyen yürüyüş uzunluğu (walk length) ve her bir aktivite düğümü için yapılacak rastgele yürüyüş sayısı (number of walks) ile başlatılmıştır (Mostofi vd., 2025). Bu süreçte p ve q değerleri, hem genişlik öncelikli (breadth-first) hem de derinlik öncelikli (depth-first) arama stratejilerini dengelemek için ayarlanmıştır. Model, tüm düğümlerin göz önünde bulundurulmasını sağlamak amacıyla bir skip-gram yaklaşımı ile eğitilmiştir. Bu işlem, aktiviteler arasındaki gizli yapısal ve bağlamsal ilişkileri yakalayan düğüm yerleştirmelerini oluşturmaktadır (Kaczmarek, 2023). Şekil 32, çalışmada kullanılan node2vec mimarisi gösterilmektedir.



Şekil 32. Node2vec model mimarisi

Tahmin aşamasında, herhangi bir aktivite için en benzer aktiviteleri (yani tahmin edilen ardılları) tespit etmek üzere kosinüs benzerliği (cosine similarity) fonksiyonu kullanılmıştır. Bu fonksiyon, gömme uzayında en benzer ilk üç aktiviteyi döndürerek bir sıralama önerisi sunar (You vd., 2021). Ayrıca, önerilen node2vec modelinin aktivite sıralaması öneri performansını değerlendirmek amacıyla, tahminler gerçek ardıllarla karşılaştırılmıştır.

Mevcut veri kümesi için en uygun yapılandırmayı belirlemek üzere, varsayılan p ve q parametreleri, genişlik öncelikli ve derinlik öncelikli arama stratejilerini dengelemek amacıyla 1 olarak ayarlanmış, farklı yerleştirme boyutları (64, 128, 256), yürüyüş uzunlukları (10, 20, 30) ve yürüyüş sayıları (100, 150, 200) üzerinde iterasyonlar gerçekleştirilmiştir. Ek 4'te önerilen node2vec modelinin Python kodunun ayrıntıları sunulmaktadır.

2.2.6. DeepWalk Modelinin Yapılandırılması

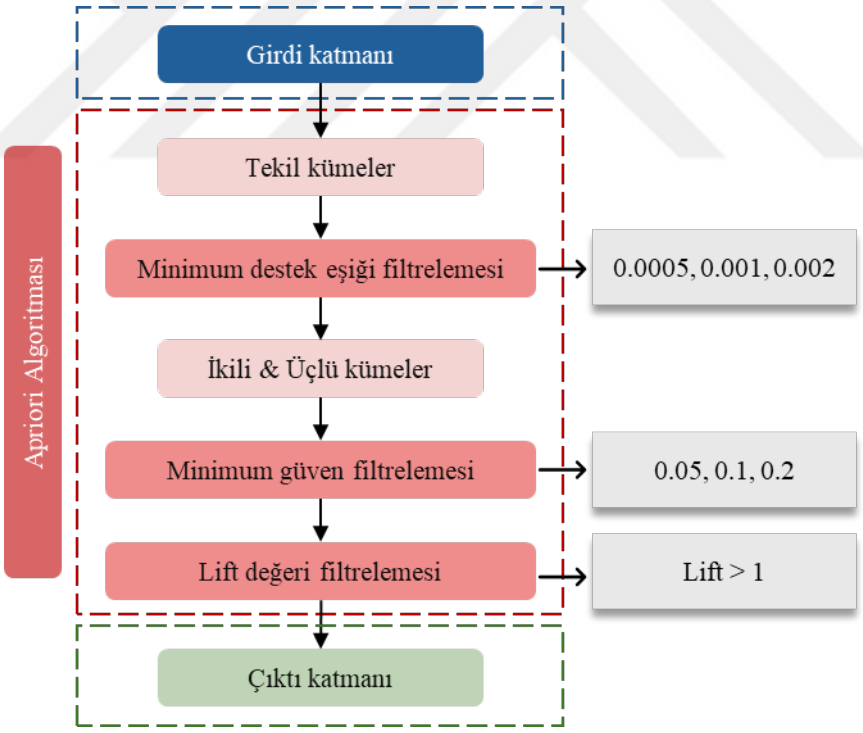
Node2vec'in aksine DeepWalk, her komşunun bir yürüyüş sırasında eşit olasılıkla ziyaret edildiği tekdüze rastgele yürüyüşlere dayanmaktadır (Bandyopadhyay vd., 2018). Özellikle küçük inşaat aktivitesi sıralama veri kümelerinde DeepWalk'in sade yapısı sağlam yerleştirmelere yol açabilmektedir (Kaczmarek, 2023).

DeepWalk, node2vec ile aynı yönlendirilmiş girdi grafını kullanarak eğitilmiştir. Bu graf, benzersiz aktiviteleri temsil eden düğümlere ve bu aktiviteler arasındaki ardıl ilişkilerini gösteren yönlendirilmiş kenarlara sahiptir. DeepWalk modeli, belirlenen yerleştirme boyutu, her bir rastgele yürüyüşte yer alacak düğüm sayısını tanımlayan yürüyüş uzunluğu ve her aktivite düğümü için yürüyüş sayısı parametreleriyle yapılandırılmıştır. Ek olarak, p ve q parametreleri 1'e ayarlanarak rastgele yürüyüşlerin tekdüze olması sağlanmıştır. Node2vec'te olduğu gibi, DeepWalk'ta düğüm yerleştirmelerini elde etmek amacıyla bir skip-gram yaklaşımıyla eğitilmiştir. Bu doğrultuda eğitilmiş DeepWalk modeli, öğrenilen yerleştirmelerin kosinüs benzerliği temelinde, belirli bir aktivitenin en benzer üç aktivite düğümünü tespit etmek için kullanılmıştır. Bu çalışmada tahmin edilen düğümler önerilen ardıllar olarak kabul edilmiştir. Yapılandırılmış DeepWalk modelinin performansı, görünmeyen veriler üzerinde test edilerek değerlendirilmiştir.

DeepWalk, farklı yerleştirme boyutları (64, 128, 256), yürüyüş uzunlukları (10, 20, 30) ve yürüyüş sayıları (100, 150, 200) kullanılarak yinelemeli bir süreçle denenmiştir. Önerilen DeepWalk modeline ilişkin Python kodunun ayrıntıları Ek 5'te yer almaktadır.

2.2.7. ARM Modelinin Yapılandırılması

ARM, etkinlikler arasındaki sıklıkla tekrar eden öğe kümelerini belirlemek ve ilişkileri ortaya çıkarmak için kural tabanlı bir mekanizmayı kullanır (Mou vd., 2020). Bu çalışmada, belirli bir minimum destek (support) eşiğinden daha sık birlikte görülen aktivite kombinasyonlarını tespit etmek amacıyla ARM uygulanmıştır. Bu kapsamda bir öğe kümesi, birlikte gerçekleştiği gözlemlenen inşaat aktiviteleri grubunu ifade etmekte olup, kural keşfinin temel yapı taşıdır. Çalışmada kullanılan ARM modeline ait mimari, Şekil 33'te gösterilmektedir.



Şekil 33. ARM model mimarisi

Kullanılan Apriori algoritması, öncelikle minimum destek eşiğini karşılayan tekil aktiviteleri (tek öğeli kümeleri) belirleyerek başlar ve istatistiksel olarak anlamlı meydana gelme sıklıklarına sahip öğelerin dikkate alınmasını sağlar. Daha sonra bu öğeleri ardışık

biçimde ikili, üçlü gibi daha büyük kümeler hâlinde birleştirerek destek kriterini karşılamayanları eler (Lee vd., 2019). Her bir sık öğe kümesi için olası ilişki kuralları üretilir; örneğin, 'Activity A' ve 'Activity B' öğelerini içeren bir kümeye dayalı olarak algoritma, "Activity A → Activity B" gibi bir kural elde edebilir. Bu kurallar, minimum güven (confidence) eşiği ve lift değerine göre ek olarak filtrelenir. Burada lift değeri 1'in altında olan kurallar elenerek ilişkinin rastlantısal olmasının önüne geçilir.

Planlamada doğru bir sıralama yapabilmek için nadir fakat kritik öneme sahip ilişkileri de yakalamak adına ARM, özellikle düşük bir minimum destek eşiği ve orta düzeyde bir güven eşiği kullanacak şekilde uyarlanmıştır. Böylece ARM, veri kümesi geleneksel DL modellerinin sıralı bağımlılıkları etkili bir şekilde yakalayamayacağı kadar küçük olduğunda dahi, proje aktiviteleri için eyleme dönük ve veri temelli öneriler sunabilen uyarlanabilir ve güvenilir bir alternatif olarak işlev görmektedir.

Çalışmada, ARM'nin farklı parametrelerle tahmin performansının değerlendirilmesi için minimum destek (min_support) değeri 0.0005, 0.001 ve 0.002 olarak alınarak test edilmiş ve kritik olabilecek geçişleri belirlemek amacıyla düşük destek eşiklerine de yer verilmiştir. Ardıl olasılığını ifade eden güven (confidence) için 0.05, 0.1 ve 0.2 değerleri ve rastlantısal olmayan ilişkileri seçmek amacıyla lift>1 koşulu uygulanmıştır. Çok öğeli öncülleri de inceleyebilmek için, maksimum öğe kümesi uzunluğu sınırsız tutulmuştur. Ek 6, önerilen ARM modelinin Python kodu ile ilgili detayları göstermektedir.

2.2.8. ML Modellerinin Değerlendirilmesi

Bu bölümde, iş programlarındaki aktivite sıralaması tahminine yönelik geliştirilen ML modellerinin performanslarının, özel olarak tasarlanan bir performans göstergesi (metrik) ışığında kapsamlı bir şekilde incelenmesi amaçlanmaktadır. Söz konusu inceleme çerçevesi, çalışmada yapılandırılan altı farklı modelin ardıl aktivite tahminine ilişkin öngörü yeteneklerinin ayrıntılı ve çok yönlü bir analizini sunacak şekilde tasarlanmıştır. Geleneksel performans ölçütlerinden (örneğin, kesinlik (precision), F1 skoru vb.) farklı olarak, bu çalışmada yalnızca doğruluk oranı metriğini esas alan bir değerlendirme yaklaşımı benimsenmiştir.

Tipik performans metrikleri, ML modellerinin genel başarı düzeyine ilişkin belirli göstergeler sunmakla birlikte, aktivite-ardıl eşleşmesi gibi spesifik bir bağlamda gerçeğe daha yakın bir performans değerlendirmesi yapmaya her zaman elverişli olmayabilir.

Geleneksel sınıflandırma metrikleri (örneğin; doğruluk (accuracy), kesinlik, F1 skoru) tipik olarak her bir sınıf için doğru ve yanlış tahminlerin (doğru pozitif (True Positive, TP), yanlış pozitif (False Positive, FP), yanlış negatif (False Negative, FN), doğru negatif (True Negative, TN)) belirlenebilmesine dayanmaktadır. Ancak tez kapsamında gerçekleştirilen aktivite ardıl tahmin problemi, doğası gereği klasik bir sınıflandırma problemi değildir. Aktivite ardıl tahmininde her tahminin kendine özgü (doğru veya yanlış) bir eşleşmesi olduğundan, TN veya FP gibi kavramlar modellere tanımlanamamakta ve dolayısıyla bu metriklerin kullanılması doğru olmamaktadır. Bu nedenle, çalışmada doğruluk oranı metriği kullanılması tercih edilmiştir. Doğruluk oranı, toplam sınıflandırılmış örnekler üzerinden doğru sınıflandırılan örneklerin yüzdesini ifade etmektedir. Literatürde de benzer bir yaklaşımın yer aldığı, özellikle Gupta vd. (2022) tarafından yürütülen çalışmada bu metriğin, model parametrelerinin optimizasyonu amacıyla kullanıldığı görülmektedir. Bu doğrultuda, hem ML modellerinin en uygun parametre değerlerinin elde edilmesi hem de bu modellerin birbirleriyle karşılaştırılması aşamalarında Denklem (1)'de verilen "Doğruluk Oranı" metriğinden yararlanılmıştır:

$$Doğruluk\ Oranı = \frac{Doğru\ Tahmin\ Sayısı}{Toplam\ Tahmin\ Sayısı} \times 100 \quad (1)$$

Burada doğru tahmin sayısı, modellerin test projelerindeki aktiviteler için öngördüğü ardıllardan kaçının gerçekte doğru eşleşme olduğunu; toplam tahmin sayısı ise ilgili modeller tarafından tahmin edilen tüm aktivite-ardıl satırlarının toplamını temsil etmektedir. Bu yaklaşım, özellikle inşaat projelerinin yapısal karakteristiklerini dikkate alarak, ML modellerinin etkinliğini daha hedef odaklı ve güvenilir bir biçimde değerlendirmeye imkân tanıyacaktadır.

2.2.9. Aktivite Sıralamasının Dış Ortama Aktarılması

İş programlarındaki aktivite sıralamasının ML modelleriyle elde edilmesi sürecinin son aşaması, en yüksek doğruluk oranına sahip modelin kullanılmasıyla oluşturulan aktivite sıralamasının dış ortama aktarılmasıdır. Bu amaçla, farklı parametre kombinasyonlarıyla eğitilip test edilen modeller arasından en iyi performans sergileyen ML modeli belirlendikten sonra, ilgili aktiviteler ve ardıl ilişkileri ".csv" formatında dışa

aktarılır. Aktarım işlemi sırasında, modelin son aşamada elde edeceği iş programının gereksiz aktivitelerden arındırılması ve BIM modeline sorunsuz entegrasyonun sağlanması amacıyla, kullanıcıdan "Başlangıç Aktivitesi", "Bodrum Kat Sayısı" ve "Normal Kat Sayısı" bilgilerini girmesi/bildirmesi istenmektedir. Bu bilgiler doğrultusunda, aktivite sıralaması güncellenerek nihai iş programının oluşturulması kolaylaştırılmaktadır.

Başlangıç aktivitesi olarak seçilen aktivite sıralamada ilk satıra yerleştirilir ve bu aktivitenin ardılı, bir sonraki satırda gösterilir. Ardıl aktivite yeniden işlenerek kendi ardılı alt satıra eklenir ve bu süreç, ardılı bulunmayan bir aktiviteye ulaşılana dek tekrarlanır. Bodrum kat sayısı ve normal kat sayısı bilgileri ise BIM modelinde bulunmayan aktivitelerin ve bunların ardıllarının gereksiz yere tahmin edilmesini engelleyerek iş programında daha doğru bir sıralama elde edilmesine katkı sağlar. Böylece, test projelerindeki kat sayısının BIM modelinde tanımlı olandan fazla olması durumunda, yanlış aktivitelerin model elemanlarıyla eşleştirilmesinin önüne geçilmektedir. Bunun yanı sıra, test projelerinde yer alan ancak ML modellerinin ardıl tahmin edemediği aktiviteler de "Tahmin Edilemeyen Aktiviteler" başlığı altında aynı aktivite sıralaması dosyasında listelenerek gözden kaçma ihtimalinin ortadan kalkması sağlanır. Bu sayede, BIM modelinin kat bilgileriyle tutarlı bir iş programı taslağı elde edilmektedir. Kullanıcı, ihtiyaç duyduğu takdirde oluşturulan ".csv" dosyasında aktivite sıralamasına ilişkin değişiklikleri uygulayarak projeye özgü parametrelerin esnek biçimde gözetilebilmektedir. Bu aşama için geliştirilen Python kodu, Ek 7'de gösterilmektedir.

2.3. 3B BIM Modelinin Oluşturulması

Tez çalışması kapsamında, BIM tabanlı 3B bina modeli üzerinden otomatik inşaat iş programı üretilmesini sağlayan Dynamo kodları ve bir Revit eklentisi geliştirilmiştir. Geliştirilen bu sistemin doğru ve eksiksiz çalışabilmesi için, modelin Autodesk Revit 2024 üzerinde belirli tasarım kurallarına uygun şekilde oluşturulması büyük önem taşımaktadır. Dynamo kodları ve Revit eklentisi, modeldeki yapı elemanlarına yeni parametreler tanımlayarak, yeni ve mevcut parametrelerle birlikte iş programını otomatik olarak oluşturmakta ve güncellemektedir. Bu sürecin sağlıklı işleyebilmesi adına, modelleme aşamasından itibaren bazı kısıtlamalara uyulması gerekmektedir.

Bu kısıtlamalar üç başlıkta toplanabilir: İş programı oluşturulurken dikkate alınacak yapı elemanları, bu yapı elemanlarının doğru şekilde isimlendirilmesi ve katların tutarlı

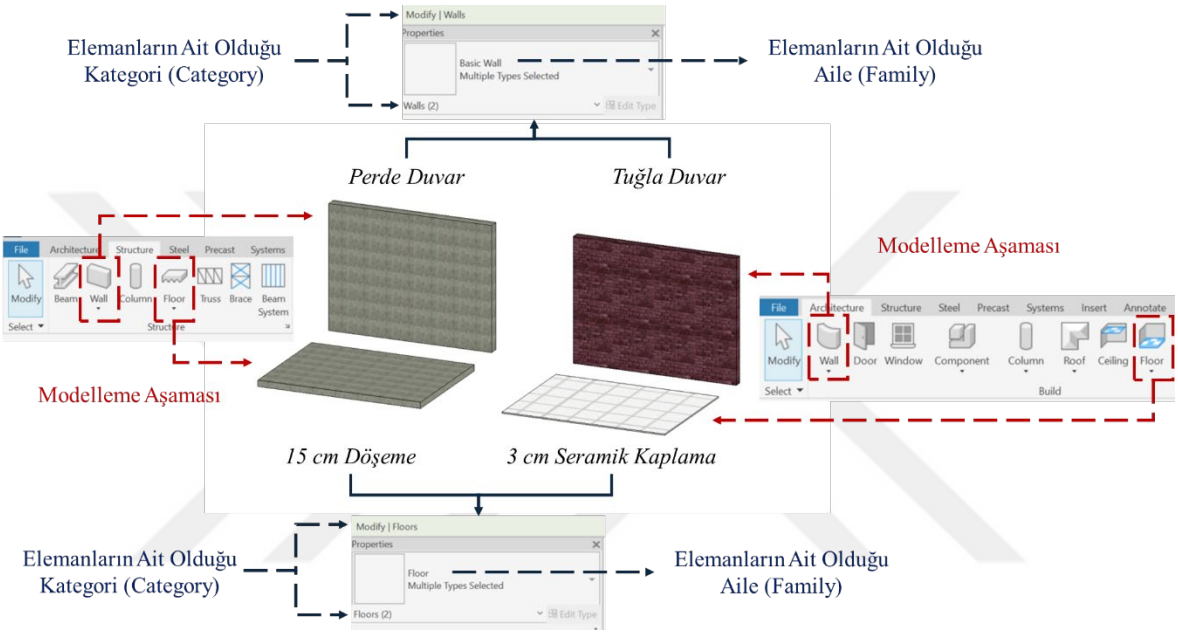
biçimde adlandırılması. Söz konusu kısıtlamalar, hem geliştirilen otomasyon kodunun sorunsuz şekilde çalışmasını hem de modelde yer alan verilerin doğru biçimde işlenmesi açısından kritik bir öneme sahiptir.

1. Dikkate Alınan Yapı Elemanları: Tez çalışmasında, yalnızca mimari ve statik yapı elemanları dikkate alınmakta olup, elektrik ve mekanik elemanlar ile belirtilmeyen diğer mimari ya da yapısal elemanlar kapsam dışı bırakılmıştır. Dikkate alınan yapısal elemanlar; grobeton, temel, kolon, perde duvar, kiriş, döşeme ve merdivenden oluşmaktadır. Mimari elemanlar ise duvar, sıva, boya, duvar kaplama, seramik (duvar), şap, yalıtım, laminant, seramik (döşeme), zemin kaplama, asma tavan, asma tavan kaplama, kapı, pencere ve çatı gibi elemanları içermektedir. Ayrıca, geliştirilen eklenti, Revit ortamında modellenmeyen kalıp imalatı aktivitesini oluşturmanın yanı sıra, Revit'te modellenebilir olmasına rağmen modellenmemiş durumdaki donatı imalatı aktivitesini de üretmektedir.

Bu sınırlamanın temel nedeni, inşaat iş programının hazırlanmasında kullanılacak bilgi setini netleştirmek ve gereksiz verilerin yaratabileceği karmaşıklığın önüne geçmektir. Örneğin, elektrik veya mekanik sistemlere ait elemanlar, bu çalışmada yer alan iş programının kapsamına doğrudan dâhil edilmediğinden, otomasyon kodu bu elemanlara ilişkin veri işleme sürecini gerçekleştirmemektedir. Böylece, modelde sadece ilgili yapı elemanları üzerinden daha tutarlı ve sade bir veri akışı sağlanmaktadır.

2. Yapı Elemanlarının Doğru İsimlendirilmesi: Autodesk Revit yazılımında her eleman türü için spesifik bir kategori bulunmadığı için, benzer ya da farklı elemanların aynı kategori altında modellenmesi gerekebilmektedir. Örneğin, grobeton ve temel elemanları “Structural Foundations” kategorisi altında, şap ve laminant gibi elemanlar “Floors” kategorisi altında bulunmaktadır. Ayrıca, Şekil 34’te de gösterildiği üzere, modelleme aşamasında yapısal döşeme elemanı “Structure > Floor” ve mimari döşeme kaplama elemanları “Architecture > Floor” sekmelerinden seçilerek modellenmelerine rağmen, tüm elemanlar “Floors” kategorisi altında yer almaktadır. Benzer şekilde perde duvar elemanı “Structure > Walls” sekmesinden modellenmesine rağmen, mimari olarak modellenen (Architecture > Wall) tuğla duvar, seramik, boya gibi elemanlarla “Walls” kategorisinde yer almaktadır. Bu durum, kategorilerde yer alan elemanların seçimini zorlaştırmakta ve bu nedenle parametre tanımlama ve veri işleme süreçlerinde hatalara yol açabilmektedir. Dolayısıyla, kategorilerde yer alan elemanlarla ilgili yanlış algılama yapmaması için, çalışma kapsamında belirlenen elemanlara özgün bir isimlendirme

stratejisi geliştirilmiştir. Tablo 3'te, çalışma kapsamında incelenen mimari ve yapısal elemanlar ile elemanların nasıl isimlendirilmesi gerektiği detaylı biçimde açıklanmaktadır. Bu tabloda belirtilen kurallara uyulması, geliştirilen eklentinin elemanları doğru algılayıp işlemesi için büyük önem taşımaktadır. Öte yandan, kolon ve kiriş gibi bazı elemanlarda özel bir isimlendirme gerekliliği bulunmamaktadır ve bu durum Tablo 3'te ayrıca belirtmektedir.



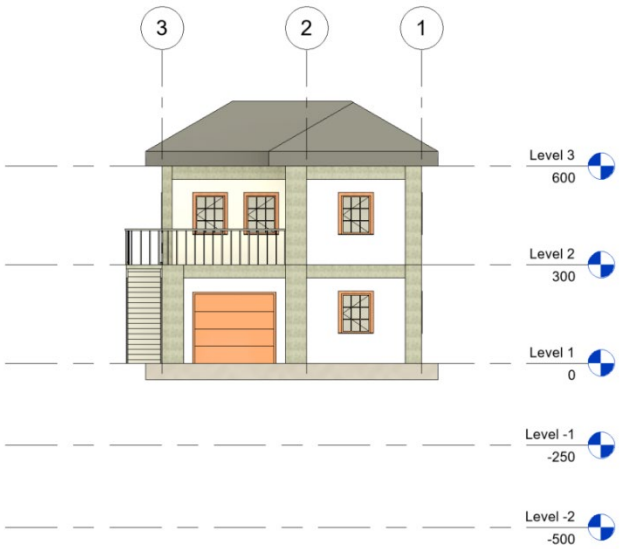
Şekil 34. Revit örnek modelleme aşaması ve elemanların ait olduğu kategori

Tablo 3. Çalışma kapsamında incelenen elemanlar ve isimlendirme kuralı

Eleman İsmi (Type Name)	**Ait Olduğu Kategori (Category)**	**İsimlendirme Kuralı**
Grobeton	Structural Foundations	"Grobeton" kelimesini içermelidir.
Temel	Structural Foundations	Bir kural bulunmamaktadır.
Kolon	Structural Columns	Bir kural bulunmamaktadır.
Perde Duvar	Walls	"Perde" kelimesini içermelidir.
Kiriş	Structural Framing	Bir kural bulunmamaktadır.
Döşeme	Floors	Bir kural bulunmamaktadır.
Merdiven	Stairs	Bir kural bulunmamaktadır.
Duvar	Walls	Bir kural bulunmamaktadır.
Sıva	Walls	"Sıva" kelimesini içermelidir.
Boya	Walls	"Boya" kelimesini içermelidir.
Duvar Kaplama	Walls	"Kaplama" kelimesini içermelidir.
Seramik (Duvar)	Walls	"Seramik" kelimesini içermelidir.
Şap	Floors	"Şap" kelimesini içermelidir.
Yalıtım	Floors	"Yalıtım" kelimesini içermelidir.
Laminant	Floors	"Laminant" kelimesini içermelidir.
Seramik (Döşeme)	Floors	"Seramik" kelimesini içermelidir.
Zemin Kaplama	Floors	"Kaplama" kelimesini içermelidir.
Asma Tavan	Ceilings	Bir kural bulunmamaktadır.
Asma Tavan Kaplama	Ceilings	"Kaplama" kelimesini içermelidir.
Kapı	Doors	Bir kural bulunmamaktadır.
Pencere	Windows	Bir kural bulunmamaktadır.
Çatı	Roofs	Bir kural bulunmamaktadır.

3. Katların Tutarlı Biçimde Adlandırılması: Bir diğer kritik kısıtlama ise katların tutarlı biçimde isimlendirilmesidir. Bu çalışmada, Revit'in varsayılan kat isimlendirme düzeni olan "Level 1, Level 2, …" düzeni kullanılmıştır. Bodrum katlar için "Level -1, Level -2, …" gibi sıfırın altında indeks numaraları tercih edilmiş, Şekil 35'te gösterildiği gibi zemin kat (temel üst kot) ise "Level 1" olarak tanımlanmıştır. Bu yaklaşım, modeldeki kat düzeylerini anlaşılır kılmakta ve iş programında kat bazlı planlamalar yapılırken tutarlı veri akışını desteklemektedir. Özellikle, ML tabanlı olarak tahmin edilen aktiviteler ile BIM modelinden elde edilen aktivitelerin tutarlı biçimde eşleştirilebilmesi için katların isimlendirme sistematiğinin uyumlu olması gerekmektedir. Eğer kat adları düzensiz veya

çelişkili şekilde verilirse, oluşturulan aktivitelerin adlandırma süreci karışıklığa uğrayabilmekte; bu da iş programının doğru oluşturulmasını olumsuz etkilemektedir.



Şekil 35. Örnek bir kat isimlendirmesi

Yukarıda açıklanan üç temel kısıtlama (dikkate alınan yapı elemanlarının belirlenmesi, yapı elemanlarının isimlendirilmesi ve katların tutarlı adlandırılması), tez boyunca kullanılan BIM modellemesinin güvenilir bir veri kaynağı olarak işlev görmesi için hayati öneme sahiptir. Geliştirilen eklenti ve iş programı oluşturma yaklaşımı, bu kısıtlamalara titizlikle uyulduğu takdirde, hatasız ve etkin bir şekilde çalışmaktadır. Dolayısıyla, tezde sunulan 3B BIM modeli oluşturma sürecinde bu noktalara dikkat edilmesi hem araştırmanın metodolojik bütünlüğünü hem de elde edilen sonuçların geçerliliğini artıracaktır.

2.4. Dynamo Kodları

Bu bölümde, tez çalışmasının ana hedefi doğrultusunda hazırlanan dokuz adet Dynamo kodu ve birbirleriyle etkileşimi ayrıntılı olarak sunulmaktadır. Söz konusu kodlar, bir önceki bölümde aktarılan kısıtlar ve tasarım ilkeleri esas alınarak oluşturulan 3B BIM modelinden yararlanmak suretiyle iş programının oluşturulmasını sağlamaktadır. Aşağıda, sırasıyla uygulanması gereken kodların başlıkları yer almaktadır. Sonraki aşamalarda, her bir kod ayrıntılı olarak görseller eşliğinde ele alınmaktadır:

1. Proje Parametrelerinin Oluşturulması
2. Varsayılan Değerler
3. Kalıp Metrajının ve Merdiven Beton Hacminin Hesaplanması
4. Kullanıcı Arayüzü ile Varsayılan Değerlerin Kontrolü
5. Aktivite Sürelerinin Hesaplanması
6. Metraj Listelerinin Oluşturulması
7. Aktivite Adlarının Oluşturulması
8. Navisworks Search Sets Dosyasının Oluşturulması
9. ML Model Çıktısı ile İlişkilendirme ve İş Programının Elde Edilmesi

Dynamo kodlarının sıralı biçimde ve birbirini tamamlayacak şekilde kullanılması, BIM tabanlı proje yönetimi yaklaşımının verimliliğini artırmakta ve iş programının oluşturulması sürecinde hataları asgari düzeye indirgemektedir.

2.4.1. Proje Parametrelerinin Oluşturulması

Bu aşamada, metraj listelerinin oluşturulması, adam-saat değerlerinin ve aktivite isimlerinin elemanlara aktarılması ile iş programının oluşturulması için Revit ortamında bulunmayan ve eleman düzeyinde tanımlı olmayan toplam 48 yeni parametre oluşturulmuştur. Revit'te oluşturulan parametrelerin özellikleri incelendiğinde, tez kapsamında metraj listelerinin oluşturulması ve kullanıcı tanımlı yeni parametreler oluşturulması amacıyla "Proje Parametreleri (Project Parameters)" kullanılmış ve kullanım amacına göre "Instance" veya "Type" parametresi olarak yapılandırılmıştır. Tablo 4'te, bu yeni parametrelere ilişkin isim, uygulama türü, hangi kategorilerdeki elemanlar için tanımlandığı, parametre türü ve ilgili gruplar sunulmaktadır. Parametre kategorisi sütununda, oluşturulan parametrenin hangi kategorideki elemanlara uygulanacağı belirtilmekte olup, parantez içinde verilen ifade, yalnızca bu isimde bir eleman modeli mevcutsa parametrenin tanımlandığını göstermektedir. Örneğin, "Grobeton Adamsaat" parametresi, BIM modelinde "Grobeton" adlı bir eleman bulunması hâlinde "Structural Foundation" kategorisindeki elemanlar için oluşturulmaktadır ve söz konusu eleman yoksa parametre oluşturulması gerçekleşmemektedir. Bu durum, aşağıda verilen Dynamo kodu açıklamalarında da ayrıca vurgulanmaktadır. Parametre türü sütunu, ilgili parametreye girilecek verinin metin (Text), sayı (Number), uzunluk (Length), açı (Angle), alan (Area), Evet/Hayır (Yes/No) gibi türlerden hangisi olacağını ifade etmektedir. Çalışma

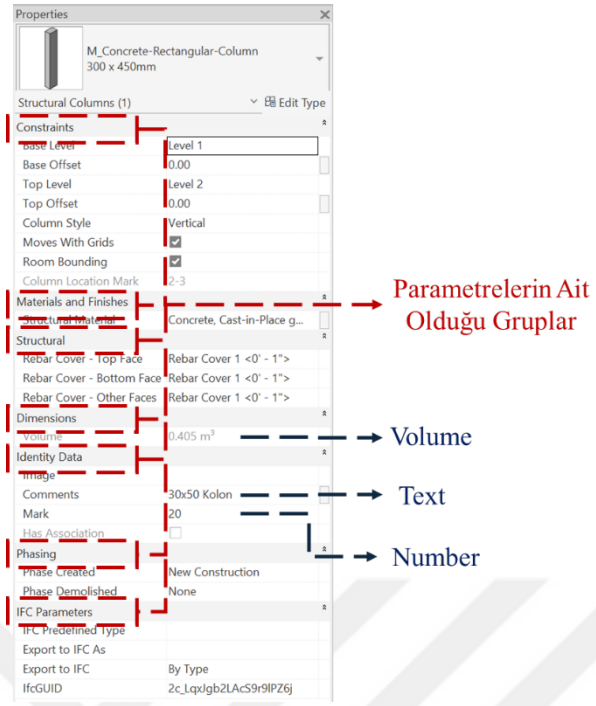
kapsamında oluşturulan parametrelerin, metin (Text) ve sayı (Number) türünde değerler alması öngörülmüştür. Parametrenin ait olduğu grup sütunu ise, eklenen parametrenin BIM modelinde bir eleman seçildiğinde hangi başlık altında görüntüleneceğini göstermektedir. Yeni bir başlık oluşturulması ve tüm parametrelerin bir arada görülmesi açısından "General" başlığı seçilmiştir. Şekil 36'da farklı parametre türleri ve grupları, örnek bir Revit modeli üzerinden gösterilmektedir.

Tablo 4. Tanımlanan proje parametreleri ve temel özellikleri

Parametre İsmi	**Parametre Uygulama Türü**	**Parametre Kategorisi**	**Parametre Türü**	**Parametrenin Ait Olduğu Grup**
Aktivite Adı	Instance	Tüm Kategoriler	Text	General
Aktivite Adı-Kalıp	Instance	Str. Foundations, Str. Column, Str. Framing, Floors, Walls, Stairs	Text	General
Aktivite Adı-Demir	Instance	Str. Foundations, Str. Column, Str. Framing, Floors, Walls, Stairs	Text	General
Günlük Çalışma Süresi	Type	Tüm Kategoriler	Number	General
Ekip Sayısı	Type	Tüm Kategoriler	Number	General
Donatı Metrajı	Instance	Str. Foundations	Number	General
Toplam Kalıp Metrajı	Instance	Str. Foundations	Number	General
Kalıp Metrajı	Instance	Generic Models	Number	General
Donatı İmalat Süresi	Instance	Str. Foundations, Floors	Number	General
Kalıp İmalat Süresi	Instance	Str. Foundations, Floors	Number	General
Toplam Kalıp İmalat Süresi	Instance	Str. Foundations	Number	General
Beton Döküm Süresi	Instance	Str. Foundations, Str. Column, Str. Framing, Floors, Walls, Stairs	Number	General
Volume	Instance	Stairs	Number	General
Beton Adamsaat	Type	Str. Foundations, Str. Column, Str. Framing, Floors, Walls, Stairs	Number	General
Kalıp Adamsaat	Type	Str. Foundations, Str. Column, Str. Framing, Floors, Walls, Stairs	Number	General
Donatı Adamsaat	Type	Str. Foundations, Str. Column, Str. Framing, Floors, Walls, Stairs	Number	General
Grobeton Adamsaat	Type	Str. Foundations (Grobeton)	Number	General
Yalıtım Adamsaat	Type	Floors (Yalıtım)	Number	General
Şap Adamsaat	Type	Floors (Şap)	Number	General

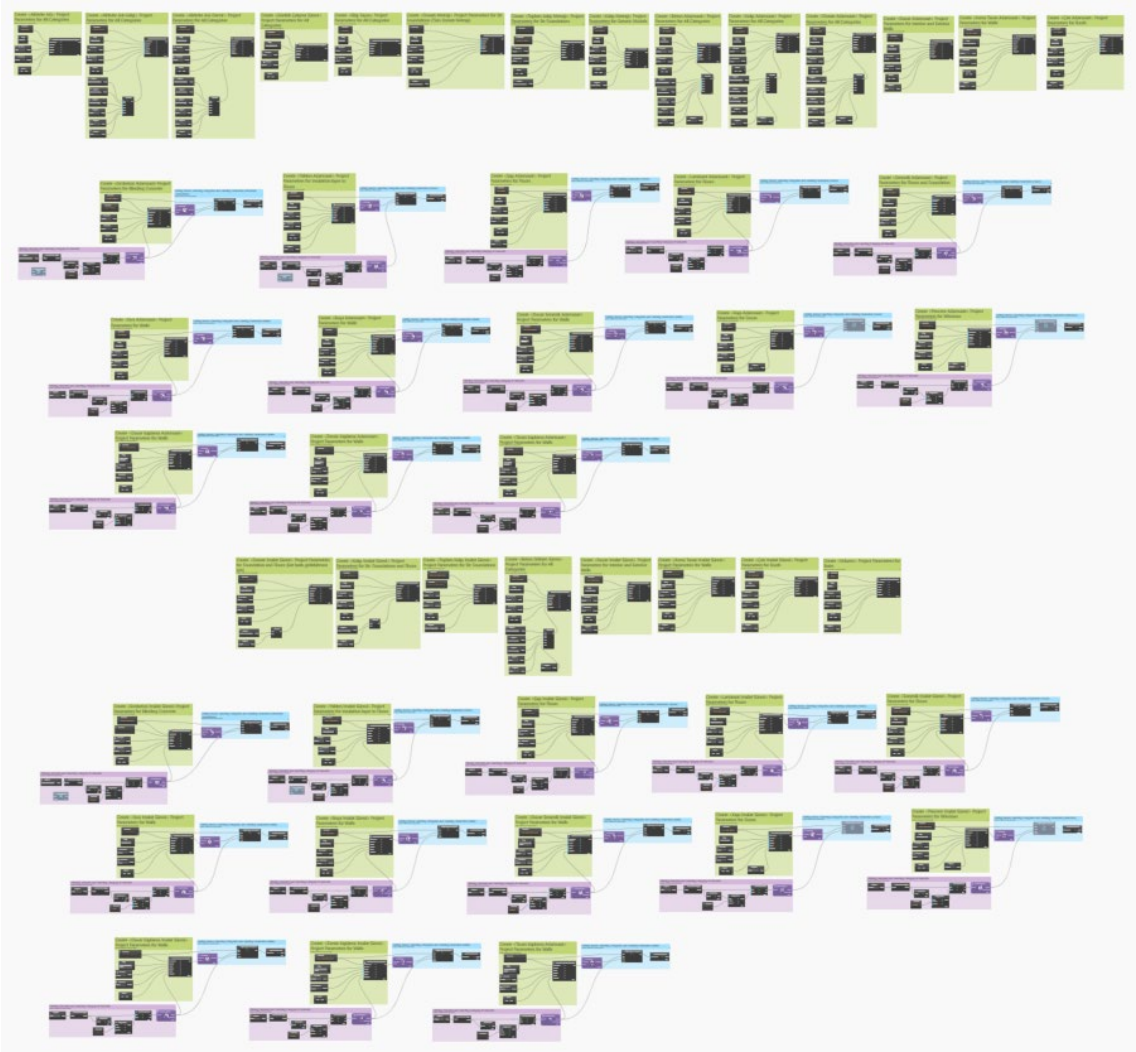
Tablo 4'ün devamı

Laminant Adamsaat	Type	Floors (Laminant)	Number	General
Seramik Adamsaat	Type	Floors (Seramik)	Number	General
Zemin Kaplama Adamsaat	Type	Floors (Kaplama)	Number	General
Duvar Adamsaat	Type	Walls	Number	General
Sıva Adamsaat	Type	Walls (Sıva)	Number	General
Boya Adamsaat	Type	Walls (Boya)	Number	General
Duvar Seramik Adamsaat	Type	Walls (Seramik)	Number	General
Duvar Kaplama Adamsaat	Type	Walls (Kaplama)	Number	General
Asma Tavan Adamsaat	Type	Ceilings	Number	General
Tavan Kaplama Adamsaat	Type	Ceilings (Kaplama)	Number	General
Kapı Adamsaat	Type	Doors	Number	General
Pencere Adamsaat	Type	Windows	Number	General
Çatı Adamsaat	Type	Roofs	Number	General
Grobeton İmalat Süresi	Instance	Str. Foundations (Grobeton)	Number	General
Yalıtım İmalat Süresi	Instance	Floors (Yalıtım)	Number	General
Şap İmalat Süresi	Instance	Floors (Şap)	Number	General
Laminant İmalat Süresi	Instance	Floors (Laminant)	Number	General
Seramik İmalat Süresi	Instance	Floors (Seramik)	Number	General
Zemin Kaplama İmalat Süresi	Instance	Floors (Kaplama)	Number	General
Duvar İmalat Süresi	Instance	Walls	Number	General
Sıva İmalat Süresi	Instance	Walls (Sıva)	Number	General
Boya İmalat Süresi	Instance	Walls (Boya)	Number	General
Duvar Seramik İmalat Süresi	Instance	Walls (Seramik)	Number	General
Duvar Kaplama İmalat Süresi	Instance	Walls (Kaplama)	Number	General
Asma Tavan İmalat Süresi	Instance	Ceilings	Number	General
Tavan Kaplama İmalat Süresi	Instance	Ceilings (Kaplama)	Number	General
Kapı İmalat Süresi	Instance	Doors	Number	General
Pencere İmalat Süresi	Instance	Windows	Number	General
Çatı İmalat Süresi	Instance	Roofs	Number	General

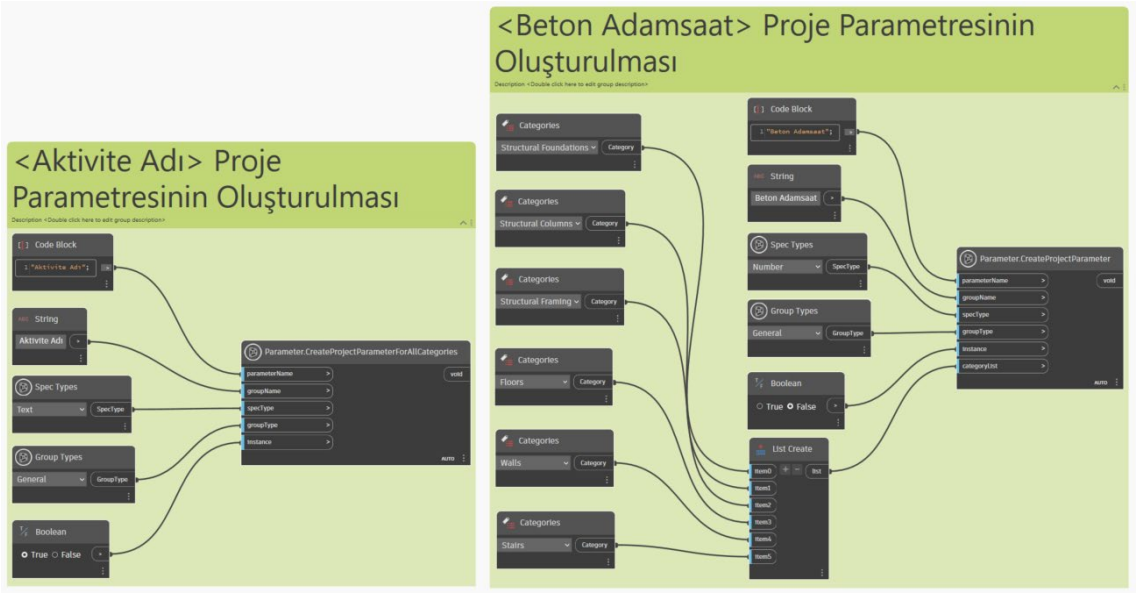


Şekil 36. Revit üzerinde örnek parametre türleri ve grupları

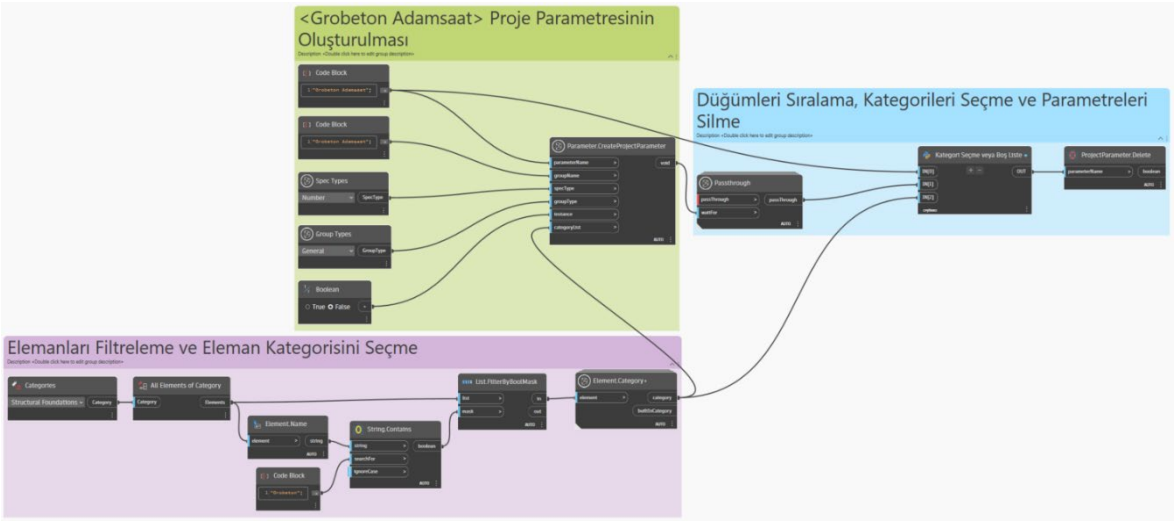
Tablo 4'te tanımlanan 48 parametrenin ilgili isim ve özelliklerle manuel olarak oluşturulması hem zaman açısından maliyetli olmakta hem de hata riskini artırmaktadır. Bu nedenle, tez kapsamında bu işlem Dynamo kodu aracılığıyla otomatikleştirilmiştir. Şekil 37'de, söz konusu amaç doğrultusunda hazırlanan Dynamo kodunun genel görünümü verilmektedir. Şekil 38'de ise birden çok kategoriye aynı anda parametre tanımlama, farklı parametre uygulama türleri, parametre türleri ve ait olduğu gruplara ilişkin örnekler yer almaktadır. Belirli bir elemanın varlığına bağlı olarak parametrelerin oluşturulup oluşturulmayacağına dair kod bölümü ise Şekil 39'da gösterilmiştir. Bu süreçte, parametrelerin kategori bazında tanımlanması nedeniyle, ilk aşamada parametre oluşturulmakta, ardından ilgili elemanın bulunup bulunmadığı filtrelenerek, eleman yoksa tanımlanan parametre silinmekte; eleman mevcutsa parametre geçerli kalmaktadır. Ayrıca Dynamo kodu içerisinde eklenen Python kodu sayesinde, kodun hacminin aşırı büyümesi ve çalışmasının yavaşlaması engellenmiştir. Grobeton elemanına ilişkin örnek, Şekil 39'da ayrıntılı biçimde sunulmaktadır.



Şekil 37. Proje parametrelerinin oluşturulması Dynamo kodunun genel görünümü



Şekil 38. Örnek parametre tanımlama Dynamo kodları



Şekil 39. Eleman varlığına göre parametrelerin oluşturulması kodu

2.4.2. Varsayılan Değerler

Bu aşamada, daha önce tanımlanan ilgili parametrelere varsayılan değerlerin (adam-saat, günlük çalışma süresi, ekip sayısı ve yaklaşık donatı metrajı) atanması gerçekleştirilmektedir. Söz konusu değerler, iş programı hazırlanırken aktivite sürelerinin belirlenmesi açısından kritik öneme sahiptir. Adam-saat verileri, Çevre, Şehircilik ve İklim Değişikliği Bakanlığının (ÇŞİDB) Birim Fiyat kitabında yer alan poz analizindeki saat ifadelerinden yararlanılarak elde edilmiştir. Yaklaşık donatı metrajının belirlenmesinde ise, inşaat sektöründe de sıklıkla tercih edilen m^2 başına 34 kg donatı kullanıldığı varsayımı kullanılmıştır (URL-7, 2024). Bu hesabın ayrıntılarına, Dynamo kodu açıklamalarında daha detaylı şekilde yer verilmiştir. Çalışma kapsamında belirlenen değerler, değerlerin atandığı elemanlar, parametre isimleri ve dikkate alınan ilgili poz numaraları Tablo 5'te özetlenmiştir.

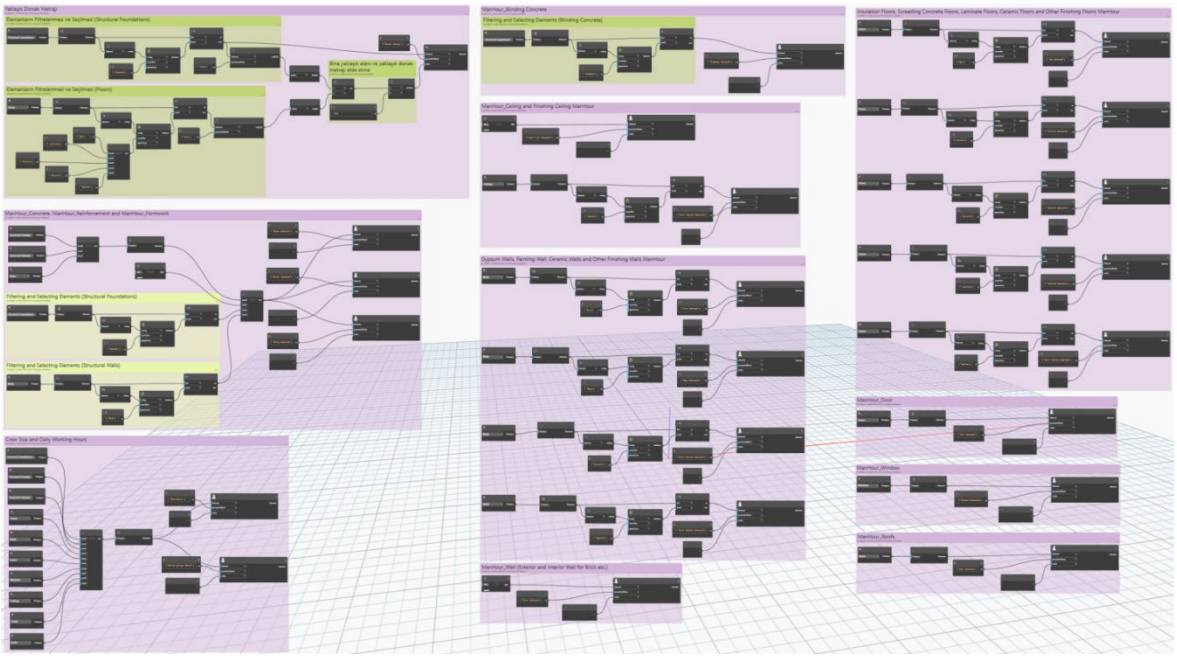
Tablo 5. Varsayılan adam-saat, günlük çalışma süresi, ekip sayısı değerleri ve yaklaşık donatı metrajı

Parametre İsmi	İlgili Elemanlar	Adam-saat	Poz No
Beton Adamsaat	Str. Foundations (Grobeton hariç), Str. Columns, Str. Framing, Stairs, Floors (Yalıtım, Şap, Laminant, Seramik, Kaplama hariç), Walls (Perde)	0.450 sa/m^3	15.150.1006
Kalıp Adamsaat		2.000 sa/ m^2	15.180.1002
Donatı Adamsaat		42.000 sa/kg	15.160.1004
Grobeton Adamsaat	Str. Foundations (Grobeton)	0.450 sa/m^3	15.150.1003
Yalıtım Adamsaat	Floors (Yalıtım)	1.050 sa/m^2	15.270.1005
Şap Adamsaat	Floors (Şap)	1.250 sa/m^2	15.250.1101
Laminant Adamsaat	Floors (Laminant)	0.450 sa/m^2	15.490.1003
Seramik Adamsaat	Floors (Seramik)	1.100 sa/m^2	15.375.1054
Zemin Kaplama Adamsaat	Floors (Kaplama)	1.250 sa/m^2	15.250.1101
Duvar Adamsaat	Walls (Sıva, Boya, Seramik, Kaplama, Perde hariç)	1.890 sa/m^2	15.220.1004
Sıva Adamsaat	Walls (Sıva)	1.900 sa/m^2	15.280.1009
Boya Adamsaat	Walls (Boya)	0.600 sa/m^2	15.540.1265
Duvar Seramik Adamsaat	Walls (Seramik)	1.200 sa/m^2	15.380.1056
Duvar Kaplama Adamsaat	Walls (Kaplama)	1.890 sa/m^2	15.220.1004
Asma Tavan Adamsaat	Ceilings (Kaplama hariç)	3.000 sa/m^2	15.530.1905
Tavan Kaplama Adamsaat	Ceilings (Kaplama)	3.000 sa/m^2	15.530.1905
Kapı Adamsaat	Doors	2.000 sa/m^2	15.510.1001
Pencere Adamsaat	Windows	1.050 sa/m^2	15.470.1012
Çatı Adamsaat	Roofs	2.000 sa/m^2	15.300.1001

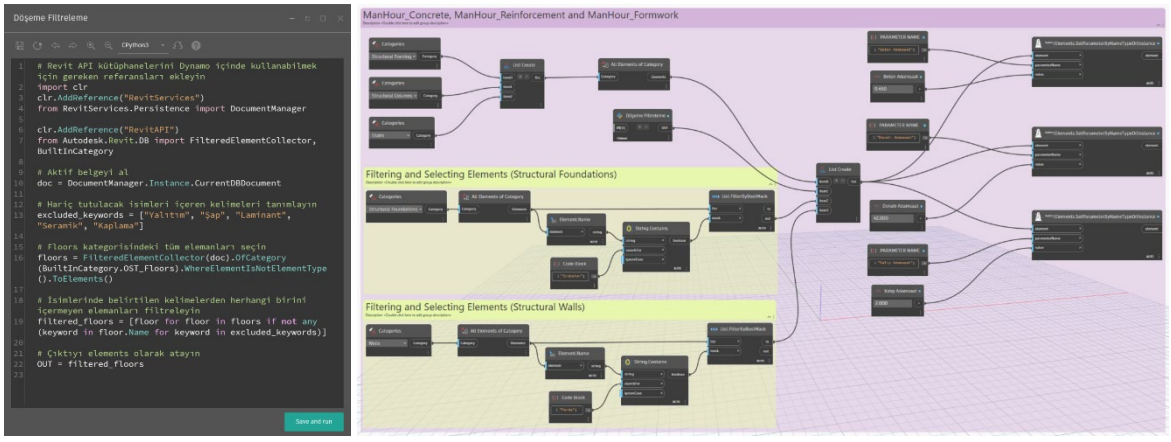
Parametre İsmi	İlgili Elemanlar	Tanımlanan Varsayılan Değer
Donatı Metrajı	Str. Foundations	m^2 başına 34 kg
Günlük Çalışma Süresi	Tüm Kategoriler	10 sa
Ekip Sayısı	Tüm Kategoriler	4 işçi

Şekil 40'da, bu aşama için oluşturulan Dynamo kodunun genel görünümü sunulmaktadır. Parametre değerlerini elemanlara atama sürecinde "Element.SetParameterByName" ve "Element.SetParameterByNameTypeorInstance" düğümlerinden yararlanılmıştır. İki farklı düğümün tercih edilmesinin nedeni,

“Element.SetParameterByName” düğümünün “Type Parametrelerine” değer atayamamasıdır. Grobeton, laminant veya şap gibi belirli eleman tiplerine değer atamak amacıyla, önceki aşamada olduğu gibi filtreleme yöntemleri kullanılmıştır. Ayrıca, bazı kategorilerde çok sayıda elemanın aynı anda hariç tutulması gerektiğinde işlem kolaylığı sağlamak amacıyla Python kodları da kullanılmıştır. Şekil 41’de, bu iki işlemin uygulanmasına dair örnek kod parçacıkları gösterilmektedir.



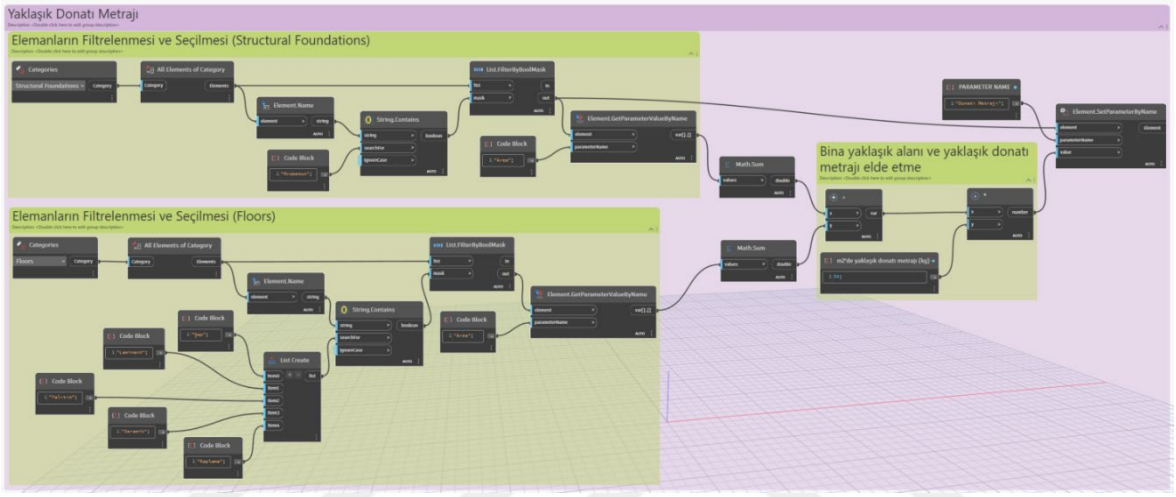
Şekil 40. Varsayılan değerlerin elemanlara atanması Dynamo kodunun genel görünümü



(a) Python kodu (b) Dynamo düğümleri

Şekil 41. (a) Python kodu ve (b) Dynamo düğümleri kullanılarak eleman filtreleme

Yaklaşık donatı metrajı hesabında, bina toplam inşa alanı dikkate alınmıştır. Bu alanı elde etmek için, bina temel alanı ve döşeme alanları Dynamo düğümleri yardımıyla toplanmış ve elde edilen toplam alan, m^2 başına 34 kg donatı varsayımıyla çarpılarak kg bazında yaklaşık donatı metrajı hesaplanmıştır. Elde edilen sonuç, temel elemanları için oluşturulan "Donatı Metrajı" parametresine atanmıştır. Şekil 42'de, söz konusu hesaplama ve buna ilişkin Dynamo kodu ayrıntılı biçimde gösterilmektedir.



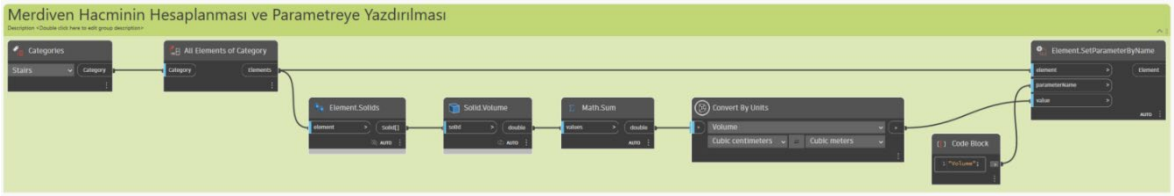
Şekil 42. Dynamo düğümleri ile yaklaşık donatı metrajı

2.4.3. Kalıp Metrajının ve Merdiven Beton Hacminin Hesaplanması

Bu aşamada, 3B BIM modelinde kalıp metrajının ve merdiven beton hacminin hesaplanmasına yönelik ilave işlemler gerçekleştirilmektedir. İş programlarında kolon, kiriş, duvar gibi modellenebilen elemanlara ait aktivitelerle birlikte, kalıp gibi Revit'te modellenmeyen aktivitelerde bulunmaktadır. Revit içindeki mevcut parametrik veriler sayesinde pek çok elemanın metraj listesi hazırlanabilmekte ve sonraki aşamalarda detaylandırılacak yöntemlerle metrajlar hesaplanabilmektedir. Ancak, özellikle kalıp metrajı ve merdiven hacmi (volume) hususunda Revit'te doğrudan kullanılabilecek hazır bir parametre bulunmamaktadır. Bu nedenle, modeldeki tüm elemanlara ilişkin kapsamlı bir metraj analizinin yapılabilmesi amacıyla söz konusu değerlere özel hesaplamalar geliştirilmiştir. Bu sayede kalıp ve merdiven beton metrajları elde edilmiştir.

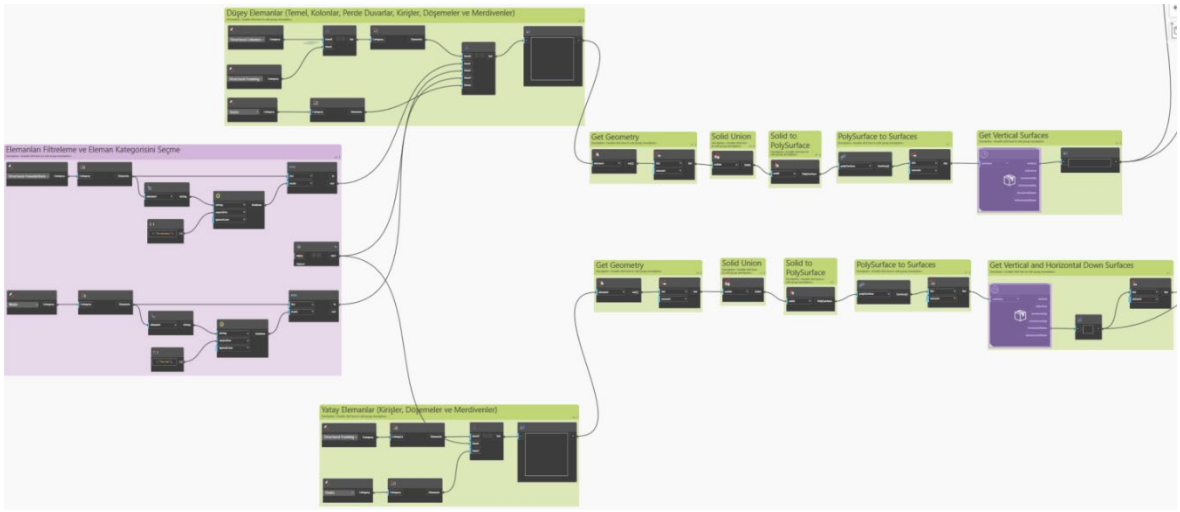
Merdiven beton hacminin hesaplanmasında, "Parametrelerin Tanımlanması" aşamasında merdiven elemanları için oluşturulan "Volume" parametresi kullanılmıştır.

Şekil 43'te görüldüğü üzere, önce merdiven elemanları seçilmekte, katı (solid) modele dönüştürülmekte ve "Solid.Volume" düğümü yardımıyla söz konusu katı geometrinin hacmi hesaplanmaktadır. Elde edilen hacim verisi m^3 cinsine dönüştürüldükten sonra, oluşturulan "Volume" parametresine aktarılmaktadır. Bu sayede, merdivenlerin beton metrajının belirlenmesinde doğrudan "Volume" parametresinden yararlanmak mümkün hâle gelmektedir.



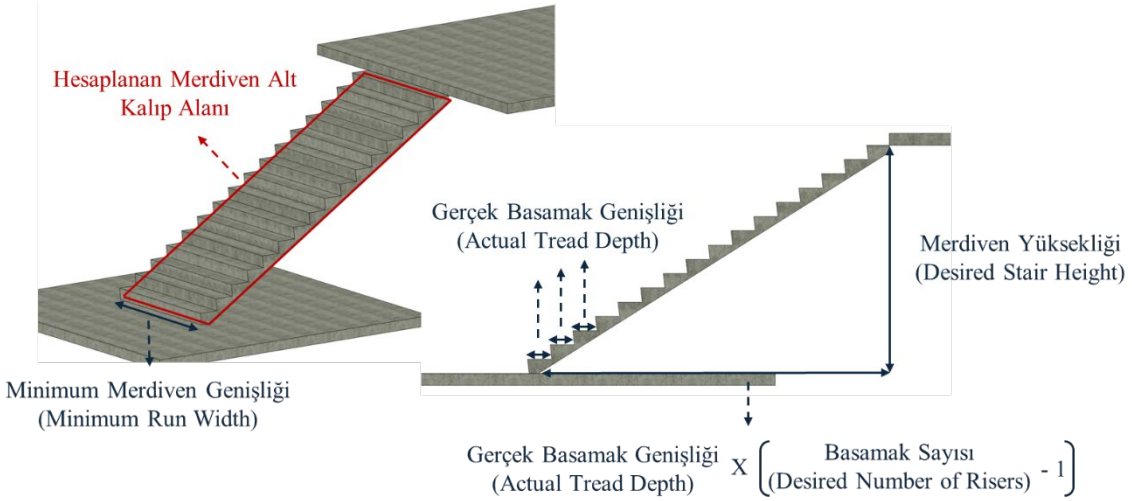
Şekil 43. Merdiven beton metrajı

Kalıp metrajı hesabı, elemanların yatay ve düşey yüzeyleri esas alınarak iki aşamada gerçekleştirilmektedir. Bu aşamaya geçmeden önce, temel, kolon, kiriş, döşeme, perde duvar ve merdiven gibi kalıp gereksinimi bulunan elemanlar filtrelenmiştir. Örneğin, kolonların ve temellerin alt-üst yüzeylerinde kalıp uygulanmazken, sadece yanal yüzeyleri kalıpla kaplanmakta, kirişlerde ise alt yüzey ile yanal yüzeylerde kalıp söz konusu olmaktadır. Bu amaçla, Şekil 44'te gösterildiği şekilde, elemanlar önce filtrelenmiş, ardından "Element.Geometry" düğümü ile katı geometrileri elde edilerek, "Solid.ByUnion" düğümüyle tek bir katı geometri şeklinde birleştirilmiştir. Daha sonra, "PolySurface.BySolid" düğümüyle söz konusu katı geometri çoklu yüzey formuna dönüştürülmüş ve "PolySurface.Surfaces" düğümüyle her bir yüzey bağımsız olarak işleme uygun hâle getirilmiştir. Yüzeyler, düşey (vertical) veya yatay (horizontal) oluşlarına göre ayrıştırılarak yüzey alanları m^2 cinsine dönüştürülmüştür. Elde edilen veriler, "Generic Model" kategorisi altındaki yüzeyler için önceden tanımlanan "Kalıp Metrajı" parametresine işlenmiştir. Böylece seçilen elemanlardan merdiven tabanı haricindeki tüm yatay ve dikey kalıp yüzeylerinin alanları hesaplanmıştır.

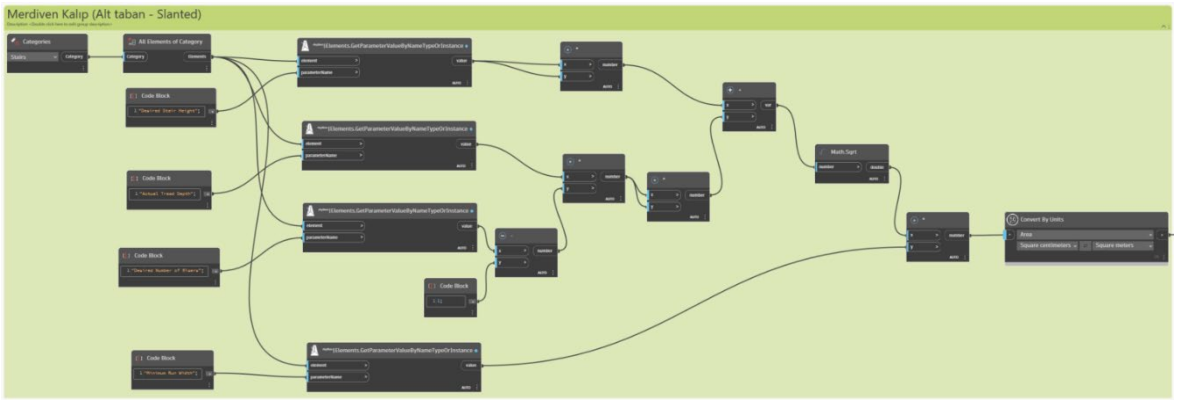


Şekil 44. Elemanların yatay ve düşey yüzeylerin seçilmesi

Merdiven taban kalıp alanını belirlemek amacıyla, Revit'te yer alan "Desired Stair Height (Merdiven Yüksekliği)", "Actual Tread Depth (Gerçek Basamak Genişliği)", "Desired Number of Risers (Basamak Sayısı)" ve "Minimum Run Width (Minimum Merdiven Genişliği)" parametrelerinden faydalanılmıştır. Şekil 45'te, bu parametreler örnek bir merdiven modeli üzerinde gösterilmektedir. "Elements.GetParameterValue ByNameTypeOrInstance" düğümü ile merdivenlerden bu bilgiler alınmakta ve Şekil 46'da Dynamo kodu ile gösterilen formülasyon yardımıyla taban kalıp metrajı değeri hesaplanmaktadır.

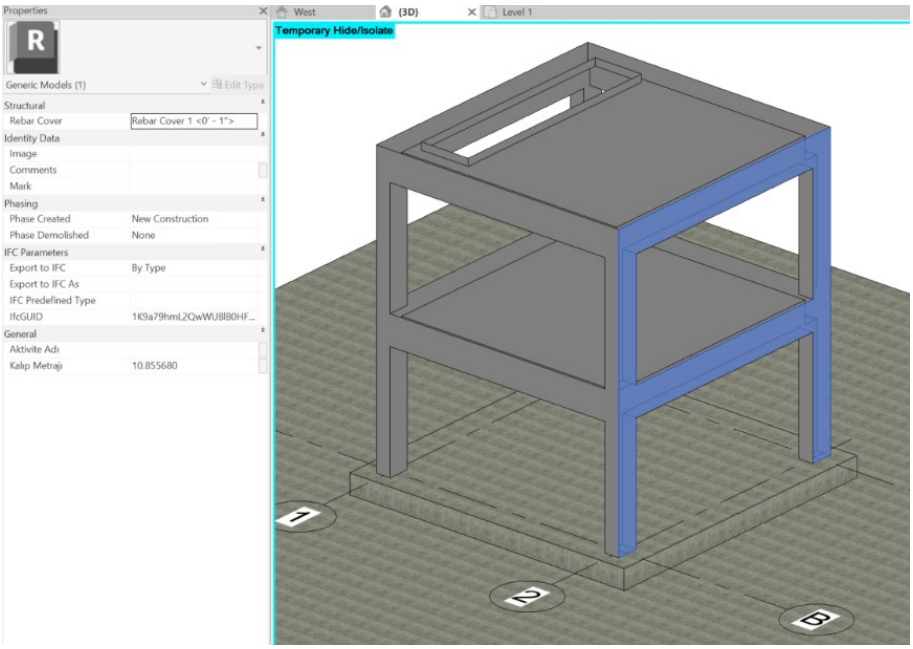


Şekil 45. Revit'te dikkate alınan merdiven parametreleri



Şekil 46. Merdiven taban kalıp metrajı

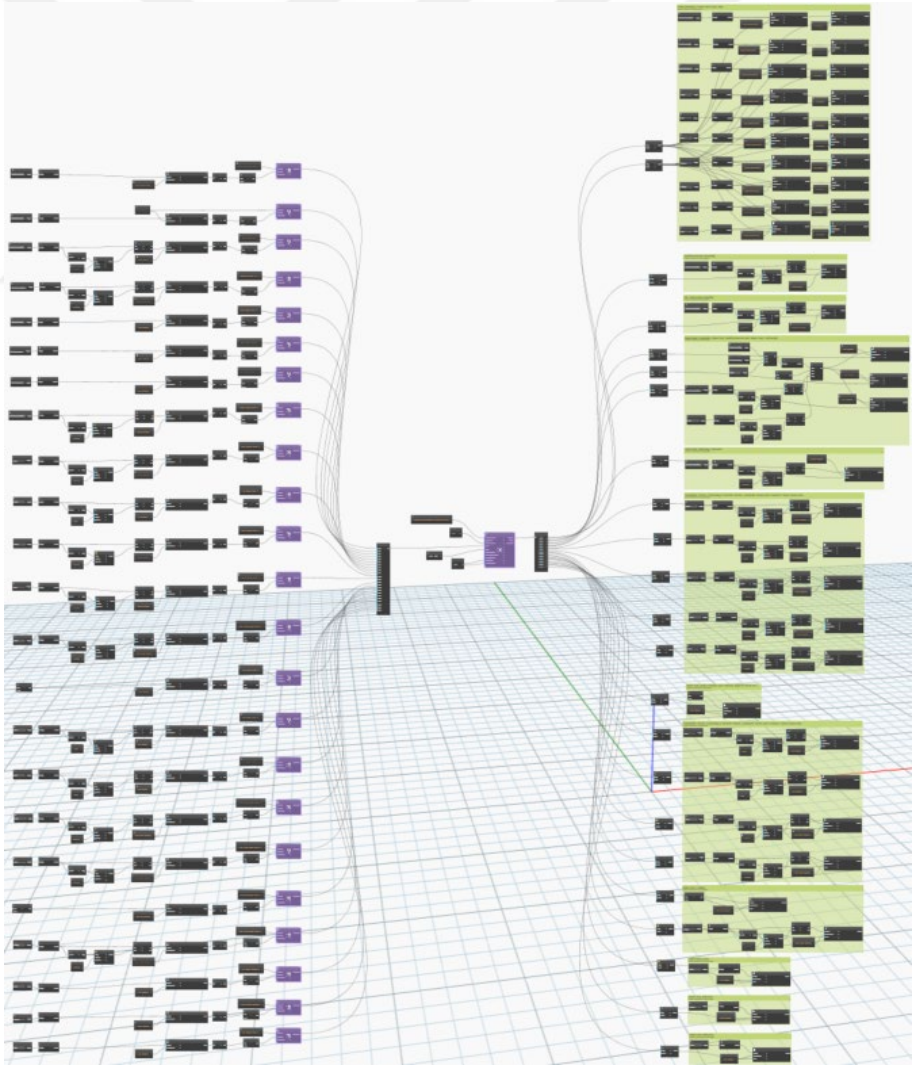
Son aşamada, yatay yüzeylerin, düşey yüzeylerin ve merdiven taban kalıp metrajı değerleri toplanarak, temel elemanları için önceden tanımlanmış olan “Toplam Kalıp Metrajı” parametresine atanmaktadır. Bu yöntemin nedeni, “Generic Model” olarak işlem gören kalıp yüzeylerinin hangi elemanlara ve hangi katlara ait olduğunun doğrudan belirlenememesi, bazı durumlarda tek bir yüzeyin birden fazla kat boyunca uzanabilmesidir. Şekil 47’de bu duruma bir örnek gösterilmektedir. Bu aşamalı yaklaşım sayesinde, modeldeki her bir elemanın kalıp ihtiyacı doğru biçimde ayrıştırılmakta ve toplam kalıp metraj verileri eksiksiz biçimde elde edilmektedir.



Şekil 47. Dynamo ile örnek bir kalıp metrajı

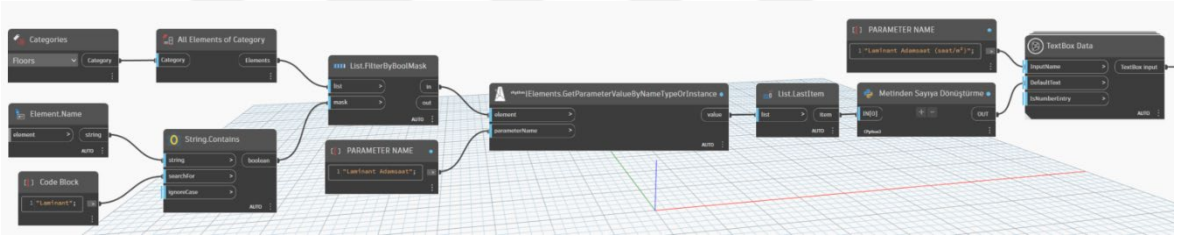
2.4.4. Kullanıcı Arayüzü ile Varsayılan Değerlerin Kontrolü

Bu aşamada, ikinci aşamada elemanlara aktarılan adam-saat, günlük çalışma süresi, ekip sayısı ve yaklaşık donatı miktarı gibi veriler ile üçüncü aşamada hesaplanan toplam kalıp metrajı bilgileri bir kullanıcı arayüzü (User Interface, UI) üzerinden görüntülenmekte ve gerektiğinde düzenlenebilmektedir. Bu yaklaşım, proje ekiplerinin kendi kurumsal koşullarına ve kaynaklarına uygun veri girişleri yapmasına imkân tanıyarak, planlama süreçlerinin gerçekçi ve özelleştirilmiş şekilde yürütülmesine olanak sağlamaktadır. Arayüzde gerçekleştirilen değişiklikler, ilgili eleman parametrelerine anında yansıtılmakta ve sonraki aşamalarda kullanılmak üzere güncellenmektedir. Şekil 48'de arayüz oluşturulma aşamasında hazırlanan Dynamo kodu gösterilmektedir.

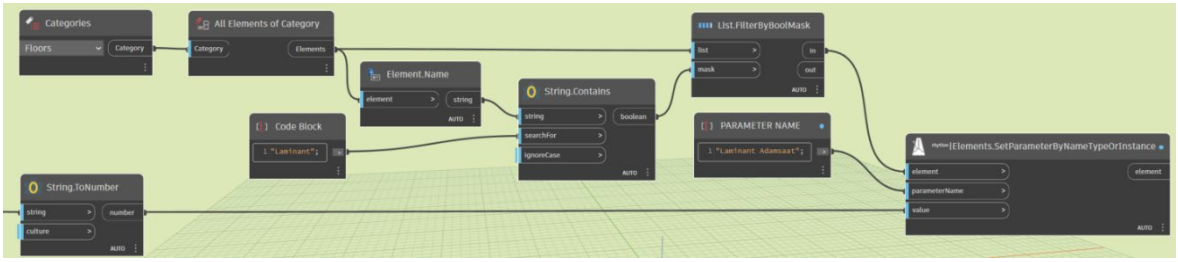


Şekil 48. Dynamo kullanıcı arayüz kodu

Bu süreçte, doğru parametre seçiminin sürekliliğini sağlamak ve arayüze girilen değeri doğru parametrelere atamak için parametreler önceki aşamalarda olduğu gibi filtrelenmiştir. Şekil 49'da, örnek olarak "Laminant Adamsaat" parametresi ve buna ait değerlerin arayüzde gösterilmesi, ardından kullanıcının tanımladığı yeni değerin aynı parametreye geri atanması süreci yer almaktadır. Parametrelerin elde edilmesi aşamasında, verileri metinden sayıya (String to Number) dönüştürme aşaması standart Dynamo düğümleriyle hedeflenen biçimde gerçekleştirilemediği durumlarda, Şekil 49(a)'da gösterildiği üzere Python kodu yardımıyla veri formatı doğru şekilde düzenlenmiştir. Şekil 49(b)'de, arayüzde değiştirilen verilerin ilgili elemanlara yeniden aktarılması gösterilmekte olup, kullanıcı herhangi bir değişiklik yapmadığında mevcut değerlerin korunması esas alınmaktadır. Diğer parametreler için de benzer yöntem izlenerek, tüm kullanıcı taleplerinin rahatlıkla uygulanabileceği esnek bir yapı oluşturulmuştur.



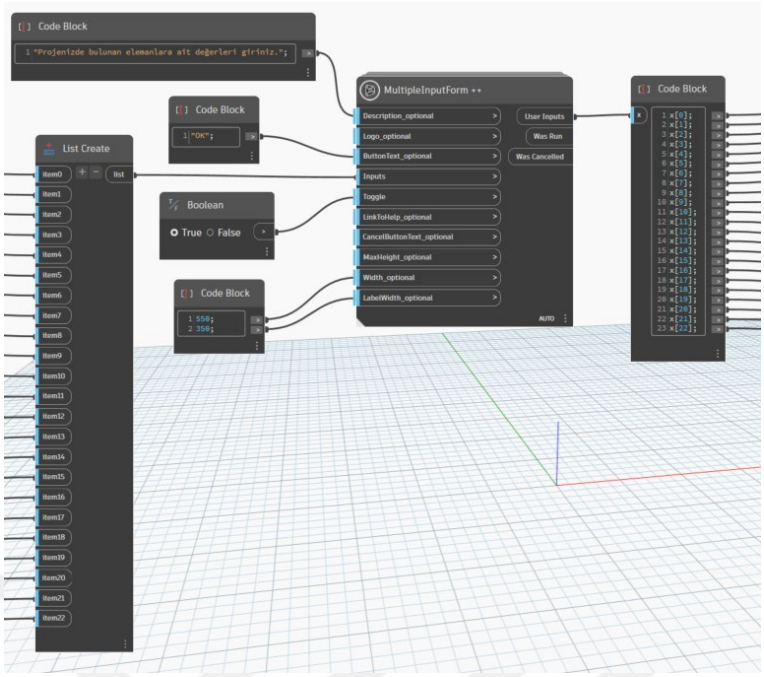
(a) Parametre değerlerinin elde edilmesi



(b) Elemanlara tekrar parametre tanımlanması

Şekil 49. (a) Parametre değerlerinin alınması ve (b) tekrar parametreye tanımlanması

Son olarak, Şekil 50'de arayüz oluşturma aşamasında kullanılan "MultipleInputForm++" düğümünün giriş ve çıkış verileriyle ilişkilendirilmesi gösterilmektedir. Bu aşamada, açılan arayüzün boyutları, açıklaması ve kurumsal kimliği yansıtan amblem gibi özellikler tanımlanabilmekte; böylelikle proje paydaşlarına daha kullanıcı dostu bir arayüz sunulmaktadır.



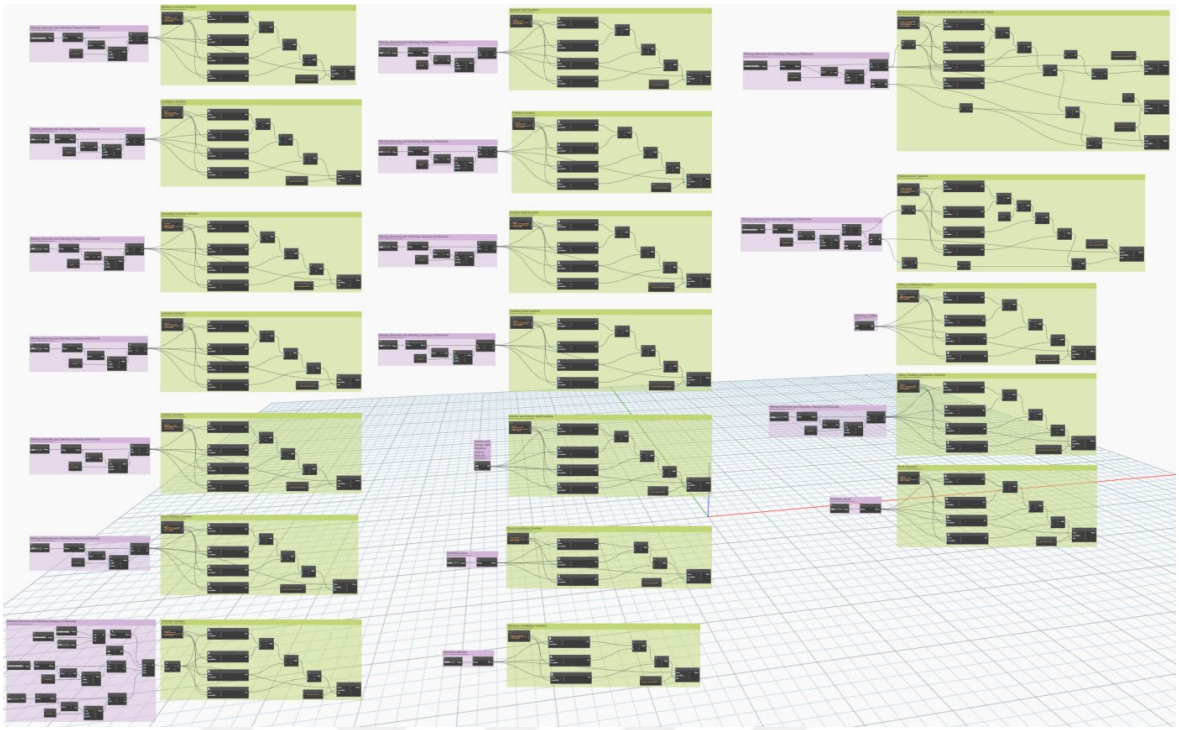
Şekil 50. Kullanıcı arayüzü düğümü

2.4.5. Aktivite Sürelerinin Hesaplanması

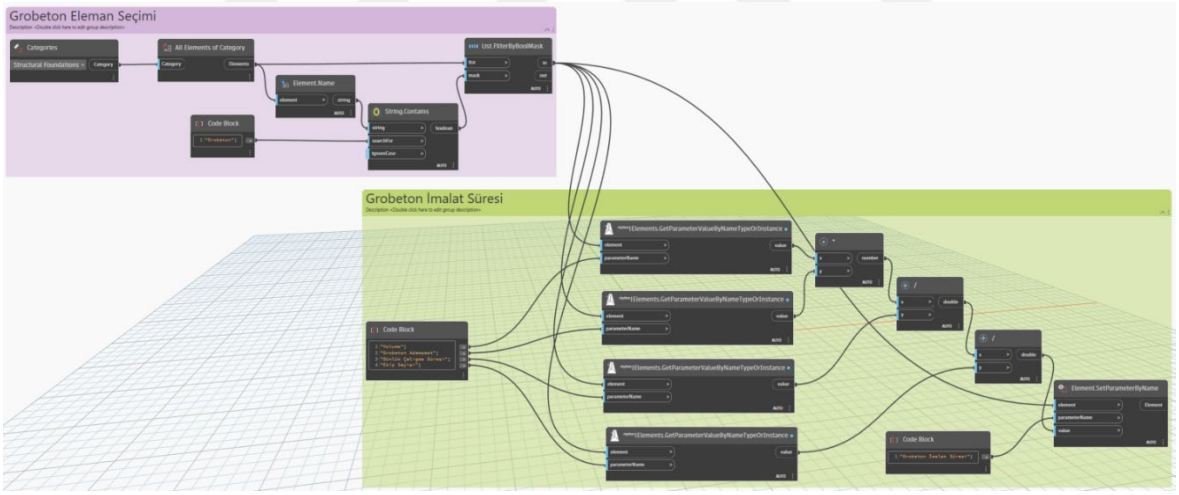
Bu aşamada, önceki kodlama adımlarında elde edilen tüm veriler kullanılarak her bir imalat kalemine ait aktivite süreleri hesaplanmaktadır. Şekil 51'de bu amaç doğrultusunda geliştirilen Dynamo koduna ilişkin genel görünüm yer almaktadır. Hesaplama sürecinde, metraj değerlerinin (yaklaşık donatı, toplam kalıp metrajı vb.) yanı sıra, Revit'te tanımlı parametreler ve kullanıcı arayüzünde değiştirilebilecek nitelikte tanımlanmış olan adam-saat, günlük çalışma süresi ve ekip sayısı gibi parametreler dikkate alınmıştır. Bu parametrelerin Revit içerisinde yeni oluşturulan veya hali hazırda mevcut olan değerleri, aktivite sürelerinin güvenilir ve esnek bir şekilde belirlenmesine olanak tanımaktadır. Tablo 6'da eleman bazlı metraj hesabında kullanılan parametreler ile aktivite sürelerinin tanımlandığı parametre isimleri özetlenmektedir. Her bir elemanın kendine özgü metraj hesaplaması olması nedeniyle, süre hesaplaması tüm elemanlar için ayrı ayrı yürütülmüştür. Örneğin, grobetona ilişkin süre hesabında Revit'te mevcut "Volume" parametresi üzerinden grobeton metrajı elde edilmiş; bu değer, "Grobeton Adamsaat" ile çarpılmış, "Günlük Çalışma Süresi" ve "Ekip Sayısı" parametrelerine bölünerek "Grobeton İmalat Süresi" hesaplanmıştır. Şekil 52'de "Grobeton İmalat Süresi" değerinin elde edilmesi ve parametreye tanımlanması gösterilmektir.

Tablo 6. Aktivite süresi hesabında kullanılan parametreler

Aktivite Süresi Parametresi	**Aktivite Süresi Hesaplanan Elemanlar**	**Kullanılan Parametreler**
Beton Döküm Süresi	Temel (Grobeton hariç), Kolon, Kiriş, Merdiven, Döşeme (Yalıtım, Şap, Laminant, Seramik, Kaplama hariç), Perde Duvar	Volume, Beton Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Kalıp İmalat Süresi	Temel (Grobeton hariç), Döşeme (Yalıtım, Şap, Laminant, Seramik, Kaplama hariç)	Toplam Kalıp Metrajı, Kalıp Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Donatı İmalat Süresi	Temel (Grobeton hariç), Döşeme (Yalıtım, Şap, Laminant, Seramik, Kaplama hariç)	Donatı Metrajı, Donatı Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Grobeton İmalat Süresi	Grobeton	Volume, Grobeton Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Yalıtım İmalat Süresi	Yalıtım	Area, Yalıtım Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Şap İmalat Süresi	Şap	Volume, Şap Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Laminant İmalat Süresi	Laminant	Area, Laminant Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Seramik İmalat Süresi	Seramik	Area, Seramik Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Zemin Kaplama İmalat Süresi	Kaplama	Area, Zemin Kaplama Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Duvar İmalat Süresi	Duvar (Sıva, Boya, Seramik, Kaplama, Perde hariç)	Area, Duvar Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Sıva İmalat Süresi	Sıva	Area, Sıva Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Boya İmalat Süresi	Boya	Area, Boya Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Duvar Seramik İmalat Süresi	Seramik	Area, Duvar Seramik Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Duvar Kaplama İmalat Süresi	Kaplama	Area, Duvar Kaplama Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Asma Tavan İmalat Süresi	Asma Tavan (Kaplama hariç)	Area, Asma Tavan Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Tavan Kaplama İmalat Süresi	Kaplama	Area, Tavan Kaplama Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Kapı İmalat Süresi	Kapı	Kapı Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Pencere İmalat Süresi	Pencere	Pencere Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı
Çatı İmalat Süresi	Çatı	Area, Çatı Adamsaat, Günlük Çalışma Süresi, Ekip Sayısı



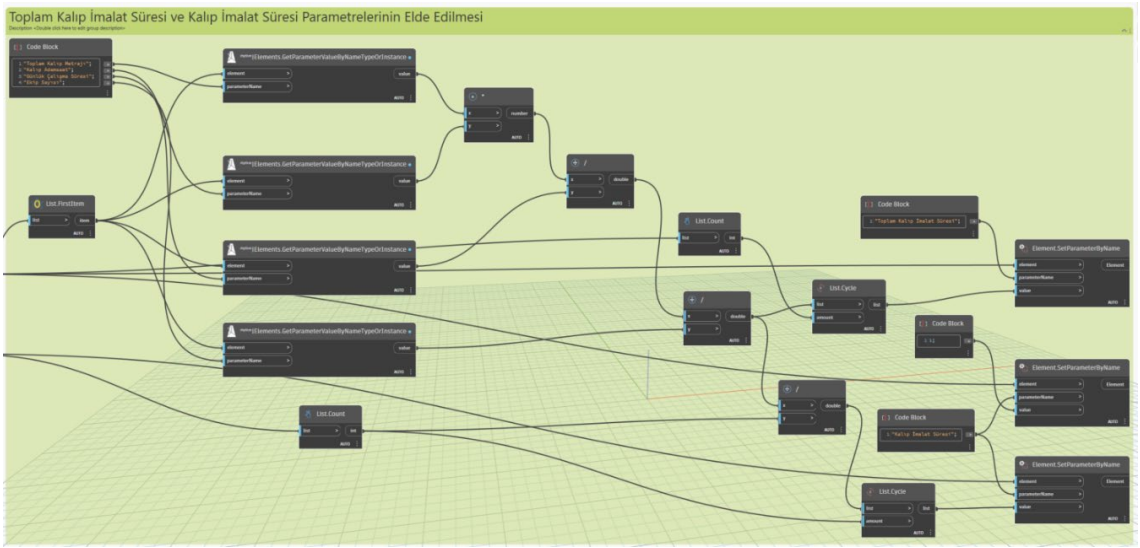
Şekil 51. Aktivite sürelerinin hesaplanması Dynamo kodu



Şekil 52. Dynamo ile grobeton imalat süresi hesabı

Kalıp ve donatı imalat sürelerinin belirlenmesi ve ilgili elemanlara tanımlanmasında, önceki kodlama aşamalarından farklı yaklaşımlar uygulanmıştır. Kalıp imalat süresi hesabında, temel elemanları için tanımlanan “Toplam Kalıp Metrajı”, “Kalıp Adamsaat”, “Günlük Çalışma Süresi” ve “Ekip Sayısı” parametreleri bir araya getirilerek binadaki “Toplam Kalıp İmalat Süresi” oluşturulmuş ve bu değer temel elemanlarına aktarılmıştır. Ancak, iş programı hazırlanırken her kata ait kalıp süresinin ayrı bir aktivite olarak

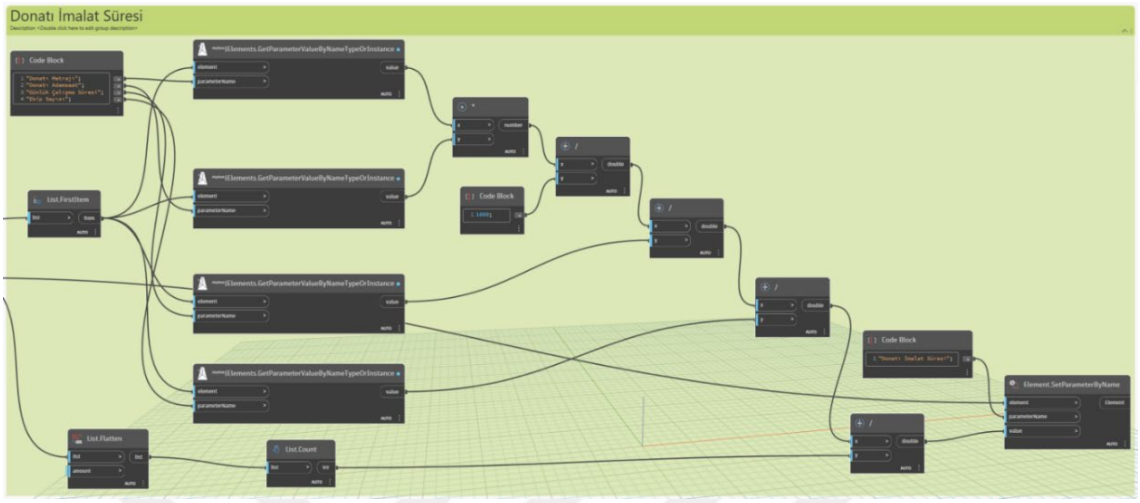
girilmesi amaçlandığından, tez kapsamında temel imalatının “Kalıp İmalat Süresi” 1 gün olarak varsayılmış; bununla birlikte “Toplam Kalıp Metrajı” BIM modelindeki bina kat sayısına bölünerek döşeme elemanlarındaki “Kalıp İmalat Süresi” parametresine tanımlanmıştır. Şekil 53’te, bu yaklaşım çerçevesinde geliştirilmiş Dynamo kodunun (eleman filtreleme aşaması hariç) görünümü yer almaktadır. Bu aşamada “List.Count” düğümü aracılığıyla önceki düğümden gelen eleman veya değer sayısı tespit edilmiş, “List.Cycle” düğümü ise bu sayıya bağlı olarak tekrarlı değer atamayı mümkün kılmıştır. Bu yöntemle, elde edilen hesaplama sonuçları tüm döşeme elemanlarına sistematik bir biçimde aktarılmıştır.



Şekil 53. Dynamo ile kalıp imalat süresi hesabı

Donatı imalat süresinin hesaplanmasında ise, temeller için tanımlanmış “Donatı Metrajı” parametresinden gelen, önceki aşamalarda kg cinsinde elde edilmiş veriler öncelikle ton (t) birimine dönüştürülmüştür. Daha sonra, “Donatı Adamsaat”, “Günlük Çalışma Süresi” ve “Ekip Sayısı” parametreleri dikkate alınarak, binanın toplam donatı imalat süresi elde edilmiştir. Bu toplam süre, kalıp imalatında olduğu gibi belirli bir dağıtım yöntemiyle temel ve döşeme elemanlarına eşit olarak atanmıştır. Ancak kalıp imalatından farklı olarak, örneğin dört katlı bir binada, temel ve dört döşeme olmak üzere toplam beş imalat birimi dikkate alınmış; oluşturulan toplam donatı imalat süresi beşe bölünerek, “Donatı İmalat Süresi” parametresi temel ve döşemelerin her birine eşit dağıtılacak şekilde tanımlanmıştır. Bu yöntem, donatı işlerinin iş programında kat bazında planlanabilir ve takip edilebilir olmasını sağlamaktadır. Şekil 54’te bu yaklaşım

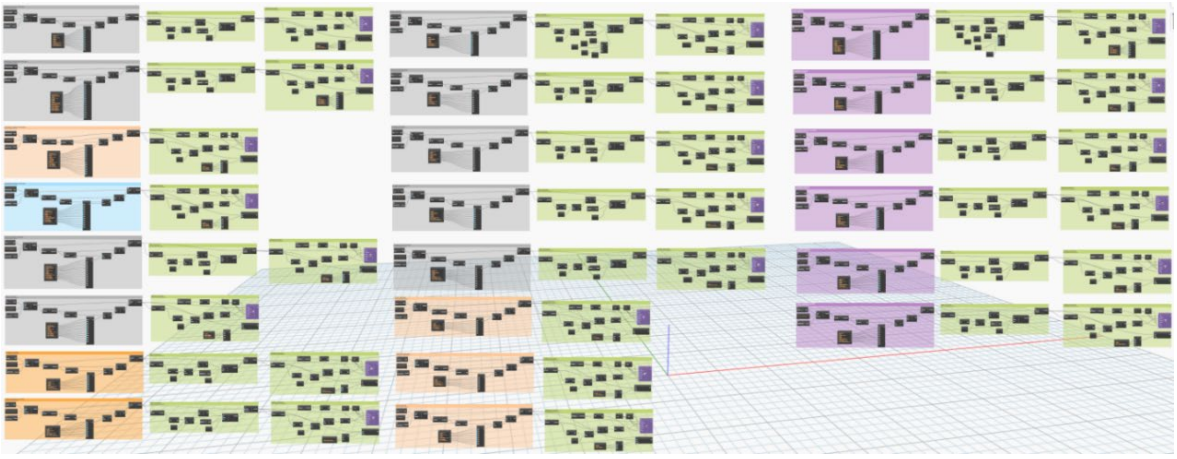
çerçevesinde geliştirilmiş Dynamo kodunun (eleman filtreleme aşaması hariç) görünümü yer almaktadır.



Şekil 54. Dynamo ile donatı imalat süresi hesabı

2.4.6. Metraj Listelerinin Oluşturulması

Bu aşamada, kullanıcıların hızla metraj listeleri oluşturabilmesine yönelik bir Dynamo kodu geliştirilmiştir. Böylelikle, elemanların alan ve hacim gibi temel niceliklerinin yanı sıra, önceki aşamalarda oluşturulan ve belirli değerler atanan parametrelerin de kontrolü sağlanabilmektedir. Söz konusu kontrol adımı, iş programının doğru biçimde oluşturulmasına katkı sağlayan kritik bir unsur olarak tasarlanmıştır. Şekil 55'te, bahsi geçen Dynamo kodunun genel yapısı gösterilmektedir.



Şekil 55. Metraj listeleri Dynamo kodu

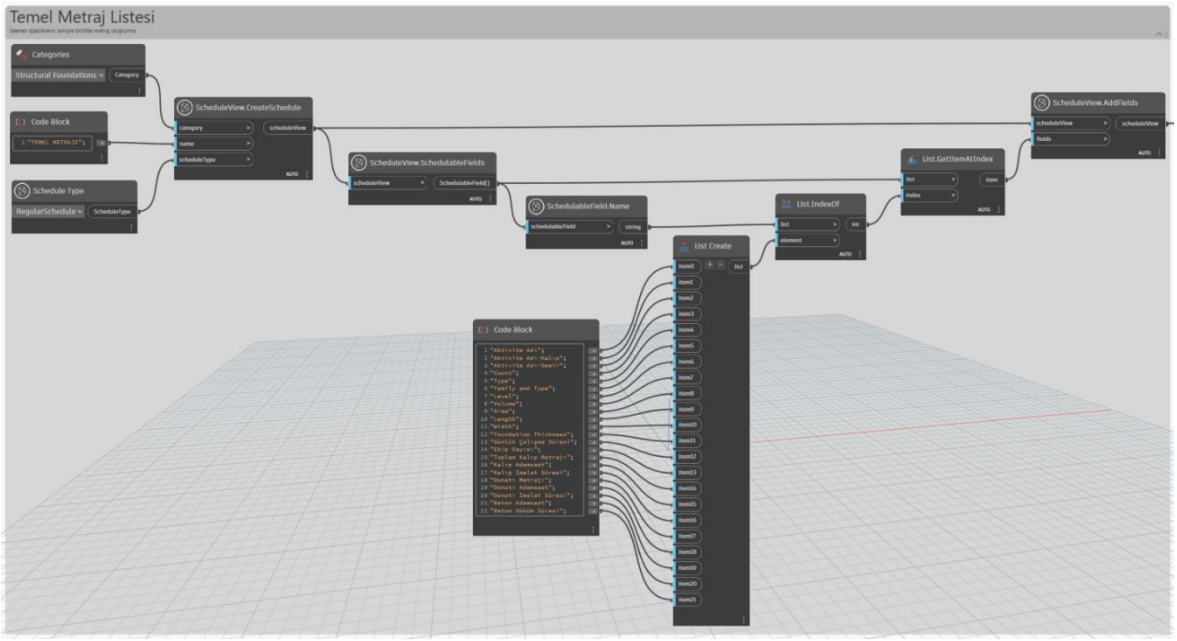
İlk olarak, ilgili kategoriler ve metraj listesinde yer alması istenen parametreler belirlenerek listeler oluşturulmuştur. Daha önce de ifade edildiği gibi, her elemanın kendi metraj hesabıyla birlikte, o elemana özgü parametreler metraj listesi içinde yer almaktadır. Bu ilişki, Tablo 7'de belirtilen elemanlar ve bu elemanlara ait parametreler üzerinden netleştirilmektedir. Şekil 56'da ise "Temel Metrajı" adıyla bir metraj listesinin oluşturulması örneklenmektedir. Ardından, kategori bazında seçilen tüm elemanların metraj listeleri, eleman isimlerine göre filtrelenerek yalnızca istenilen öğelerin görünürlüğü sağlanmıştır. Daha sonra, oluşturulan listelerin katlara göre küçükten büyüğe doğru sıralanması ve önemli olduğu değerlendirilen parametrelerin toplamlarının metraj listelerinde yer alması için ek işlemler yapılmıştır. Bu süreçte Python kodu da kullanılarak, sıralama aşamasının daha esnek ve hızlı gerçekleştirilmesi amaçlanmıştır. Şekil 57 ve 58'de, "Temel Metrajı" kapsamında yürütülen filtreleme ve sıralama işlemlerine ait örnek kod bölümleri sırasıyla gösterilmektedir.

Tablo 7. Metraj listeleri ve kullanılan parametreler

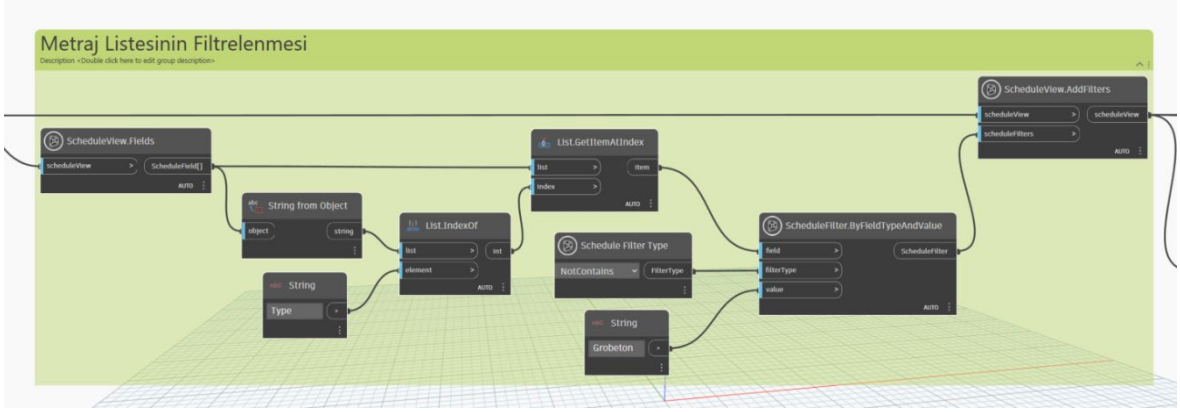
Metraj Listesi	**Parametreler**
Grobeton Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Volume, Area, Length, Width, Foundation Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Grobeton Adamsaat, Grobeton İmalat Süresi
Temel Metrajı	Aktivite Adı, Aktivite Adı-Kalıp, Aktivite Adı-Demir, Count, Type, Family and Type, Level, Volume, Area, Length, Width, Foundation Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Toplam Kalıp Metrajı, Kalıp Adamsaat, Kalıp İmalat Süresi, Donatı Metrajı, Donatı Adamsaat, Donatı İmalat Süresi, Beton Adamsaat, Beton Döküm Süresi
Kolon Metrajı	Aktivite Adı, Aktivite Adı-Kalıp, Aktivite Adı-Demir, Count, Type, Family and Type, Volume, Top Offset, Base Offset, Top Level, Base Level, Length, Günlük Çalışma Süresi, Ekip Sayısı, Kalıp Adamsaat, Donatı Adamsaat, Beton Adamsaat, Beton Döküm Süresi
Kiriş Metrajı	Aktivite Adı, Aktivite Adı-Kalıp, Aktivite Adı-Demir, Count, Type, Family and Type, Volume, Length, Reference Level, Reference Level Elevation, Günlük Çalışma Süresi, Ekip Sayısı, Kalıp Adamsaat, Donatı Adamsaat, Beton Adamsaat, Beton Döküm Süresi
Perde Duvar Metrajı	Aktivite Adı, Aktivite Adı-Kalıp, Aktivite Adı-Demir, Count, Type, Family and Type, Volume, Area, Length, Base Constraint, Width, Günlük Çalışma Süresi, Ekip Sayısı, Kalıp Adamsaat, Donatı Adamsaat, Beton Adamsaat, Beton Döküm Süresi
Merdiven Metrajı	Aktivite Adı, Aktivite Adı-Kalıp, Aktivite Adı-Demir, Count, Type, Family and Type, Volume, Base Level, Top Level, Günlük Çalışma Süresi, Ekip Sayısı, Kalıp Adamsaat, Donatı Adamsaat, Beton Adamsaat, Beton Döküm Süresi
Döşeme Metrajı	Aktivite Adı, Aktivite Adı-Kalıp, Aktivite Adı-Demir, Count, Type, Family and Type, Level, Volume, Area, Default Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Kalıp Adamsaat, Kalıp İmalat Süresi, Donatı Adamsaat, Donatı İmalat Süresi, Beton Adamsaat, Beton Döküm Süresi
Duvar Metrajı	Aktivite Adı, Count, Type, Family and Type, Base Constraint, Width, Volume, Area, Length, Günlük Çalışma Süresi, Ekip Sayısı, Duvar Adamsaat, Duvar İmalat Süresi
Sıva Metrajı	Aktivite Adı, Count, Type, Family and Type, Base Constraint, Width, Volume, Area, Length, Günlük Çalışma Süresi, Ekip Sayısı, Sıva Adamsaat, Sıva İmalat Süresi
Boya Metrajı	Aktivite Adı, Count, Type, Family and Type, Base Constraint, Width, Volume, Area, Length, Günlük Çalışma Süresi, Ekip Sayısı, Boya Adamsaat, Boya İmalat Süresi

Tablo 7'nin devamı

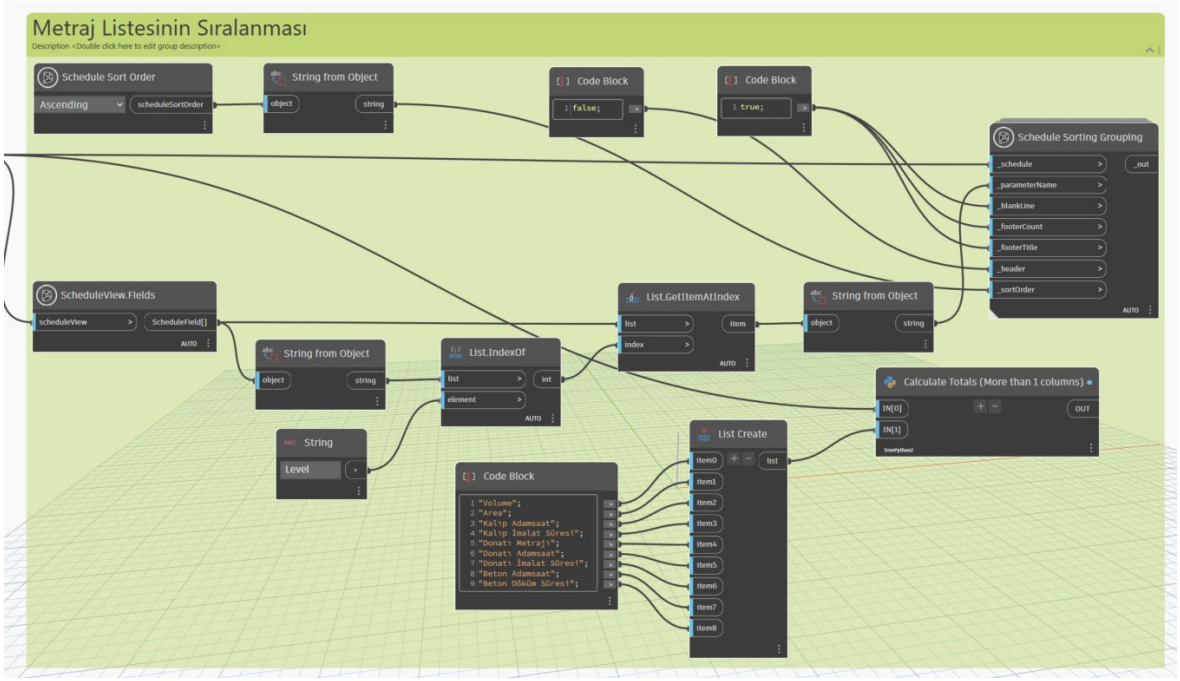
Seramik Duvar Metrajı	Aktivite Adı, Count, Type, Family and Type, Base Constraint, Width, Volume, Area, Length, Günlük Çalışma Süresi, Ekip Sayısı, Duvar Seramik Adamsaat, Duvar Seramik İmalat Süresi
Duvar Kaplama Metrajı	Aktivite Adı, Count, Type, Family and Type, Base Constraint, Width, Volume, Area, Length, Günlük Çalışma Süresi, Ekip Sayısı, Duvar Kaplama Adamsaat, Duvar Kaplama İmalat Süresi
Şap Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Volume, Area, Default Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Şap Adamsaat, Şap İmalat Süresi
Yalıtım Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Volume, Area, Default Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Yalıtım Adamsaat, Yalıtım İmalat Süresi
Laminant Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Volume, Area, Default Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Laminant Adamsaat, Laminant İmalat Süresi
Seramik Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Volume, Area, Default Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Seramik Adamsaat, Seramik İmalat Süresi
Zemin Kaplama Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Volume, Area, Default Thickness, Günlük Çalışma Süresi, Ekip Sayısı, Zemin Kaplama Adamsaat, Zemin Kaplama İmalat Süresi
Asma Tavan Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Area, Günlük Çalışma Süresi, Ekip Sayısı, Asma Tavan Adamsaat, Asma Tavan İmalat Süresi
Tavan Kaplama Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Area, Günlük Çalışma Süresi, Ekip Sayısı, Tavan Kaplama Adamsaat, Tavan Kaplama İmalat Süresi
Kapı Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Günlük Çalışma Süresi, Ekip Sayısı, Kapı Adamsaat, Kapı İmalat Süresi
Pencere Metrajı	Aktivite Adı, Count, Type, Family and Type, Level, Günlük Çalışma Süresi, Ekip Sayısı, Pencere Adamsaat, Pencere İmalat Süresi
Çatı Metrajı	Aktivite Adı, Count, Type, Family and Type, Volume, Area, Base Level, Günlük Çalışma Süresi, Ekip Sayısı, Çatı Adamsaat, Çatı İmalat Süresi



Şekil 56. Temel metraj listesi



Şekil 57. Metraj listesinin filtrelenmesi

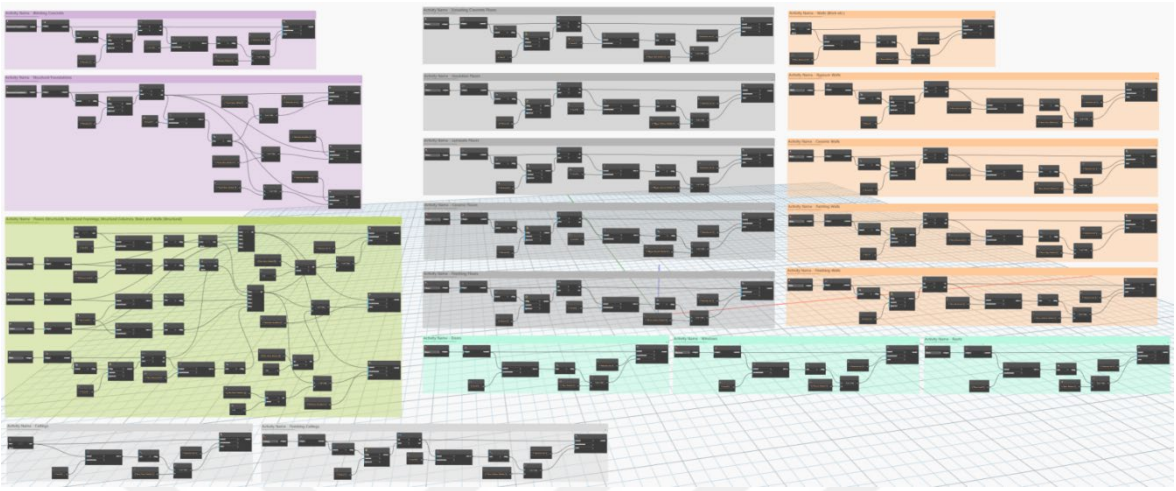


Şekil 58. Metraj listesinin sıralanması

2.4.7. Aktivite Adlarının Oluşturulması

Bu aşamada, iş programında yer alacak aktivitelerin isimleri, ML çıktılarıyla uyumlu olacak şekilde sistematik olarak oluşturulmaktadır. ML aşamasında kullanılacak aktivite adlarıyla tutarlılık sağlanabilmesi için, isimlerin önceden belirlenmiş bir kurgu içinde tanımlanması büyük önem taşımaktadır. Söz konusu kurgu, elemanın bulunduğu kat seviyesi (örneğin, “Level 1”, “Level 2” vb.), eleman adı (örneğin, “Grobeton”, “Seramik”, “Temel”, “Kiriş” vb.) ve ilgili işlemi (örneğin, “İmalatı”, “Kalıp İmalatı”, “Beton

Dökümü" vb.) içerecek şekilde şekillendirilmiştir. Bu kapsamda, Şekil 59'da aktivite isimlendirmesine yönelik Dynamo kodunun genel yapısı gösterilmektedir.

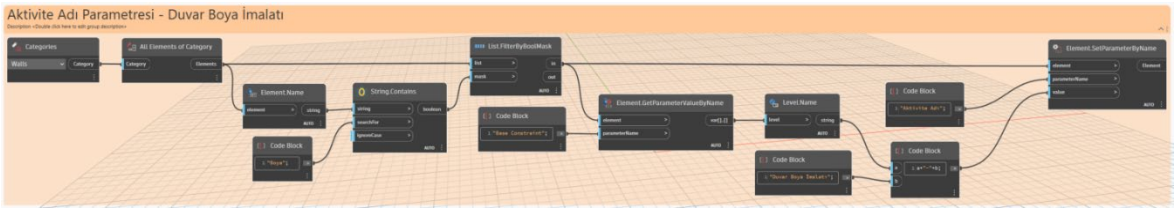


Şekil 59. Aktivite adlarının oluşturulması Dynamo kodu

İlk olarak, her elemanın kat seviyesi "Element.GetParameterValueByName" düğümü aracılığıyla elde edilmektedir. Ancak, farklı eleman tiplerinin Revit içerisindeki modelleme yaklaşımı nedeniyle kat seviyesi parametreleri değişiklik göstermektedir. Örnek olarak, grobeton için "Level" parametresi gerekirken, kirişler "Reference Level" parametresine, kolonlar "Base Level" parametresine, duvarlar ise "Base Constraint" parametresine sahiptir. Tablo 8'de, elemanlar ve bunlara ait kat seviyesi parametreleri toplu halde sunulmaktadır. Elde edilen kat seviyesi ile elemanlara özel belirlenen aktivite isimleri, bir "Code Block" düğümü yardımıyla birleştirildikten sonra "Aktivite Adı" parametresine "Element.SetParameterByName" düğümü ile tanımlanmaktadır. Şekil 60'da ise boya işlemi için geliştirilen kod örneği yer almaktadır.

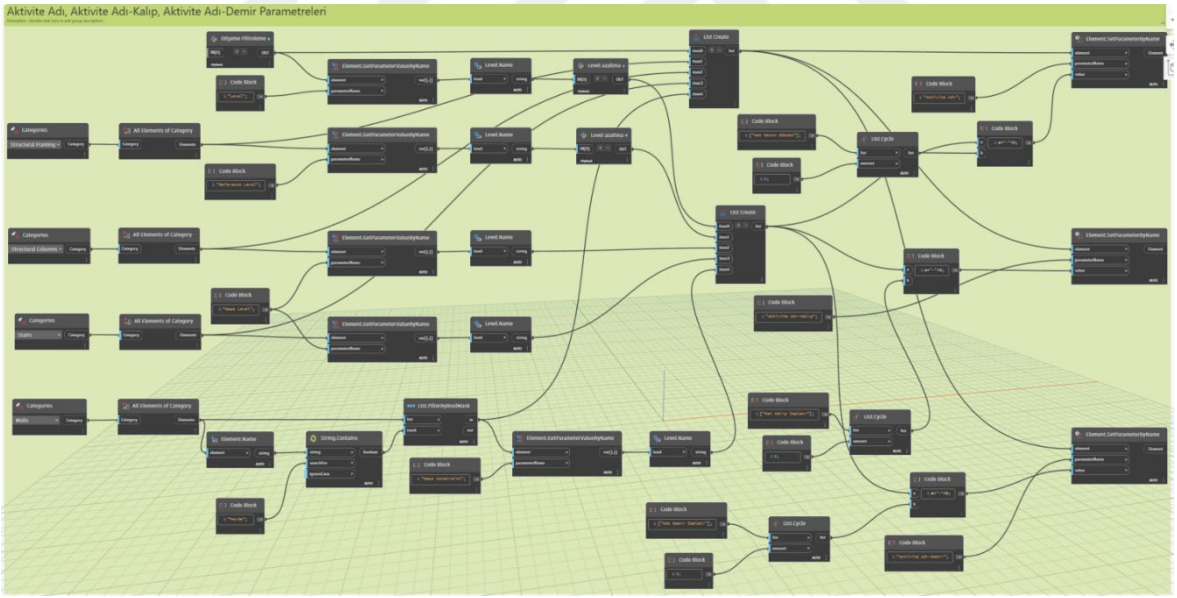
Tablo 8. Elemanların kat seviyeleri için kullanılan parametreler

Elemanlar	**Kat Seviyesi Parametreleri**
Grobeton, Temel, Döşeme, Şap, Yalıtım, Laminant, Döşeme Seramik, Zemin Kaplama, Asma Tavan, Tavan Kaplama, Kapı, Pencere	Level
Perde Duvar, Duvar, Sıva, Duvar Seramik, Boya, Duvar Kaplama	Base Constraint
Kolon, Merdiven, Çatı	Base Level
Kiriş	Reference Level



Şekil 60. Duvar boya imalatı Dynamo kodu

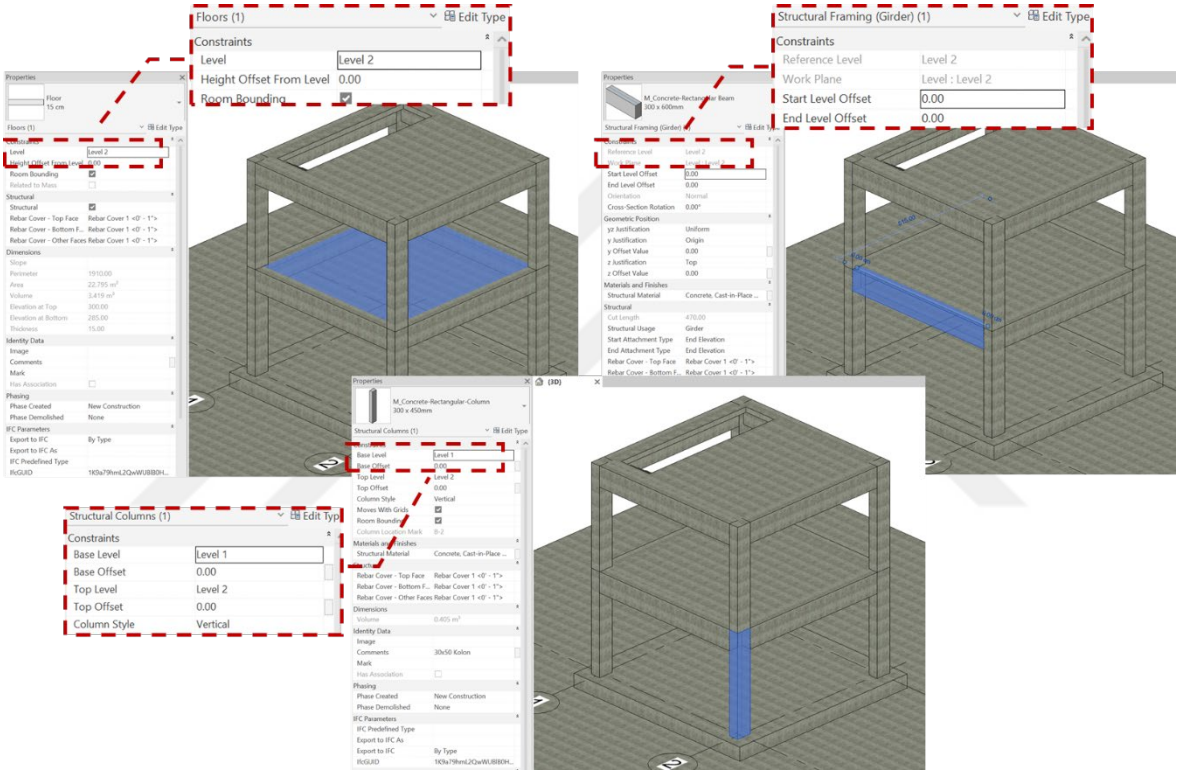
Kalıp, donatı ve beton dökümüne ilişkin aktiviteler, diğer imalatlardan farklı bir yaklaşım gerektirmektedir. Diğer elemanların her biri için tek bir aktivite üretilirken, yapısal elemanlar (kolonlar, kirişler vs.) için "Kalıp İmalatı", "Demir İmalatı" ve "Beton Dökümü" olmak üzere üç ayrı aktivitenin tanımlanması zorunludur. Bu gereksinime çözüm getirebilmek adına, yapısal elemanlar için ilk Dynamo aşamasında "Aktivite Adı", "Aktivite Adı-Kalıp" ve "Aktivite Adı-Demir" şeklinde üç ayrı parametre oluşturulmuştur. Şekil 61'de, sözü edilen üç parametreye ait değerlerin atandığı Dynamo kodu gösterilmektedir.



Şekil 61. Kalıp, demir ve beton aktivitelerinin isimlendirilmesi Dynamo kodu

Ayrıca, döşeme ve kiriş elemanlarında kat seviyesi değerinin bir kademe düşürülmesi için Python kodundan yararlanılmıştır. Bu tercih, Revit'in modelleme yaklaşımından kaynaklanan farklılıkları iş programına yansıtma sürecini kolaylaştırmak amacıyla yapılmıştır. Örneğin, birinci ve ikinci kat arasında bulunan bir kolonun "Base Level" değeri "Level 1" olarak tanımlanırken, aynı katta yer alan kirişin "Reference Level" ve

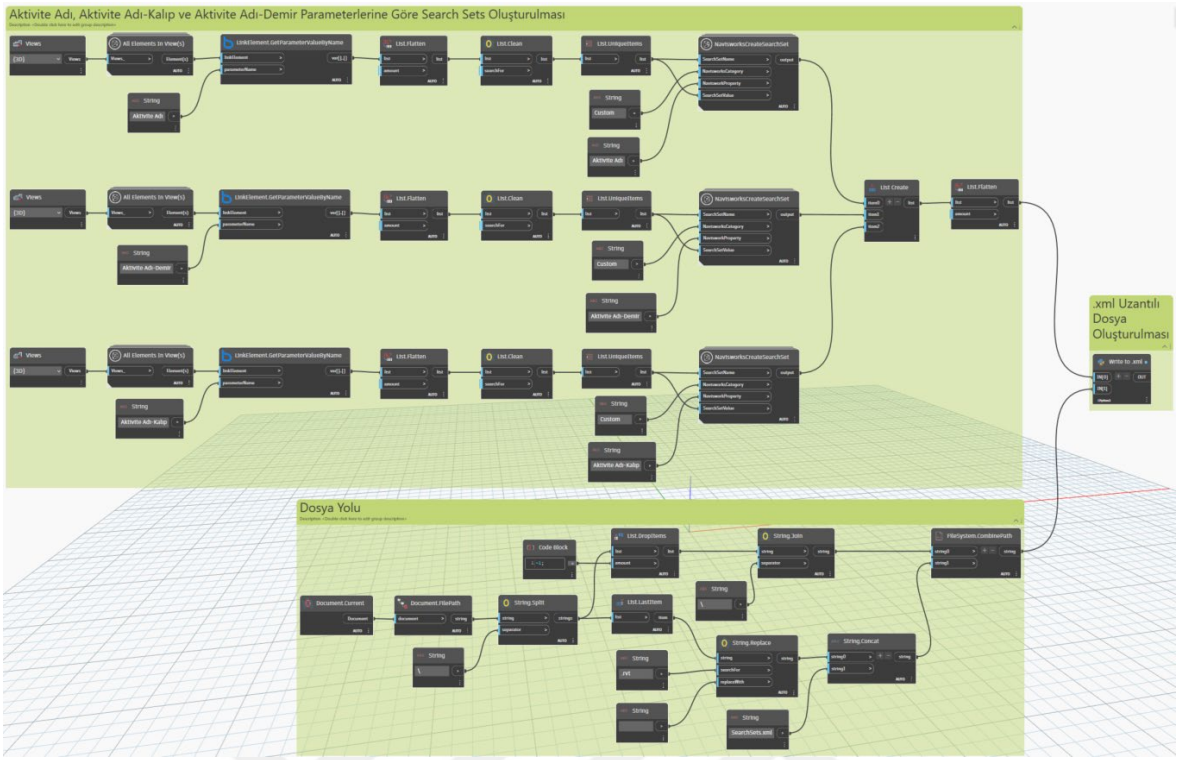
döşemenin “Level” parametreleri “Level 2” değerini almaktadır. Bu durumda, kolon ile beraber kalıp, donatı ve beton döküm işlemlerine tabi tutulacak kiriş ve döşeme elemanlarında iş programı açısından “Level 1” isminin kullanılması tercih edilmiştir. Şekil 62’de, örnek bir Revit modelinde bu duruma ait görsel açıklama sunulmaktadır. Python kodu, 3B BIM modelinde herhangi bir değişiklik yapmadan, yalnızca “Aktivite Adı” parametresi üzerinde kat seviyesi ismini düzenleyerek tutarlı bir aktivite tanımlamasını mümkün kılmaktadır.



Şekil 62. Örnek bir Revit modelinde elemanların bağlı oldukları kat gösterimi

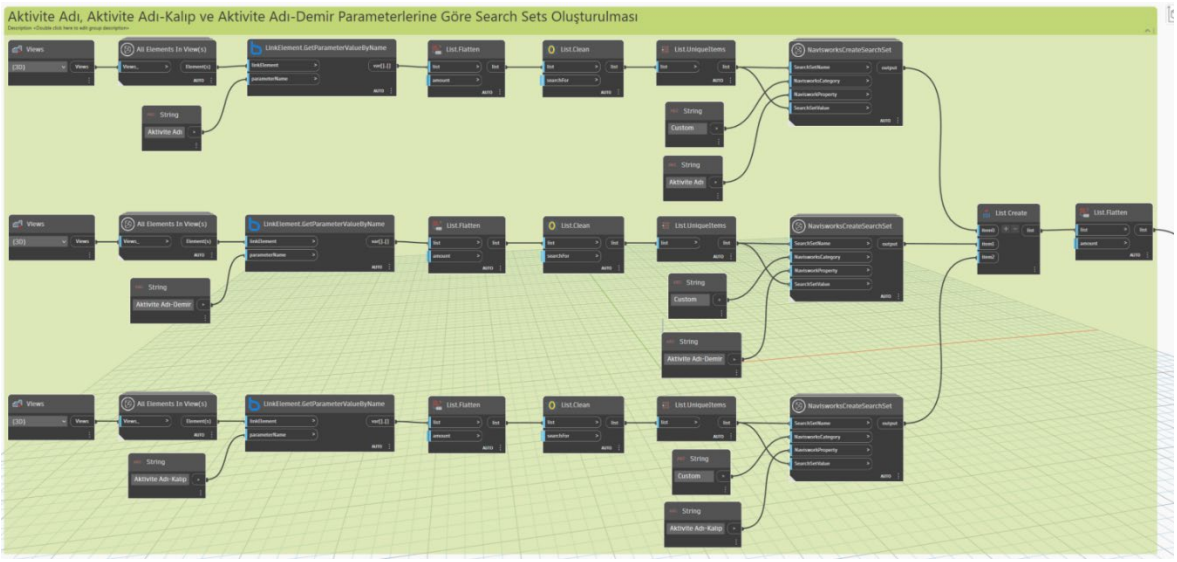
2.4.8. Navisworks Search Sets Dosyasının Oluşturulması

Bu aşamada, 4B model oluşturma sürecinde elemanlar ile aktivitelerin ilişkilendirilmesini kolaylaştırmak amacıyla Navisworks yazılımında kullanılacak “Search Sets” gruplarının hızlıca üretilebilmesi için “.xml” uzantılı bir dosya oluşturulmuştur. Böylece, Navisworks ortamında elemanları kolaylıkla seçerek farklı analiz süreçlerine dâhil etmek mümkün olmaktadır. Şekil 63’te, söz konusu Dynamo kodunun üç aşamalı yapısı sunulmaktadır.



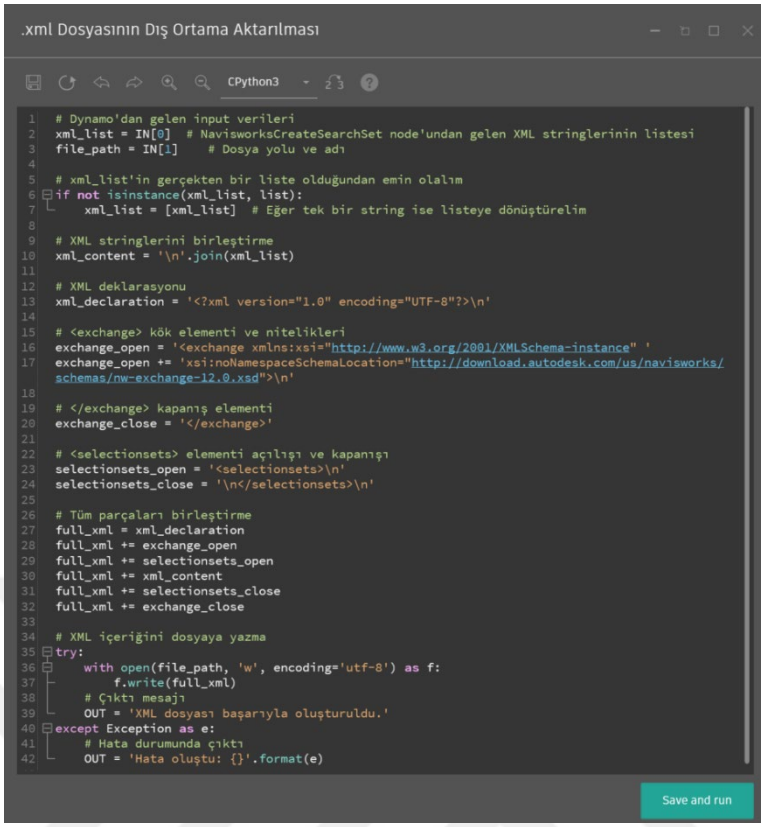
Şekil 63. Search Sets oluşturulması Dynamo kodu

İlk olarak, “NavisworksCreateSearchSet” düğümü yardımıyla önceden belirlenmiş kurallar ışığında model öğeleri filtrelenmiştir. Bu çalışma kapsamında üç kural tanımlanmıştır. Navisworks’te “Custom” kategorisi altında listelenen “Aktivite Adı”, “Aktivite Adı-Kalıp” ve “Aktivite Adı-Demir” parametre değerleri esas alınarak “Search Sets” kümeleri oluşturulmaktadır. Bu yaklaşımın temel amacı, bir önceki aşamada elemanlara tanımlanmış olan aktivitelerin, Navisworks platformunda aynı aktivite adlarına göre gruplandırılabilmesini sağlamaktır. Şekil 64’te, “Search Sets” oluşturulma aşamasının ayrıntılı görseli yer almaktadır.

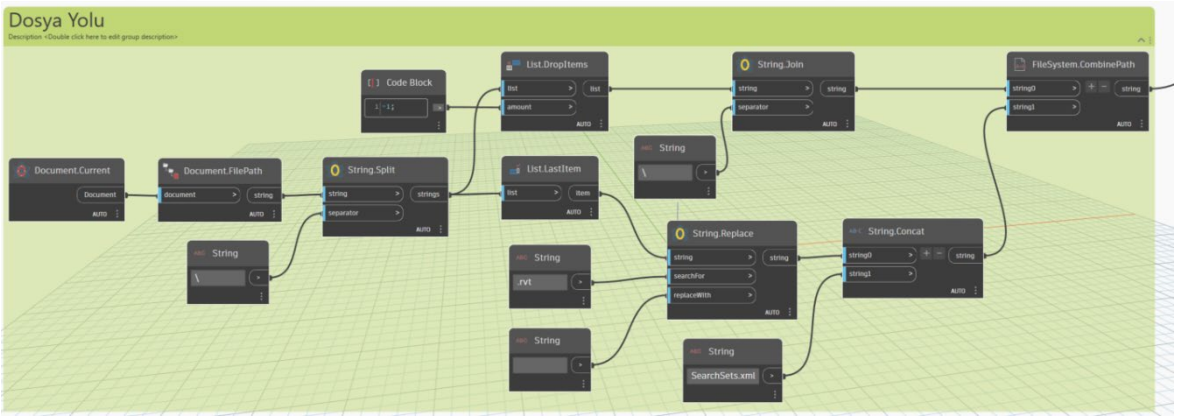


Şekil 64. Search Sets oluşturulmasındaki kurallar ve kodlamalar

Ardından, oluşturulan “Search Sets” kümelerinin Navisworks yazılımına aktarılması için “.xml” formatında bir dosyanın dış ortama kaydedilmesi gerekmektedir. Bu amaçla hazırlanmış olan Python kodu, Navisworks’ün beklentilerine uygun biçimde “.xml” uzantılı ve doğru biçimlendirilmiş bir dosya üretmeye odaklanmaktadır. Şekil 65’te, söz konusu Python kodunun genel yapısı gösterilmiştir. Kod, iki farklı veri girişi beklemektedir. İlki “NavisworksCreateSearchSet” düğümünden elde edilen liste, diğeri ise “.xml” dosyasının saklanacağı dizin ve dosya adı bilgileridir. Kullanıcı dostu bir yaklaşım benimsenerek, “.xml” dosyası 3B BIM modelinin bulunduğu dizinde, model adı ile “SearchSets” ifadesinin birleştirilmesi şeklinde kaydedilmektedir. Örneğin, masaüstünde “Proje1.rvt” adlı Revit modeli üzerinde çalışıldığı varsayılırsa, “.xml” dosyası aynı konumda “Proje1SearchSet.xml” adıyla oluşturulmaktadır. Şekil 66, bu dosya yolunun ve dosya adının tanımlanması sürecini daha ayrıntılı biçimde ortaya koymaktadır.



Şekil 65. .xml uzantılı dosya oluşturulması için kullanılan Python kodu

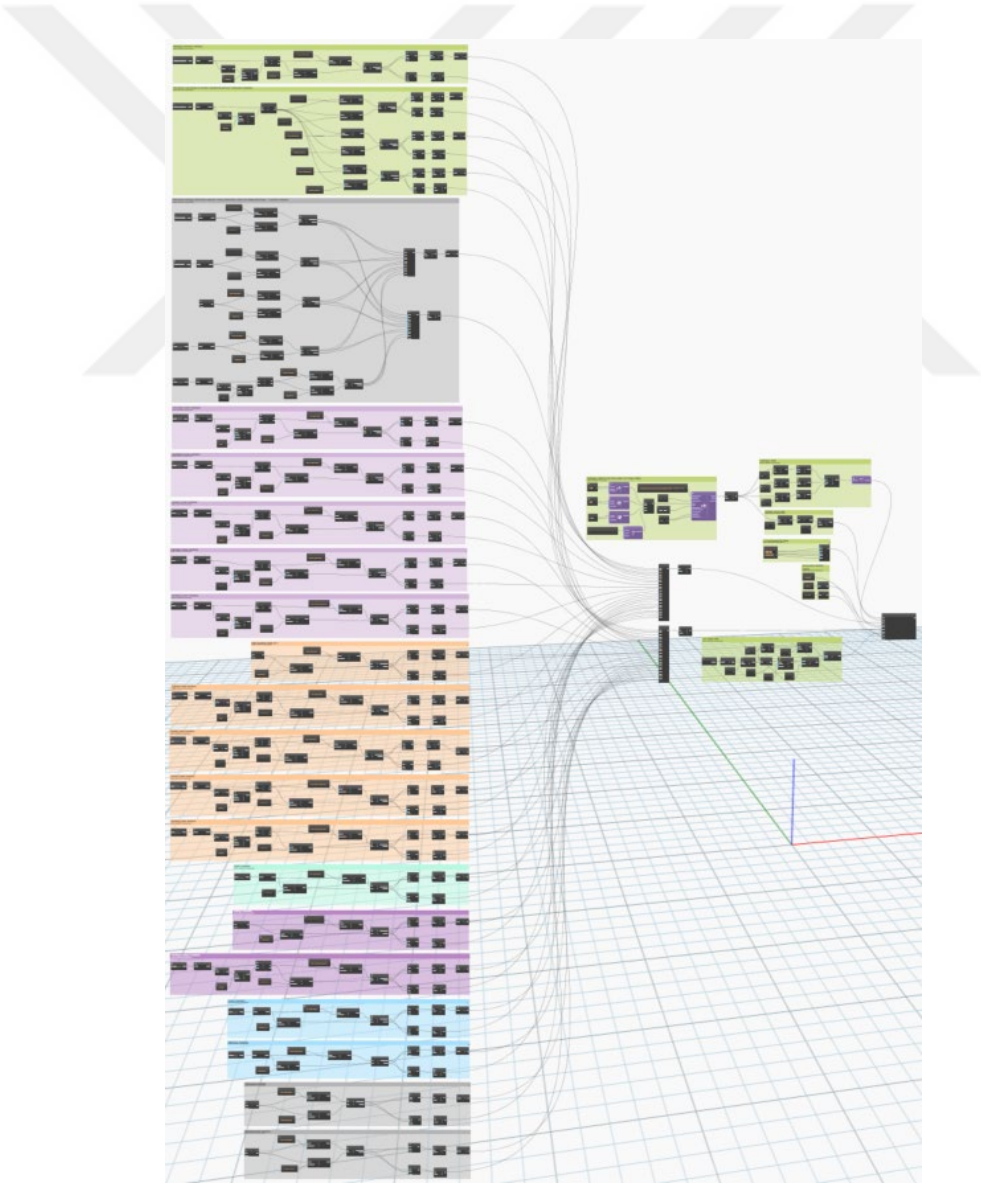


Şekil 66. .xml uzantılı dosya yolunun ve adının Dynamo kodları ile tanımlanması

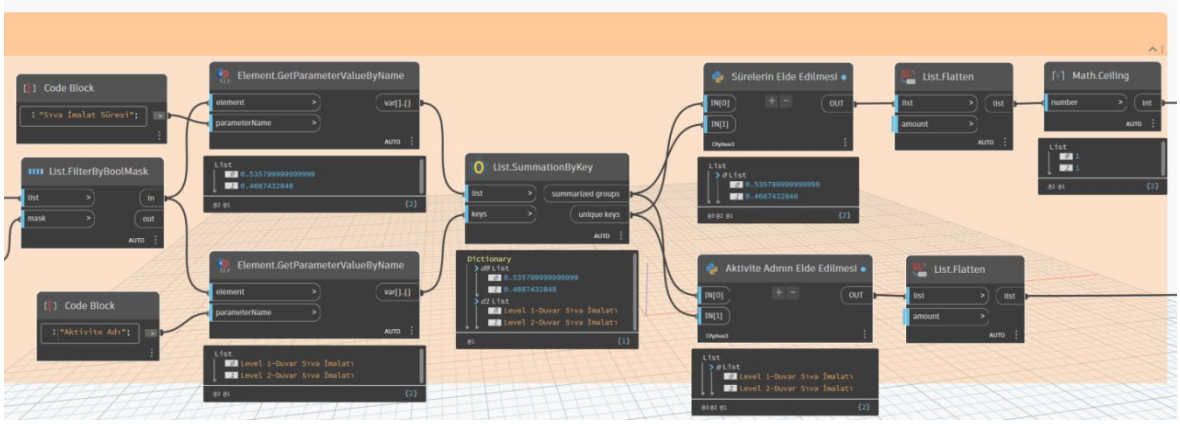
2.4.9. ML Model Çıktısı ile İlişkilendirme ve İş Programının Elde Edilmesi

Bu aşama, tez kapsamında geliştirilen tüm Dynamo kodlarının sonuncusunu oluşturmaktadır. Söz konusu kodlama adımında, .csv formatında nihai bir iş programı dosyası üretilerek ML modelinden elde edilen çıktıların iş programıyla bütünleştirilmesi sağlanmaktadır. Şekil 67'de genel Dynamo kod yapısı gösterilmiştir. İlk olarak,

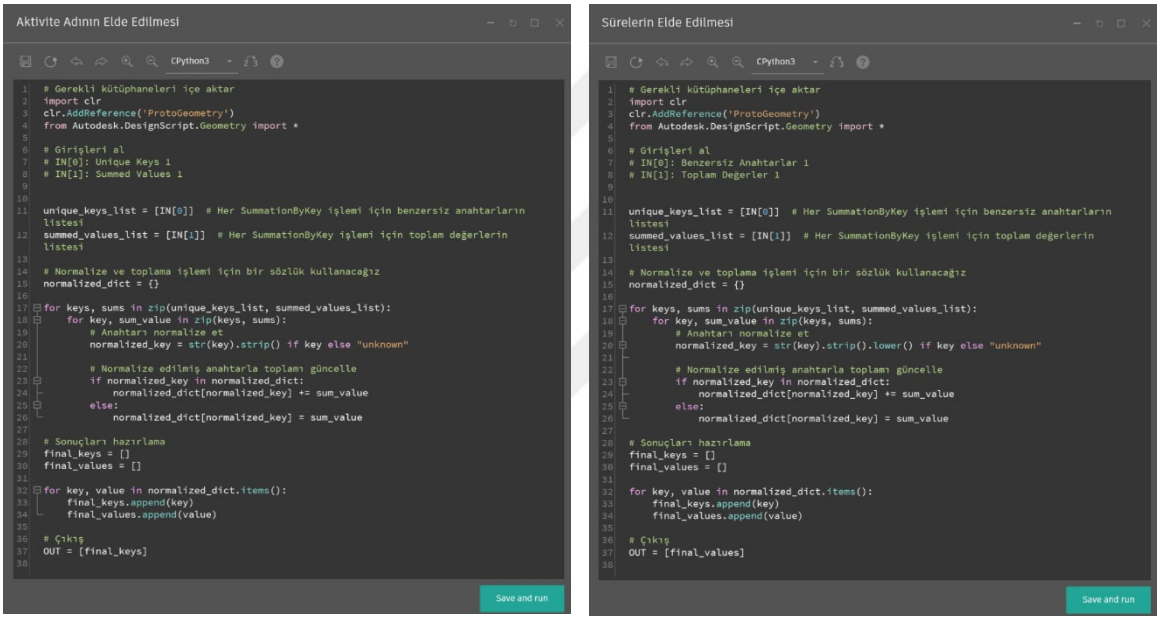
“List.SummationByKey” düğümü aracılığıyla, belirli parametrelere (örneğin, kat seviyesi, eleman adı vb.) göre gruplandırılan listeler üzerindeki değerler toplanmıştır. Bu yöntem, birden çok elemandan oluşan listelerde aynı parametreye sahip öğelerin değerlerinin ortak bir bütün halinde ele alınmasına imkân tanımaktadır. Böylece aynı katta bulunan elemanların imalat süreleri bir araya getirilerek aktivite-süre ilişkisi net bir şekilde kurgulanmaktadır. Şekil 68’de, örnek bir Revit modelindeki sıva elemanı üzerinden bu işlemler ve ilgili düğümlerden elde edilen sonuçlar gösterilmektedir. “List.SummationByKey” düğümünden çıkan veriler, Şekil 69’da gösterilen Python kodları yardımıyla aktivite adları ve süreleri olarak ayrıştırılmakta ve “Math.Ceiling” düğümü kullanılarak aktivite süreleri üst tam sayıya yuvarlanmaktadır.



Şekil 67. İş programının elde edilmesi için oluşturulan Dynamo kodu



Şekil 68. Aktivite süresi ve adının ilişkilendirilmesi Dynamo kodu

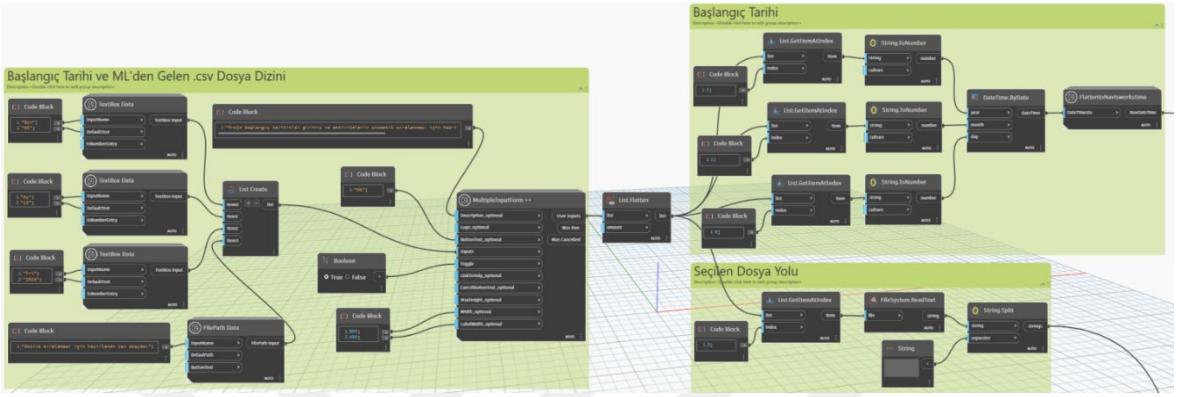


(a) Aktivite adı (b) Aktivite süresi

Şekil 69. (a) Aktivite adı ve (b) sürelerinin ayrılması için oluşturulan Python kodları

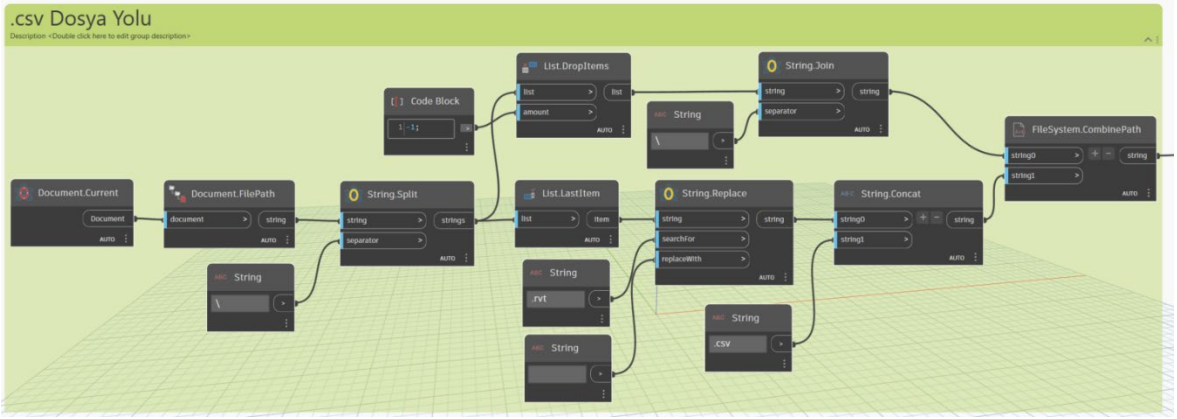
Elde edilen aktiviteler ve süreler ilave bir Python kodu ile ML'den elde edilen aktivite sıralamalarına göre sıralanarak dış ortama aktarılmak üzere düzenlenmektedir. Ek 8'de, söz konusu Python kodu sunulmuş olup, dokuz ayrı veri girişi gereksinimi olduğu görülmektedir. Bu girdilerin ikisi, yukarıda belirtilen aktiviteler ve sürelerdir. İş programının oluşturulabilmesi için, proje başlangıç tarihinin kullanıcı tarafından belirlenmesi önem taşımaktadır. Bu amaçla oluşturulan bir kullanıcı arayüzü üzerinden proje başlangıç tarihi bilgisi alınmakta ve Python koduna veri olarak aktarılmaktadır. Aynı arayüzde, ML yönteminden elde edilen aktivite sıralamasının bulunduğu .csv dosyasına ilişkin dosya yolu da istenmektedir. Kullanıcı, açılan pencereden söz konusu .csv dosyasını

seçmekte ve bu dosya Python koduna ek veri girişi olarak tanımlanmaktadır. Python kodu Revit'ten oluşturulan aktivite isimlerini, ML'den gelen aktivite sıralamasına göre sıralamaktadır. Bu işlemleri gerçekleştirmeye yönelik Dynamo kodu, Şekil 70'de gösterilmektedir.



Şekil 70. İş programı başlangıç tarihi ve ML'den elde edilen aktivite sıralama dosyası seçimi Dynamo kodu

Bir diğer veri girişi, Navisworks ile uyumlu iş programı dosyası oluşturulması için .csv dosyasının ilk satır başlıklarını içermektedir. Böylece, 4B simülasyon oluşturulurken doğru başlıkların kullanılması ve aktivitelerin düzgün bir şekilde tanınması sağlanmaktadır. Ayrıca, kullanıcıya daha net bir görsel sunmak ve farklı tipteki aktiviteleri ayırt edebilmek amacıyla, "kalıp" ile ilgili aktivitelere "Temporary", diğer tüm aktivitelere "Construct" atanmıştır. Söz konusu tip bilgileri de Python koduna ek veri girişi olarak alınmaktadır. Son olarak, oluşturulacak iş programı dosyasının kaydedileceği dizin ve dosya adı belirtilmektedir. Bu aşamada, Şekil 71'de gösterildiği üzere, 3B BIM modelinin bulunduğu konumda, model adına ".csv" uzantısı eklenerek çıktı alınmaktadır. Örneğin, "Proje1.rvt" isimli bir Revit modeli ile çalışılıyorsa, "Proje1.csv" adlı iş programı dosyası aynı dizinde oluşturulmaktadır.

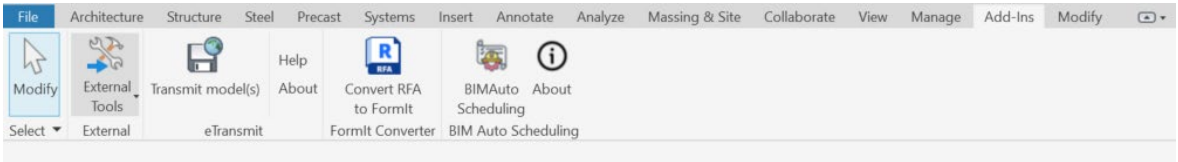


Şekil 71. İş programının dosya yolunun belirlenmesi Dynamo kodu

2.5. Revit Eklentisi Oluşturulması

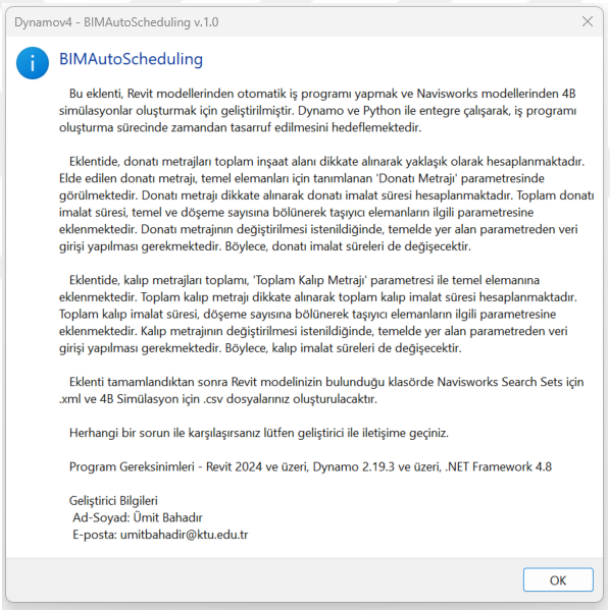
Manuel olarak Dynamo kodlarını sırasıyla açıp çalıştırmak, büyük ve karmaşık projelerde hem zaman kaybına hem de olası kullanıcı hatalarına yol açabilmektedir. Ayrıca, söz konusu Dynamo kodlarının farklı kullanıcılar tarafından tutarlı ve doğru biçimde kullanılmasını sağlamak da çoğu zaman zorlu bir süreçtir (Wang & Rezazadeh Azar, 2019). Bu nedenlerle, Revit ortamında tek bir tuşla otomatik olarak tüm Dynamo kodlarının çalıştırılmasını ve böylelikle 4B simülasyon verisinin de hızlı bir biçimde oluşturulmasını sağlayacak bir eklenti geliştirilmesine karar verilmiştir.

Geliştirme sürecinde, Visual Studio 2022 kullanılarak C# programlama dili ile "BIM Auto Scheduling" adında bir Revit eklentisi oluşturulmuştur. Eklentinin kullanıcı arayüzü, Revit içerisinde "Add-Ins" sekmesi üzerinde oluşturulan "BIM Auto Scheduling" adlı panelde yer almaktadır. Söz konusu panelde, "BIM Auto Scheduling" ve "About" olmak üzere iki buton bulunmaktadır. Butonlara ait simgeler, açıklamalar ve bu butonların eklenti içerisinde hangi işlevleri yerine getireceği, Revit API'nin ilgili sınıfları kullanılarak tanımlanmıştır. Şekil 72'de eklentinin Revit arayüzünde görünümü verilmektedir.



Şekil 72. Revit menüsünde BIM Auto Scheduling eklentisi

- BIM Auto Scheduling Butonu: Bu buton tıklandığında, proje kapsamında daha önce oluşturulmuş olan dokuz adet Dynamo kodu, önceden belirlenen sırayla otomatik olarak çalıştırılmaktadır. Özellikle büyük modellerde Dynamo'nun çalışması performans açısından zaman alabileceğinden, bu işlemi eklenti üzerinden yürütmek, kullanıcı müdahalelerini en aza indirgemekte ve işlem sürelerini kısaltabilmektedir. Dokuz kodun tamamı başarıyla çalıştırıldıktan sonra, kullanıcıya işlemlerin başarıyla tamamlandığına dair bir bilgilendirme yapılmaktadır.
- About Butonu: Bu buton, kullanıcıların eklentiye ilişkin detaylı bilgilere erişebileceği bir arayüz sunmaktadır. Eklentinin hangi amaçlarla ve hangi kapsamda geliştirildiği, kullanıcının nelere dikkat etmesi gerektiği gibi bilgilendirmeler burada yer almaktadır. Şekil 73'te oluşturulan bilgilendirmeler gösterilmektedir.



Şekil 73. About butonu ile oluşturulan bilgilendirme ekranı

Ek 9'da yer alan C# kodu incelendiğinde, eklentinin hangi sekme ve panel üzerinde yer alacağına ilişkin tanımlamaların yanı sıra, buton ikonlarının, açıklamalarının ve tıklandığında çalıştırılacak sınıfların nasıl düzenlendiği açıkça görülebilmektedir. Ayrıca, "BIM Auto Scheduling" butonuna bağlı komut sınıfları ile "About" butonunda yer alan bilgilendirme sınıfının detayları da bu kod içerisinde yer almaktadır.

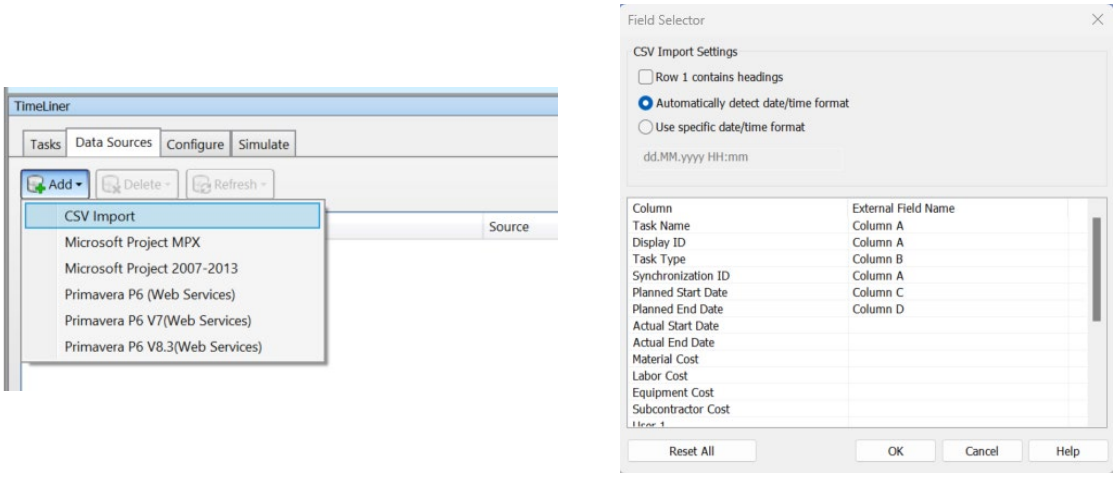
Geliştirilen bu eklenti sayesinde, Revit arayüzünde tek noktadan erişilebilen iki buton aracılığıyla Dynamo kodlarının sırasıyla ve otomatik olarak çalıştırılması sağlanmıştır. Böylece, kullanıcı hatalarını ve zaman kaybını azaltmanın yanı sıra,

projelerde tekrarlanabilir bir iş akışı elde edilmiştir. Bu yaklaşım, Revit içerisinde otomasyon ve veri bütünlüğü açısından önemli bir katkı sağlamaktadır.

2.6. 4B Model ve Simülasyon Oluşturulması

Bu aşamada, daha önceki bölümlerde oluşturulan 3B BIM modeli ile .xml uzantılı "Search Sets" ve .csv uzantılı iş programı dosyalarının Navisworks ortamında ilişkilendirilmesi yoluyla bir 4B model ve simülasyon oluşturulması hedeflenmiştir. Bu süreçte, Navisworks yazılımına iş programının aktarılması ve ilgili yapı elemanlarıyla eşleştirilmesi işlemlerinin otomatikleştirilmesi amaçlanmaktadır. Model ve aktivite sayıları arttıkça, Navisworks üzerinde gerçekleştirilen manuel eleman-aktivite eşleştirme süreci zaman alıcı olmakta ve hata riski yükselmektedir (Patil vd., 2017). Bu gereksinimden yola çıkılarak, Visual Studio ortamında C# programlama dili kullanılarak "BIM Auto Scheduling - Import CSV" adlı bir eklenti geliştirilmiştir.

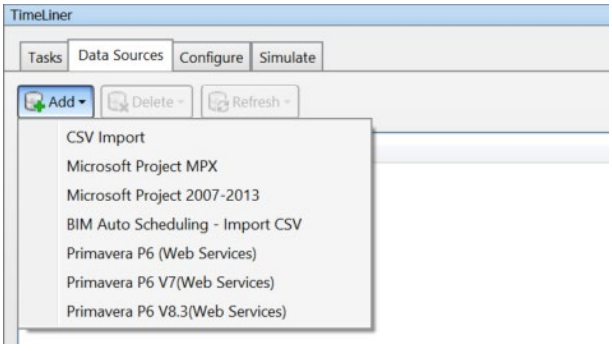
Navisworks yazılımında iş programı girişi, "Timeliner" penceresindeki "Data Sources" bölümü üzerinden gerçekleştirilmektedir. Burada, "Add" seçeneği kullanılarak Primavera, MS Project, Excel gibi farklı yazılımlarda hazırlanan iş programı dosyasının türü belirlenmektedir. Dosya türü seçildikten sonra, açılan pencerede ilgili dosyanın konumu belirtilmekte ve Navisworks ile doğru ilişki kurulması için hangi kategoride hangi verinin bulunduğu tanımlanmaktadır. Bu aşama, özellikle karmaşık iş programları söz konusu olduğunda hatalı sütun seçimlerine bağlı olarak veri aktarımında olası hata riski artabilmektedir (Fazeli vd., 2024). Şekil 74'te Navisworks'ün desteklediği dosya türleri ve seçilmesi gereken bilgiler gösterilmektedir.



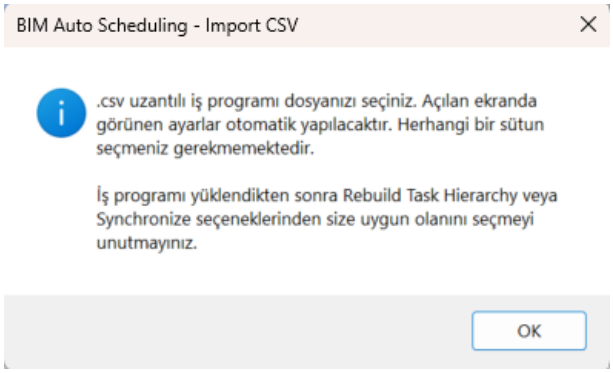
(a) Dosya türleri (b) Seçilmesi gereken sütunlar

Şekil 74. Navisworks'te bulunan iş programı (a) dosya türleri ve (b) gerekli kategoriler

Geliştirilen "BIM Auto Scheduling - Import CSV" eklentisi, Revit'teki gibi üst panelde yer alan bir sekme üzerinden değil, "Timeliner > Data Sources > Add" menüsündeki dosya türü seçimi bölümünde konumlandırılmıştır. Şekil 75'te söz konusu eklentinin Navisworks arayüzündeki görünümü sunulmaktadır. Burada "BIM Auto Scheduling - Import CSV" seçeneği işaretlendiğinde, daha önce elde edilmiş .csv uzantılı iş programı dosyası sisteme eklenmektedir. C# ile geliştirilen bu eklentide hangi bilgilerin hangi sütunda yer alacağı önceden tanımlandığından, açılan pencerede ek bir ayar yapılması gerekmemektedir. Bu durum Şekil 76'da gösterildiği üzere, eklenti seçildiğinde kullanıcıya sunulan bilgilendirme ekranında da açıkça belirtilmektedir. Ayrıca eklenti, iş programında tanımlı aktiviteler ile "Search Sets" kullanılarak gruplandırılan BIM modeli elemanlarını otomatik biçimde eşleştirmektedir. Böylece, eleman-aktivite eşleştirmesinde ortaya çıkabilecek sorunlar en aza indirilmekte ve önemli ölçüde zaman tasarrufu sağlanmaktadır. Eklentinin C# kodu Ek 10'da ayrıntılı biçimde sunulmuştur.

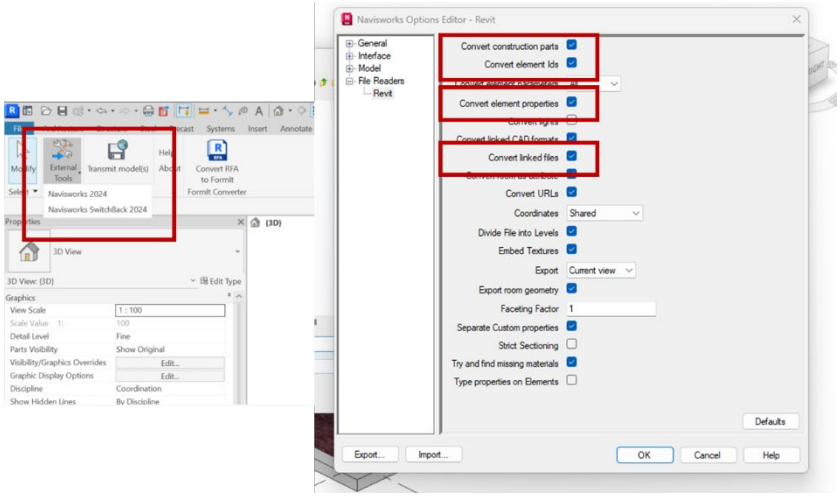


Şekil 75. Timeliner penceresinde BIM Auto Scheduling - Import CSV eklentisi



Şekil 76. BIM Auto Scheduling - Import CSV eklentisi bilgilendirme ekranı

Eklentinin sağlıklı biçimde çalışabilmesi için, Revit'te oluşturulan BIM modelinin Navisworks ortamına aktarımında dikkat edilmesi gereken bazı hususlar bulunmaktadır. Bu kapsamda, Revit üzerinden "External Tools" altında yer alan "Navisworks 2024" butonu kullanılarak modelin dış ortama aktarılması önerilmektedir. İlgili aktarım penceresinde, dosyanın .nwc uzantısıyla kaydedilmesi ve "Convert construction parts", "Convert element Ids", "Convert element properties", "Convert linked files" seçeneklerinin etkinleştirilmesi gerekmektedir. Bu sayede, model elemanlarının kimlik ve özellik bilgileri, olası inşaat parçaları ile bağlantılı dosyalar korunarak Navisworks'e daha kapsamlı ve veri zenginliği içeren bir biçimde aktarılmaktadır. Böylelikle koordinasyon, analiz ve çakışma tespiti gibi süreçlerin daha bütüncül yürütülmesi mümkün olmaktadır. Şekil 77'de, dış ortama aktarımın nasıl gerçekleştirildiğine ilişkin bir örnek gösterilmektedir.



Şekil 77. Revit'ten Navisworks'e BIM modeli aktarım süreci

Eklentinin çalıştırılmasından ve iş programı dosyasının Navisworks'e aktarılmasından önce, "Search Sets" dosyasının da Navisworks ortamına eklenmesi gerekmektedir. Bu işlem manuel olarak gerçekleştirilmekte; daha önce BIM modelinden oluşturulan dosya, "Import Search Sets" butonu kullanılarak Navisworks'e yüklenmektedir.

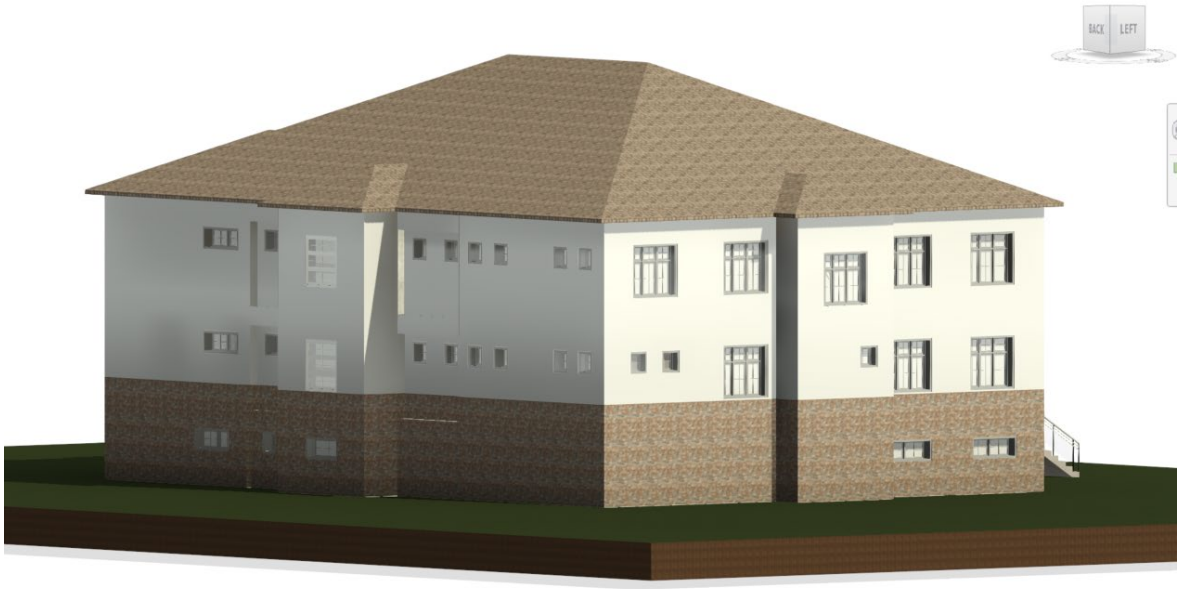
Belirtilen iki aşama (yani BIM modelinin Revit'ten dış ortama aktarılıp Navisworks'te açılması ve "Search Sets"'in Navisworks'e aktarımı) tamamlandıktan sonra, "BIM Auto Scheduling - Import CSV" eklentisi aracılığıyla iş programı aktarılmakta, model elemanları ile otomatik olarak ilişkilendirilmekte ve 4B BIM modeli elde edilmektedir. Sonrasında ise "Timeliner > Simulate" seçeneği üzerinden 4B simülasyon videosu oluşturularak yapının zaman çizelgesi boyunca nasıl inşa edileceği görselleştirilebilmektedir. Bu uygulamaya ait detaylı görseller ve örnek proje üzerinden yapılan çalışmalar, ilerleyen bölümlerde kapsamlı bir biçimde ele alınacaktır.

2.7. Örnek Proje

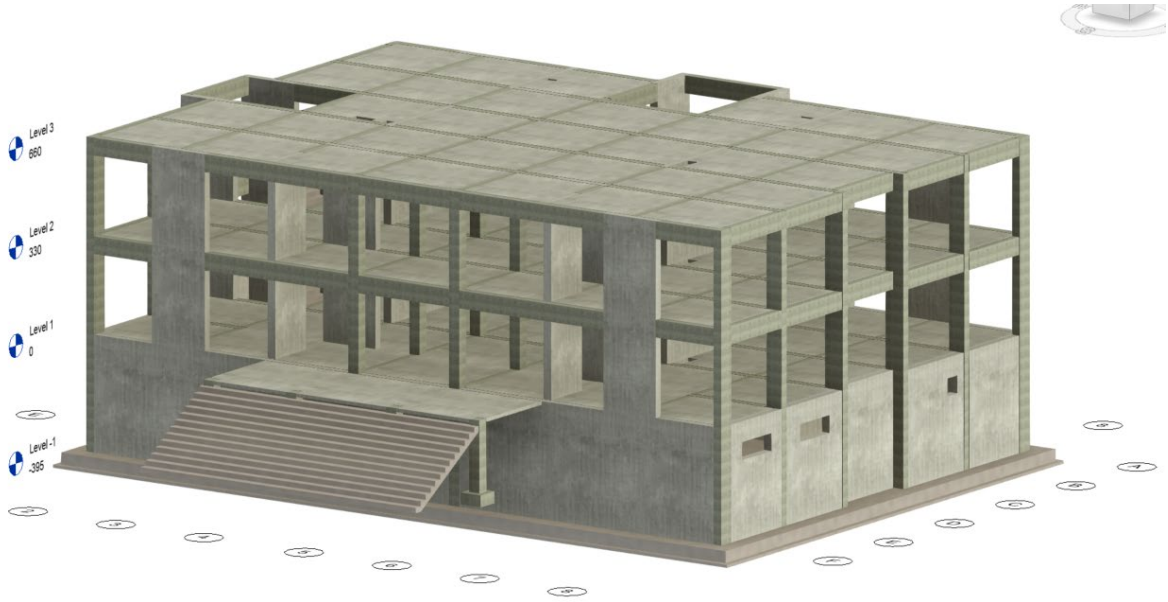
Bu bölümde, tez çalışması kapsamında geliştirilen iş programının otomatikleştirilmesi yaklaşımının test edilmesi amacıyla bir örnek proje hazırlanmıştır. Söz konusu örnek projeye ait BIM modeli, Autodesk Revit 2024 yazılımı kullanılarak oluşturulmuştur. Revit ortamında oluşturulan BIM modelindeki yapı elemanları, ayrıntılı tasarım aşamasını temsil eden LOD 300 seviyesindedir. Ayrıca, örnek projenin BIM modeli tasarlanırken, "2.3. 3B BIM Modelinin Oluşturulması" başlığı altında belirtilen tasarım kurallarına özenle uyulmuştur.

Örnek proje, Trabzon ilinde inşa edilen bir anaokulu binasıdır. Bina, bodrum kat, zemin kat ve bir normal kat olmak üzere toplam üç kattan oluşmaktadır. Bu proje seçilirken, tez kapsamında belirlenen tüm yapı elemanları ve aktivitelere yer verilmesine dikkat edilmiştir. Revit ile modellenen örnek projeye ait 3B model, 3B yapısal model ve güney cephe görünümleri Şekil 78-81'da sunulmaktadır. Ayrıca, Şekil 82 ve 83'te kullanılan elemanların örnek olarak gösterilmesi açısından temel kalıp ve zemin kat mimari planları gösterilmektedir. Diğer katlara ait kalıp planları ile mimari planlar ise Ek 11'de verilmektedir. Tablo 9'da kat yükseklikleri ve alanları gösterilmektedir.

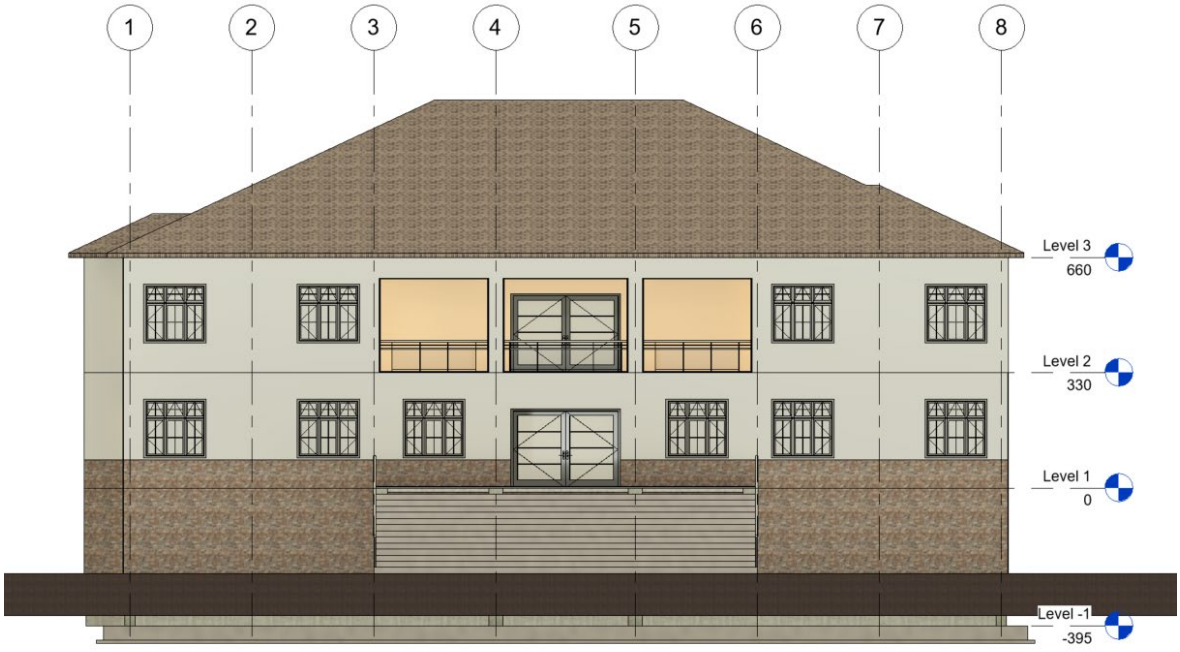
Şekil 78. 3B model güneydoğu görünümü



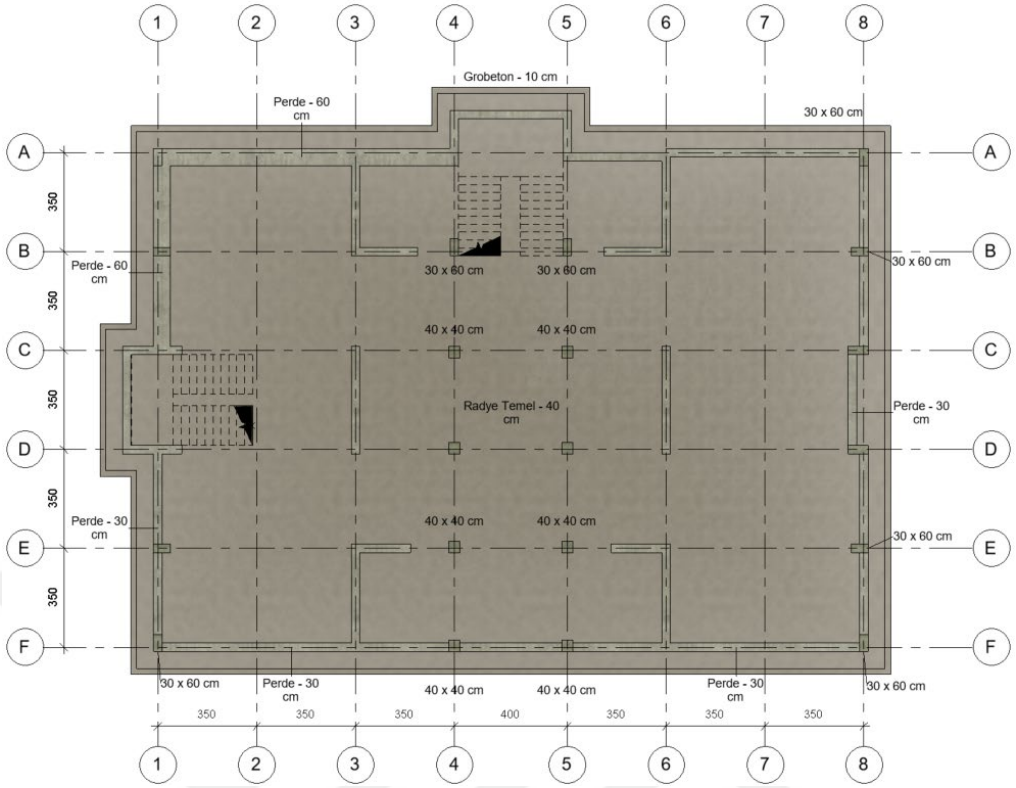
Şekil 79. 3B model kuzeybatı görünümü



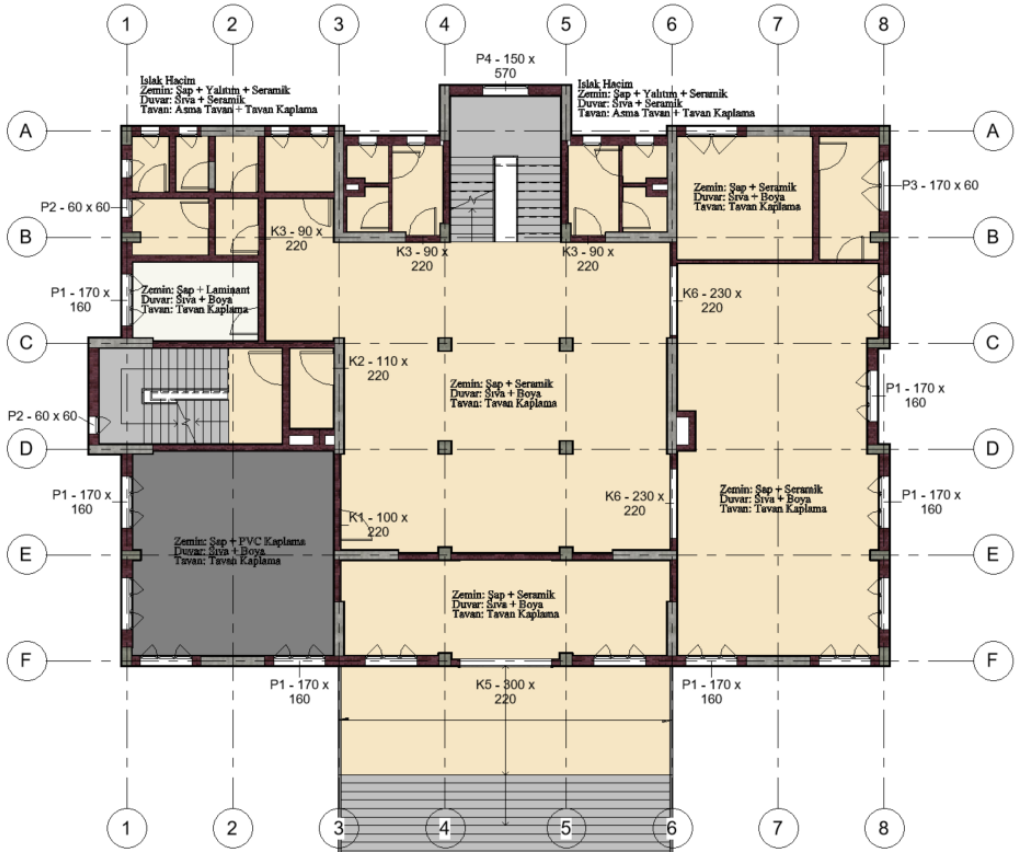
Şekil 80. 3B yapısal model



Şekil 81. Güney cephe görünümü



Şekil 82. Temel kalıp planı



Şekil 83. Zemin kat mimari plan

Tablo 9. Örnek proje kat yükseklikleri ve alanları

Katlar	Yükseklik (m)	Alan (m^2)
Bodrum Kat	3.95	458.60
Zemin Kat	3.30	497.70
1. Normal Kat	3.30	458.60

Bu örnek proje, Autodesk Revit ortamında modellenmesini takiben "BIM Auto Scheduling" eklentisi çalıştırılmadan önce, ML yaklaşımıyla elde edilen aktivite sıralamasının projeye uygun biçimde düzenlenmesini gerektirmektedir. Bu düzenleme aşamasında, ML algoritmaları ile gerçekleştirilen test projelerindeki kat sayısının, modellenen bina kat sayısına eşit veya benzer olması, eklenti sonunda daha tutarlı ve isabetli sonuçlar elde edilmesini mümkün kılmaktadır.

Bununla birlikte, ML algoritmalarının açıklandığı ilgili bölümde de belirtildiği üzere, modellenen binayla aynı kat sayısına sahip test projelerinin bulunmadığı durumlarda dahi ne ML algoritmaları ne de eklenti hata vermemektedir. ML algoritmalarına eklenen ek kodlamalar sayesinde, aktivite sıralaması .csv formatında dış ortama aktarılmadan önce, kullanıcıdan "Başlangıç Aktivitesi", "Normal Kat Sayısı" ve "Bodrum Kat Sayısı" bilgileri talep edilmekte, böylece 3B modelin kat sayısı ile uyumlu bir aktivite sıralaması dosyası oluşturulmaktadır.

Örnek proje için ML algoritmalarından elde edilen sonuç dış ortama aktarılırken aşağıdaki değerler seçilmiştir:

- Başlangıç Aktivitesi: Level -1-Grobeton İmalatı
- Normal Kat Sayısı: 2
- Bodrum Kat Sayısı: 1

ML algoritmalarının doğruluğu, daha önce seçilmiş test projeleri üzerinden değerlendirilmektedir. Bu test projeleri, örnek anaokulu binası ile tam olarak aynı kat sayısına sahip olmasa da benzer kat sayılarına sahip olması gözetilerek "P1, P4, P9, P10, P15" olarak seçilmiştir. Bu sayede, oluşturulan aktivite sıralaması dosyasının anaokulu projesi ile aynı aktivitelere karşılık gelecek şekilde düzenlenmesi sağlanmıştır.

3. BULGULAR VE İRDELEME

Bu bölümde, tez kapsamında gerçekleştirilen çalışmanın tüm bileşenleri bir bütün olarak ele alınarak değerlendirilmekte ve literatürdeki benzer araştırmalarla karşılaştırmalı analizler sunulmaktadır. Bu bağlamda; ML modellerinden elde edilen aktivite sıralama dosyalarının tutarlılık ve doğruluk düzeyleri, Autodesk Revit için geliştirilen eklenti ile desteklenen Dynamo kodlarının otomasyon potansiyeli, eklenti sonunda oluşturulan iş programı dosyasının proje yönetimine katkıları ve Navisworks platformu için geliştirilen eklentinin ve hazırlanan 4B simülasyon çıktılarının sağlayabileceği öngörüler irdelenmektedir. Söz konusu inceleme, hem model tabanlı yaklaşımların entegrasyon düzeyini hem de elde edilen sonuçların pratik uygulamalara aktarılabilirliğini ortaya koymayı hedeflemekte ve böylece proje planlama ve yönetim süreçlerine yenilikçi bir perspektif kazandırmaktadır.

3.1. ML Modellerinin Karşılaştırılması

Bu bölümde ilk aşamada geliştirilmiş olan ML modellerinin performansları nicel ölçütlerle değerlendirilmiştir. Veri hazırlık süreci sonucunda, 24 projeden derlenen ve toplam 1824 aktivite-ardıl ilişkisini içeren veri seti, bu modellerin eğitim ve test aşamaları için temel oluşturmuştur. Daha önceki bölümlerde de belirtildiği gibi, ilgili modellerin hiper parametre optimizasyonunun doğru biçimde gerçekleştirilmesi hem modelin eğitimi hem de geçerliliği açısından son derece önemlidir. Bu doğrultuda, 24 proje arasından 19'u eğitim ve 5'i test veri seti olarak ayrılmıştır. Test grubunu oluşturan projelerden P1, P4, P10 ve P15'in yanı sıra, daha az aktiviteye sahip olan ve dolayısıyla daha sınırlı bir ardıl tahmini gerektiren P16, parametre belirleme aşamasında test grubuna dahil edilmiştir.

Model eğitimi ve parametre seçimi sırasında, GNN tabanlı yaklaşımlar (GCN, GAT, GraphSAGE) ile rastgele yürüyüş temelli yöntemler (node2vec ve DeepWalk) için kendi aralarında aynı parametre yapılandırmalarının kullanılması hedeflenmiştir. Bu sayede, benzer altyapıya sahip modellerin kendi içlerindeki ve birbirleriyle olan performans farklılıkları daha adil bir şekilde değerlendirilebilmiştir. Bütün modeller için iterasyon

sayısı (number of epochs) 500 ve öğrenme oranı (learning rate) 0.01 düzeyinde sabit tutulmuştur.

Tablo 10'da bu çalışma kapsamında değerlendirilen ML modellerine ait parametre seçimleri yer almaktadır. GNN tabanlı yöntemler (GCN, GAT ve GraphSAGE) için 5 farklı gizli katman boyutu (hidden size) ve yerleştirme boyutu (embedding size) kombinasyonu, ARM modeli için 11 farklı minimum destek (support) ve minimum güven (confidence) eşleşmesi ve node2vec ile DeepWalk yöntemleri için sırasıyla 32 ve 19 farklı parametre seti denenmiştir. Her bir modelin ilgili veri kümesi üzerindeki başarımı, söz konusu parametre kümeleri arasından en uygun ayarların belirlenebilmesi amacıyla karşılaştırmalı olarak incelenmiştir. Parametreler, literatürde benzer ML yöntemlerini ele alan çalışmalar dikkate alınarak tasarlanmıştır (Cheng vd., 2015; Kim vd., 2018; Lee vd., 2019; Pan vd., 2020; Mostofi vd., 2022; Mostofi & Toğan, 2023; Mostofi vd., 2024; Sarmah vd., 2024). Parametre optimizasyonu aşamasının temel amacı, aktivite-ardıl tahmini probleminde her bir yaklaşım için en uygun ayarların belirlenmesi ve böylece model performansının bütüncül biçimde ortaya konmasıdır.

Tablo 10. ML modelleri için kullanılan parametreler

ML Modeli	Gizli Katman Boyutları (Hidden Sizes)	Yerleştirme Boyutları (Embedding Dimensions)
GCN & GAT & GraphSAGE	16	16
	32	32
	64	64
	128	128
	256	256

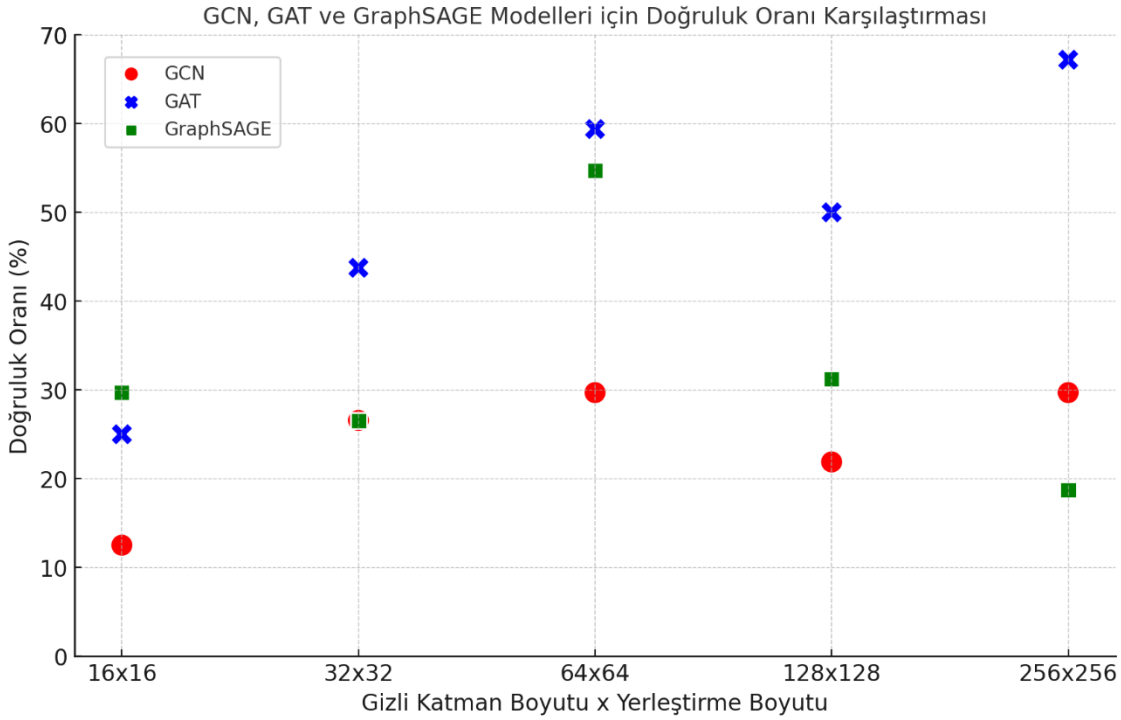
	Yerleştirme Boyutları (Dimension)	Yürüyüş Uzunlukları (Walk Length)	Yürüyüş Sayıları (Number of Walks)	Workers	p	q
node2vec	64	10	5	1	0.5	
	128	20	10	2	1	
	256	40	20	4	2	
		80	50			
		100				

Tablo 10'nun devamı

DeepWalk	64	10	5	1	1
	128	20	10	2	
	256	40	20	4	
		80	50		
		100			

	Minimum Destek (Minimum Support)	Minimum Güven (Minimum Confidence)	Lift
ARM	0.1	0.7	<1
	0.05	0.7	
	0.01	0.7	
	0.01	0.3	
	0.01	0.1	
	0.005	0.5	
	0.002	0.5	
	0.001	0.5	
	0.001	0.1	
	0.001	0.05	
	0.0005	0.05	

GCN, GAT ve GraphSAGE modellerinin, 64 aktivite üzerinden gerçekleştirilen tahmin sonuçları Şekil 84'te karşılaştırmalı olarak sunulmuştur. İlgili grafik, her bir gizli katman boyutu ve yerleştirme boyutu için elde edilen doğruluk oranlarını ve tahmin edilen doğru aktivite sayısını içermektedir. Bu üç GNN tabanlı yaklaşım arasında GAT, 256×256 boyutlu parametre setiyle %67,19 (43 doğru tahmin) doğruluk oranı sunarak en yüksek isabeti yakalamıştır. Orta ölçekli (64×64) parametre değerlerinde de GAT %59,38 (38 doğru tahmin) doğruluk oranına ulaşarak diğer iki modeli geride bırakmıştır. GCN ise 64×64 ve 256×256 boyutlarında %29,69 (19 doğru tahmin) değerine ulaşarak kendi içinde daha yüksek bir performans gösterse de mutlak düzeyde diğer yöntemlerin gerisinde kalmıştır. GraphSAGE, 64×64 boyutuyla %54,69 (35 doğru tahmin) oranında görece başarılı sonuç elde etmiş; ancak 256×256 boyutunda performansının %18,75 (12 doğru tahmin) seviyesine kadar düştüğü gözlenmiştir.

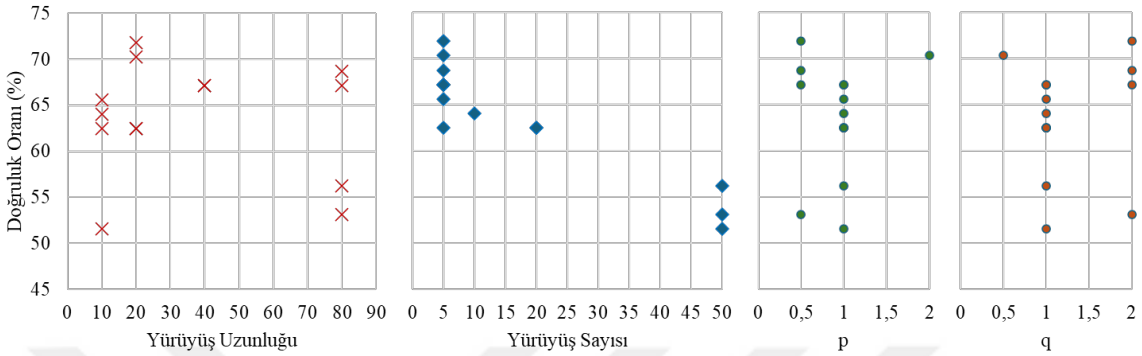


Şekil 84. GNN tabanlı yöntemlerin parametre değişimlerinin karşılaştırılması

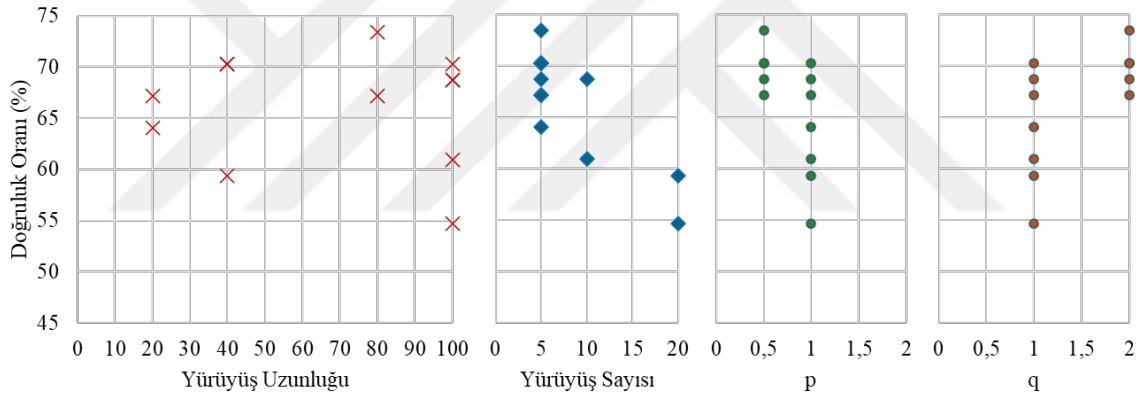
Parametre boyutunun artırılması, bazı modeller için öğrenme kapasitesini geliştirebilirken (GAT örneğinde görüldüğü gibi), diğer modellerde aşırı karmaşıklığa neden olarak performansın düşmesine yol açabilmektedir. Dolayısıyla, GNN modelleri arasında tek bir "en iyi" parametre setinden bahsetmek yerine, modelin yapısı, veri kümesinin büyüklüğü ve tahmin edilmesi beklenen aktivite sayısı gibi faktörler göz önünde bulundurulmalıdır. Bu kapsamda yürütülen analizler sonucunda, GAT için 256×256, GraphSAGE için 64×64, GCN için ise 64×64 ve 256×256 parametre setlerinin diğer projeler üzerinde daha detaylı biçimde test edilmek üzere seçilmesi kararlaştırılmıştır. Böylece, ilgili modellerin farklı ölçek ve karmaşıklıktaki veri kümelerinde bu parametre kombinasyonları ile gösterecekleri performansın derinlemesine incelenmesi hedeflenmektedir.

Şekil 85'te, node2vec yönteminin yerleştirme boyutunun (64 ve 128) değiştirilmesiyle birlikte farklı yürüyüş uzunluğu (walk length), yürüyüş sayısı (number of walks), p ve q değerleri üzerinden yapılan denemeler sonucu elde edilen doğruluk oranları sunulmaktadır. Şekil 85'teki tüm koşullarda iş parçacığı sayısı (workers) değeri 2 olarak sabitlenmiştir. Ayrıca, 256 boyutlu yerleştirme ve iş parçacığı sayısının 1 ve 4 parametrelerine eşit olduğu durumlara yönelik ek denemeler yapılmış olmasına rağmen, bu

ayarların daha düşük performans göstermesi nedeniyle ilgili sonuçlar grafiğe yansıtılmamıştır. Böylece, yöntemin farklı parametre kombinasyonlarında nasıl bir doğruluk yakalayabildiği daha sade bir şekilde izlenebilir hâle gelmektedir.



(a) 64 yerleştirme boyutlu



(b) 128 yerleştirme boyutlu

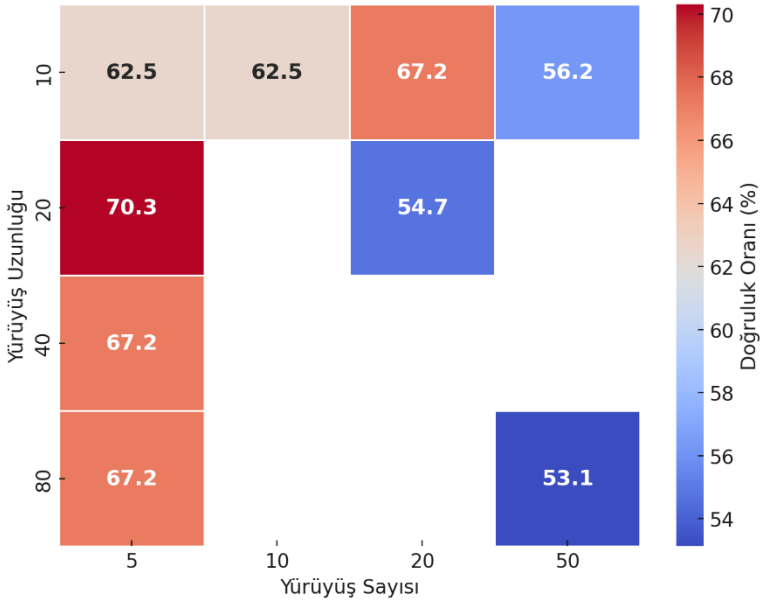
Şekil 85. Node2vec yöntemi parametre değişimleri (a) 64 ve (b)128 yerleştirme boyutlu

Şekil 85(a)'da görülen kombinasyonlar, yerleştirme boyutu 64 olduğunda özellikle yürüyüş uzunluğu, yürüyüş sayısı, p ve q parametrelerinin sırasıyla 20, 5, 0.5 ve 2 değerlerinin birleşimiyle node2vec'in yaklaşık %72 (46 doğru tahmin) gibi bir doğruluk oranı sağladığını göstermektedir. Aynı boyutta, yürüyüş uzunluğunun 40 veya 80'e çıkarıldığı senaryolarda da node2vec %67-69 civarında doğruluk elde edilebilmekte, bu da p ile q değerlerinin değişimine göre model çıktısının dalgalandığını ortaya koymaktadır. Benzer biçimde, Şekil 85(b)'den de görüldüğü üzere, 128 boyutlu yerleştirmede yürüyüş uzunluğunun 80, yürüyüş sayısının 5, p ve q değerlerinin de 0.5 ve 2 olduğu durum, node2vec'in yaklaşık %73 (47 doğru tahmin) düzeyinde daha yüksek tahmin başarısı

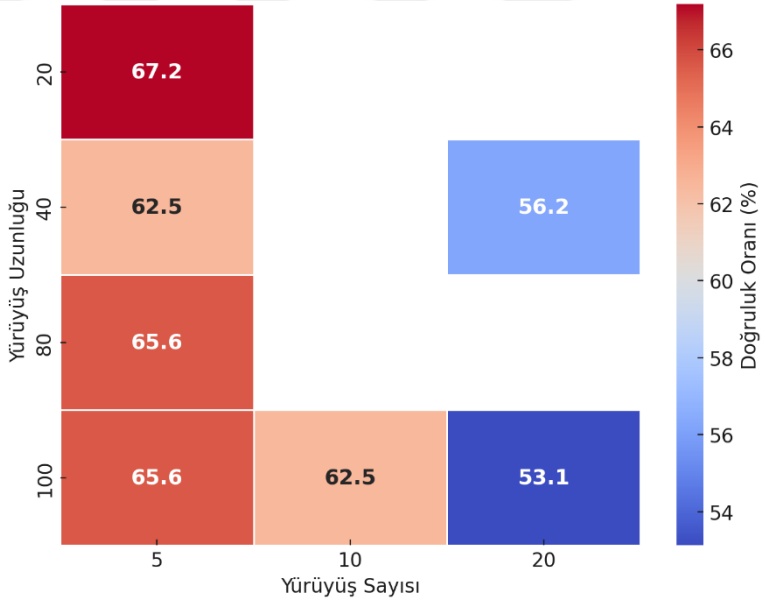
gösterdiği kombinasyon olarak öne çıkmaktadır. Buna karşılık, yürüyüş sayısının artırılması (örneğin, 20 veya 50) her zaman model performansını olumlu etkilememekte, bazen aşırı öğrenme veya gereksiz karmaşıklıkla sonuçlanarak doğruluk oranlarının %50-60 bandına gerilemesine neden olabilmektedir. Ayrıca, p = 0.5 ve q = 2 değerlerinin p = 1, q = 1 (yani klasik, tarafsız rastgele yürüyüş) veya p = 2, q = 0.5 (daha farklı ağırlıklandırılmış yürüyüş eğilimi) ayarlarına kıyasla daha güçlü yerel yapı keşfine imkân tanıdığı; böylece rastgele yürüyüşün, veri kümelerindeki alt toplulukları veya etkileşimli düğüm gruplarını daha etkin şekilde temsil edebildiği gözlenmektedir. Buna karşılık, p ve q'nun tersine çevrildiği durumlar ya da p = q = 1 gibi tarafsız senaryolar, modelin farklı yönlerde örnekleme yapmasına yol açarak geniş ölçekli keşfi artırabilmekte; ancak bu durumda belirli alt yapıların öğrenilmesi görece daha sınırlı kalabilmektedir.

Genel olarak, node2vec yönteminin düşük veya orta yerleştirme boyutlarında (64-128 aralığı) verimli sonuçlar ürettiği, ancak p ve q değerlerinin istikrarlı performans bakımından oldukça belirleyici olduğu söylenebilir. Ayrıca, yürüyüş uzunluğunun sınırlı tutulması (10 veya 20) ve yürüyüş sayısının (5 veya 10) makul düzeylerde kalması, doğruluk artışları sağlayabilmektedir. Sonuç olarak, node2vec yönteminde yerleştirme boyutu 128, yürüyüş uzunluğu 80, yürüyüş sayısı 5, iş parçacığı sayısı 2, p 0.5 ve q 2 değerlerinin oluşturduğu kombinasyon, test edilen tüm kombinasyonlar içinde en yüksek doğruluk oranını sağlamıştır. Bu nedenle, söz konusu ayarların diğer test projeleri üzerinde de ayrıntılı biçimde denenmesine karar verilmiştir.

DeepWalk yönteminin farklı yerleştirme boyutları (64 ve 128) altındaki yürüyüş uzunluğu ve yürüyüş sayısı parametrelerinin doğruluk oranı üzerindeki etkisi Şekil 86'da gösterilmektedir. Tüm koşullarda p ve q değerleri 1 olarak belirlenmiş, ayrıca iş parçacığı sayısı da 2 olarak sabit tutulmuştur. DeepWalk ve node2vec arasındaki temel fark, node2vec'in p ve q parametrelerini düğüm örnekleme stratejisini esnekleştirmek amacıyla değiştirebilmesi iken, DeepWalk yönteminin p = q = 1 şeklinde tarafsız rastgele yürüyüş yaklaşımını benimsemesidir. Grafiklerde yer almayan ek denemeler (örneğin 256 boyutlu yerleştirme veya farklı iş parçacığı sayısı değerleri) düşük performans sergilediğinden, sonuçların açıklanabilirliğini artırmak adına görselde gösterilmemiştir.



(a) 64 yerleştirme boyutlu



(b) 128 yerleştirme boyutlu

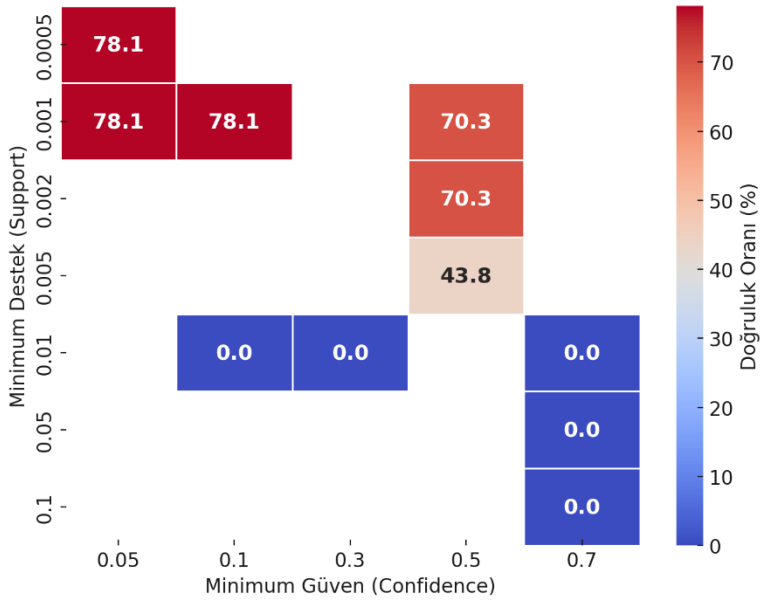
Şekil 86. DeepWalk yöntemi parametre değişimleri (a) 64 ve (b)128 yerleştirme boyutlu

Şekil 86 incelendiğinde, 64 boyutlu yerleştirmede yürüyüş uzunluğunun 20 ve yürüyüş sayısının 5 olduğu kombinasyonunun yaklaşık %70 gibi bir doğruluk oranı sağladığı dikkat çekmektedir. Aynı boyutta 40 veya 80 uzunluktaki yürüyüşler benzer düzeyde sonuçlar elde etse de yürüyüş uzunluğu ve yürüyüş sayısı parametrelerinin artırılması (örneğin, 20 ve 50) model performansını %53 civarına düşürerek olumsuz

etkide bulunmuştur. 128 boyutlu yerleştirmede ise en iyi sonuçlardan biri, yürüyüş uzunluğunun 20 ve yürüyüş sayısının 5 olduğu koşulda %67 olarak elde edilmiş ve aynı yürüyüş sayısında 80 ve 100 uzunluktaki yürüyüşlerde %66 düzeyinde performans gözlenmiştir. Ancak yürüyüş sayısının arttırılması durumunda doğruluk oranlarında anlamlı bir artış yerine, tam tersine %53-56 bandında bir gerileme kaydedilmiştir.

Genel olarak, iki yerleştirme boyutunda da yürüyüş uzunluğunun orta seviyelerde (20-40) tutulmasının ve yürüyüş sayısının makul düzeylerde (5-10) sınırlandırılmasının çoğu senaryoda daha yüksek doğruluk oranlarına ulaştığı görülmektedir. 64 boyutlu yerleştirme, 20 yürüyüş uzunluğu, 5 yürüyüş sayısı ve 2 iş parçacığı sayısı parametrelerinden oluşan yapı, incelenen tüm kombinasyonlar içerisinde en yüksek doğruluk oranını (yaklaşık %70) elde etmiştir. Bu nedenle, ilerleyen aşamalarda farklı test projeleri üzerinde de bu ayarların uygulanmasına karar verilmiştir.

Şekil 87'de, ARM yönteminin farklı minimum destek (support) ve minimum güven (confidence) değerleri altında gösterdiği performans yer almaktadır. Tüm denemelerde lift değerinin 1'den büyük olması tercih edilerek, incelenen ilişkilerin herhangi bir tesadüfi ortaklık yerine gerçekten anlamlı bir bağıntıya işaret etmesi sağlanmıştır. Başka bir deyişle, lift > 1, söz konusu kuralın beklenenden daha sık birlikte gerçekleştiğini ve dolayısıyla tahmin amaçlı kullanılabilecek nitelikte olduğunu göstermektedir (Mostofi vd., 2023a).



Şekil 87. ARM yöntemi parametre değişimleri

Şekil 87'de grafiksel olarak sunulan analiz sonuçları, destek ve güven değerlerinin aşırı düşük veya yüksek seçilmesinin üretilen kural sayısını ve dolayısıyla elde edilen doğruluk oranlarını önemli ölçüde değiştirdiğini göstermektedir. Örneğin, destek ≥ 0.01 ve güven ≥ 0.7 gibi katı eşikler, çok az sayıda (bazen 0 ya da 2) kural üretmiş ve doğruluk oranını 0'a kadar düşürmüştür. Buna karşılık, destek değerini 0.001 veya 0.0005 seviyesine çekerek güven değerini 0.05-0.1 veya 0.5 gibi orta düzeylerde tutmak, yüzlerce kuralın üretilmesine ve %70-78 aralığında yüksek doğruluk oranlarına ulaşılmasına olanak sağlamıştır. Özellikle 0.001 destek ve 0.1 güven koşulunda yaklaşık 1993 adet kural çıkarılmış ve elde edilen doğruluk oranı yaklaşık %78 düzeyine kadar yükselmiştir. Böylece, daha düşük destek ve ölçülü güven değerlerinin proje kapsamındaki ilişkileri daha geniş kapsamda yakaladığı ve tahmin doğruluğunu artırdığı gözlenmiştir.

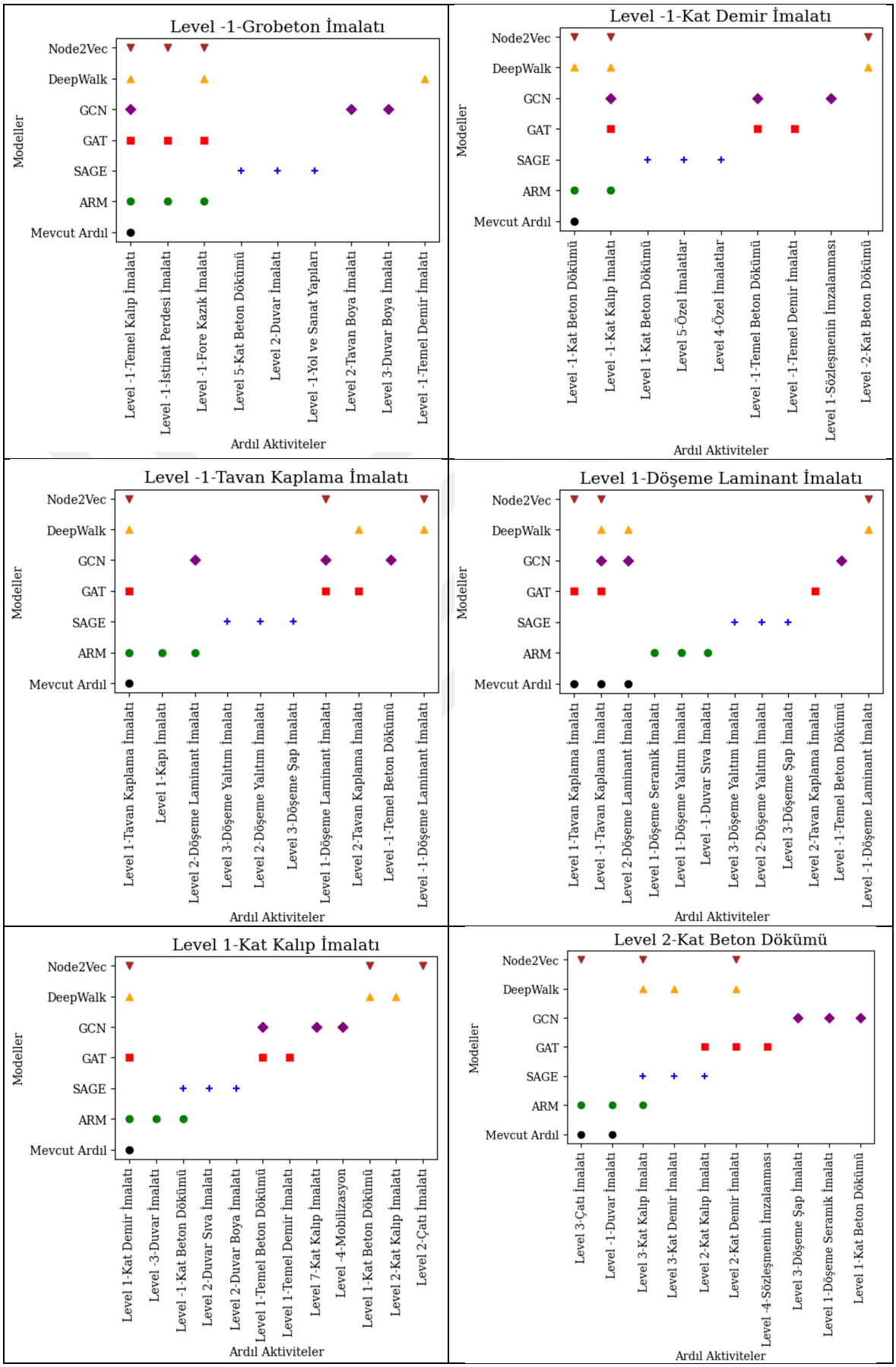
Elde edilen veriler doğrultusunda, destek ve güven parametrelerinin sırasıyla 0.001 ve 0.1 olduğu kombinasyonun diğer test projelerinde de uygulanacak en uygun parametre seti olduğuna karar verilmiştir. Bu seçim hem yüksek sayıda kural çıkarılmasına hem de anlamlı düzeyde doğruluk oranına ulaşılmasına imkân tanımakta ve dolayısıyla sahadaki aktivite-ardıl ilişkilerini etkili bir şekilde yansıtacak bir kural seti oluşturabilmektedir.

ML yöntemlerine ait elde edilen bu bulgular, incelenen tüm yöntemler (GCN, GAT, GraphSAGE, node2vec, DeepWalk ve ARM) için parametre seçiminin doğruluk oranı üzerinde belirleyici bir rol oynadığını göstermektedir. Her modelin kendi yapısı ve veri kümesinin özellikleri göz önünde bulundurularak özenle yapılandırılması, karmaşık inşaat projelerindeki aktivite-ardıl ilişkilerini doğru ve kapsamlı biçimde yakalamak açısından hayati önem taşımaktadır. Dolayısıyla, parametre hassasiyetlerini titizlikle yönetmek, yüksek doğruluk oranlarına ulaşmanın ve projenin ölçek, ağ yapısı veya veri miktarı gibi farklı zorluklarını aşmanın kilit unsuru olarak öne çıkmaktadır.

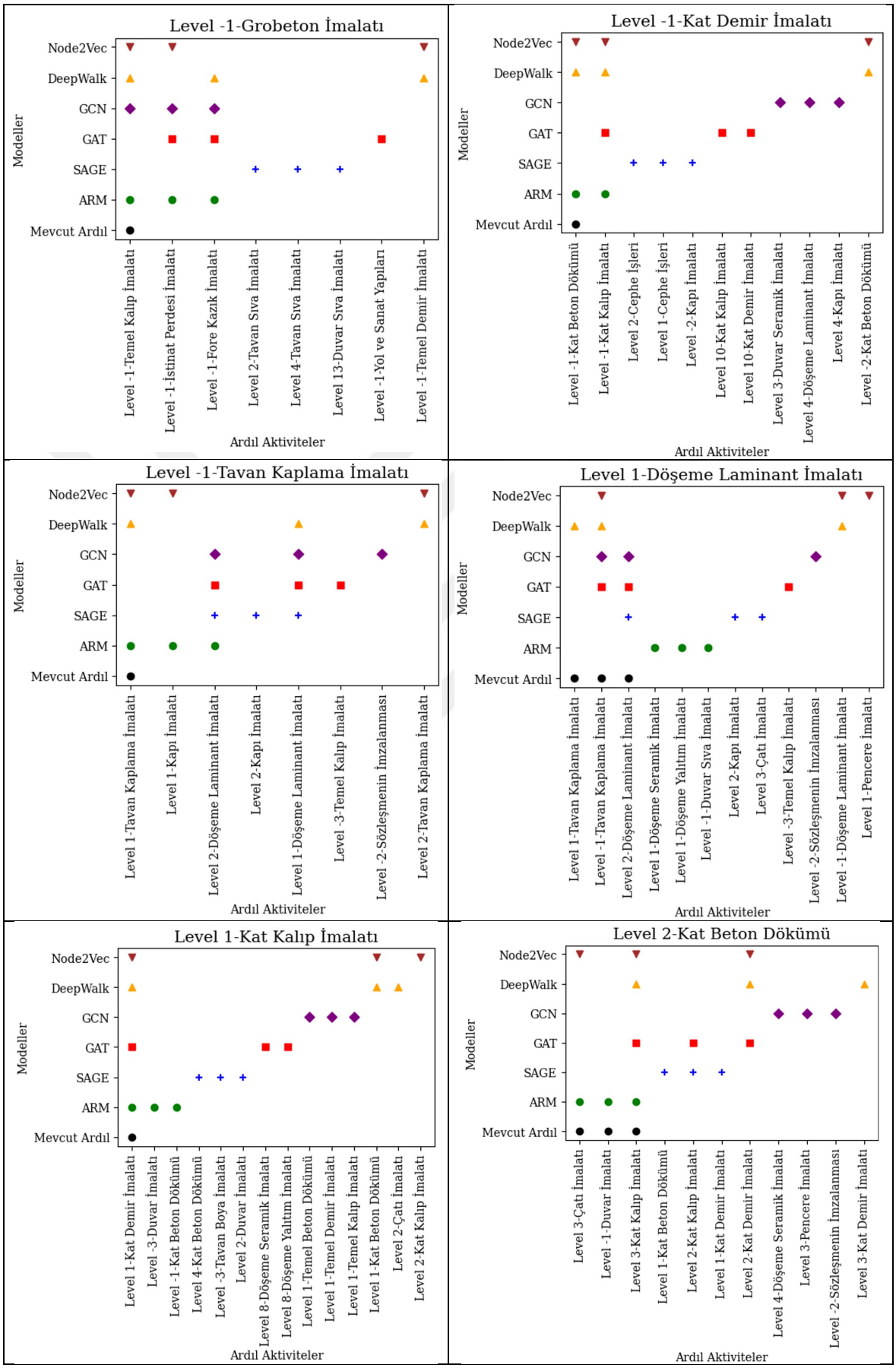
Her modelin en iyi parametreleri belirlendikten sonra, bu modellerin birbiriyle kıyaslanması aşamasına geçilmiştir. Bu amaçla, daha önceki bölümlerde tanımlanan test projeleri kullanılmıştır. Projelerin boyutu ve çalışma kapsamı dikkate alındığında, küçük ve orta ölçekli inşaat projelerine uygun veri dağılımını sağlamak adına, kat sayısı ve aktivite düzeyi görece daha düşük olan projeler (P1, P4, P10 ve P15) başlangıçta test kümesi için ayrılmıştır. Ardından, farklı aktivite yoğunluk kategorilerinden rastgele bir proje eklenerek (örneğin, P16 veya P9 vb.), farklı ölçek ve karmaşıklıktaki projelerde modellerin ne ölçüde başarılı tahminler yapabileceği sınanmıştır. Ayrıca, her yoğunluk kategorisinden ikişer proje seçmek suretiyle altı farklı eğitim-test senaryosu kurgulanarak,

veri sızıntısı riskinin en aza indirilmesi ve modellerin genellenebilirliğinin daha geniş biçimde ölçülebilmesi amaçlanmıştır. Bu strateji, eğitim aşamasında edinilen bilgi birikiminin, ölçek ve içerik bakımından değişiklik gösteren yeni projelerde de doğru tahmin ve sıralama yapabilmesini sağlamaya yöneliktir.

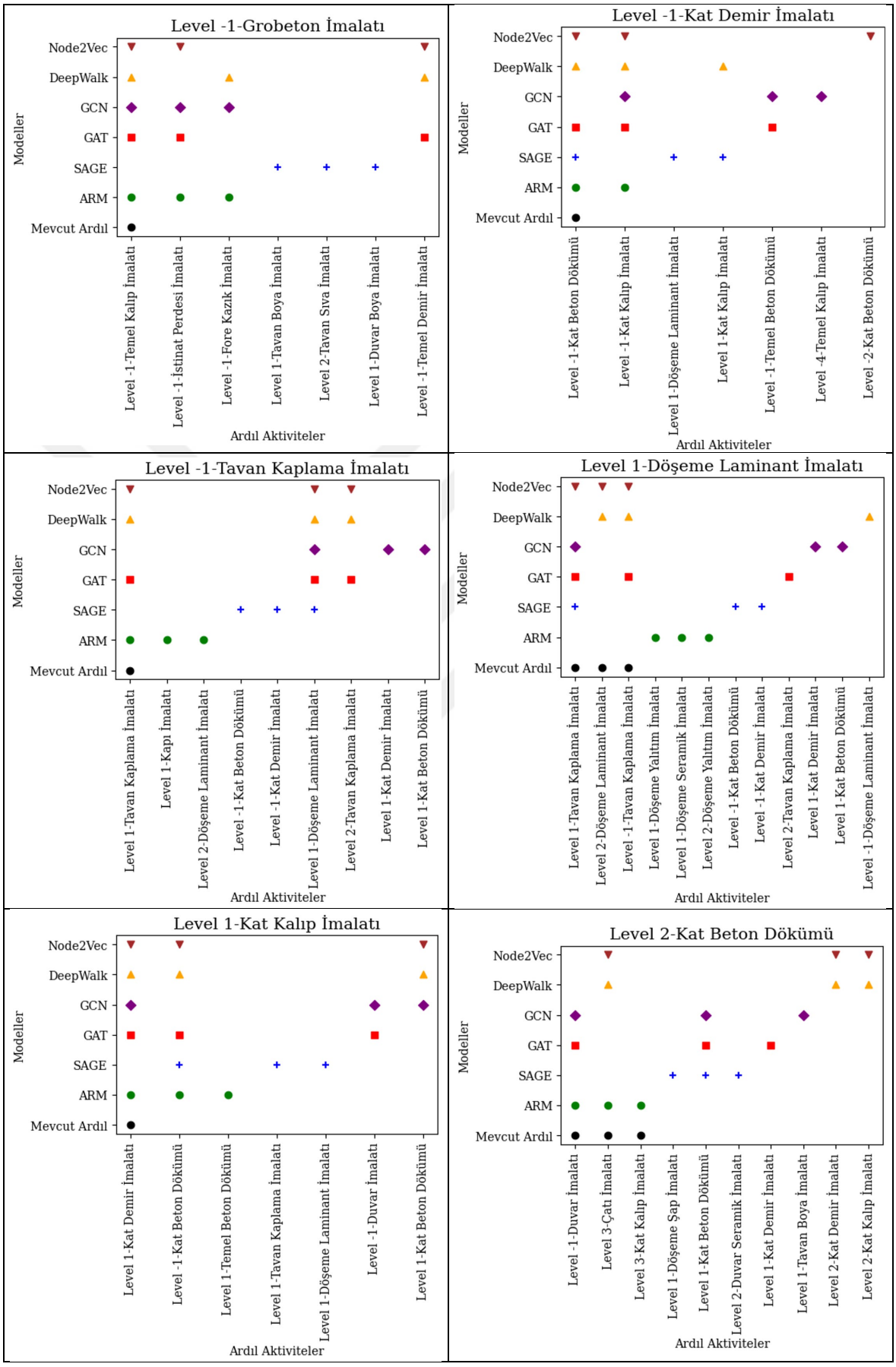
Şekil 88-90'da, en az, orta ve en çok tahmin edilecek aktivite içeren test gruplarında hem eğitim aşamasında sisteme tanımlanan "Mevcut Ardıl" (gerçekte var olan ardıl aktivite) hem de çeşitli ML modellerinin tahmin ettiği ardıl aktiviteler görsel olarak karşılaştırılmaktadır. Aktivite sayısının fazlalığı dikkate alınarak, her test grubu için örnek aktiviteler seçilerek sunulmuştur. Yatay eksende, ilgili aktivitenin (şeklin üst kısmında yazmakta, örneğin Level -1-Grobeton İmalatı) ardından gelmesi öngörülen (ML modelleri tarafından) farklı aktiviteler sıralanırken, dikey eksende kullanılan ML modelleri ve "Mevcut Ardıl" listelenmiştir. ML modellerinin ardıl olarak tahmin ettiği aktivite, modele ait işaretin yatay eksende kesiştiği aktiviteyle gösterilmektedir. Bu şekilde, aynı aktivite için modellerin yaptığı tahmin, siyah noktalarla belirtilmiş "Mevcut Ardıl" değerleriyle bir arada gözlemlenebilmektedir. Eğer bir modelin işareti siyah noktanın yer aldığı sütundaysa, o modelin tahmini gerçekte olanla birebir örtüşmektedir. Diğer sütunlarda konumlanan tahminler ise farklı bir ardıl öneren modelleri göstermektedir. Bu sayede, ML modellerinin hangi ardıl aktiviteyi tercih ettiği ve söz konusu tahminin gerçek (mevcut) aktiviteyle ne kadar uyuştuğu açıkça ortaya koyulmaktadır.



Şekil 88. P1, P4, P10, P15 ve P16 test projeleri üzerinde ML modellerin ardıl tahminleri



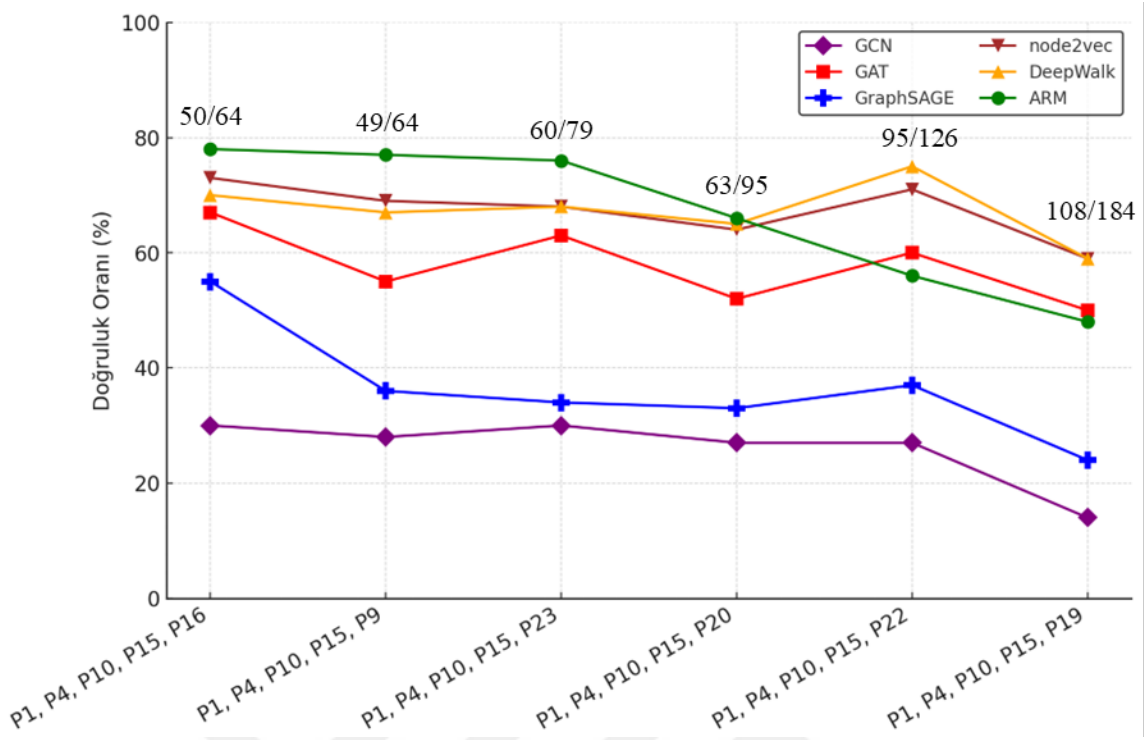
Şekil 89. P1, P4, P10, P15 ve P23 test projeleri üzerinde ML modellerin ardıl tahminleri



Şekil 90. P1, P4, P10, P15 ve P19 test projeleri üzerinde ML modellerin ardıl tahminleri

Şekil 88-90 incelendiğinde, “Level -1-Grobeton İmalatı” aktivitesinin, test kümelerindeki projelerde yer alan mevcut ardıl değerinin “Level -1-Temel Kalıp İmalatı” olduğu görülmektedir. P1, P4, P10, P15, P16 ve P1, P4, P10, P15, P19 test kümeleri (Şekil 88 ve Şekil 90) üzerinde “Level -1-Grobeton İmalatı” aktivitesinin ardılını node2vec, DeepWalk, GCN, GAT ve ARM modellerinin yaptıkları üç tahminden biri ile doğru buldukları ancak GraphSAGE modelinin üç tahmininin de yanlış olduğu görülmektedir. Şekil 89’da bulunan P1, P4, P10, P15, P23 test kümesi üzerindeki tahminlerde ise “Level -1-Grobeton İmalatı” aktivitesinin ardılını, diğer test kümelerinden farklı olarak GAT modelinin de yanlış tahmin ettiği tespit edilmiştir. “Level 2-Kat Beton Dökümü” aktivitesine ait ardıl tahmini incelendiğinde ise, test kümeleri üzerinde birbirinden farklı mevcut ardıl ve ardıl tahminleri olduğu görülmektedir. Örneğin, Şekil 88’deki test kümelerinin mevcut ardılları “Level 3-Çatı İmalatı” ve “Level -1-Duvar İmalatı” iken, Şekil 89 ve 90’daki test kümelerinde bu iki ardıla ek olarak “Level 3-Kat Kalıp İmalatı” aktivitesi de yer almaktadır. Bunun sebebi, P19 ve P23 projelerinin normal kat sayısının 2’den büyük (sırasıyla 9 ve 6 katlı) olmasıdır. Yani burada, “Level 2-Kat Beton Dökümü” aktivitesinin ardılı olarak bu iki projede “Level 3-Kat Kalıp İmalatı” yer almaktadır. Test proje grupları için oluşturulan bu görseller, ML modellerinin performanslarını değerlendirmek için kullanılarak modellerin doğruluk oranları elde edilmiştir.

Şekil 91’de tez kapsamında seçilen ML modellerinin farklı test proje kümeleri üzerindeki performans sonuçları görsel olarak karşılaştırmalı biçimde gösterilmektedir. Yatay eksen her bir test kombinasyonunun toplam tahmin edilen aktivite sayısı az olandan çok olana doğru ilerlemesini ifade ederken, dikey eksen doğruluk oranlarını (yüzde cinsinden) yansıtmaktadır. Ayrıca, her test grubunda en yüksek doğruluk oranı elde edilen model için doğru tahmin sayısı ve o gruptaki toplam aktivite sayısı da gösterilmektedir.



Şekil 91. ML modellerinin test projeleri üzerindeki performansları

Şekil 91 incelendiğinde, ilk dört test kümesi üzerinde ARM'nin en yüksek doğruluk değerlerine (örneğin, birinci test kümesi için 50/64 ≈ %78) ulaştığı görülmektedir. Buna karşılık aynı test kümesinde node2vec %47 (37/64) ve DeepWalk %70 (45/64) oranlarıyla ARM'ın gerisinde kalmıştır. Öte yandan, 126 aktivitenin tahmin edildiği P1, P4, P10, P15 ve P22 test grubunda, DeepWalk %75'e varan doğruluk oranıyla ARM'ı ve node2vec'i geride bırakabilmekte (node2vec 89 doğru ile yaklaşık %71 ve ARM 71 doğru ile yaklaşık %56 doğruluk oranı) ve böylece farklı koşullarda yöntemler arasında en yüksek doğruluğa ulaşan ML modelinin değiştiği izlenebilmektedir. Node2vec ve DeepWalk'ın çoğunlukla yakın doğruluk oranları elde ettikleri de görülmektedir. GNN tabanlı yöntemlere bakıldığında, GCN tüm gruplarda %14 (26/184) - %30 (24/79) gibi düşük oranlarda kalırken, GAT %50 (92/184) - %67 (43/64) düzeyinde orta ölçekli bir performans sunmakta ve GraphSAGE ise ilk test grubu (35 doğru ile yaklaşık % 55 doğruluk) hariç çoğunlukla %24 (44/184) - %36 (23/64) aralığı ile GCN'e yakın sonuçlar elde etmiştir. Bu farklılıklar, projelerin ölçek ve kapsam değişkenliği ile model yapılandırmalarının parametre seçimi gibi etmenlerin performans üzerinde belirleyici bir rol oynadığını ortaya koymakta; dolayısıyla, her bir yöntemin üstünlüğü ya da geride kalışı, test grubunun içeriğine ve veri setinin özelliklerine göre değişebilmektedir.

Modellerin değerlendirilmesi için elde edilen verilerde, ARM yaklaşımının görece daha az aktiviteden oluşan test kümelerinde (örneğin, toplam 64 veya 79 aktivite tahmin edilen senaryolarda) yüksek doğruluk oranlarına (örneğin, 50/64 ≈ %78 veya 60/79 ≈ %76) erişebildiği, ancak test projelerindeki aktivite sayısı ve karmaşıklığı arttıkça performansının düştüğü dikkat çekmektedir. Buna karşılık, node2vec ve DeepWalk yöntemlerinin özellikle daha büyük aktivite setlerine sahip test gruplarında (126 ve 184 aktivite tahmin edilen durumlarda) ARM'ı geride bırakarak daha yüksek doğruluk oranlarına ulaştığı görülmektedir. Bu durum, az sayıda aktivitenin yer aldığı projelerde birliktelik kurallarının (ARM) sıklıkla güçlü bir tahmin çerçevesi sunduğunu, ancak ölçek ve karmaşıklık düzeyi arttıkça rastgele yürüyüş tabanlı node2vec ve DeepWalk yöntemlerinin daha esnek ve geniş veri yapısını daha iyi yakalayabildiklerini göstermektedir. Dolayısıyla, hangi yöntemin ön plana çıkacağı, proje kapsamına ve veri setinin içeriğine yakından bağlı olup, etkin model seçimi yapabilmek için aktivite düzeyinin ve projenin bütüncül karmaşıklık özelliklerinin göz önünde bulundurulması gerekmektedir.

Literatürde, inşaat projelerindeki aktivite sıralamalarının otomatik veya yarı otomatik yollarla öğrenilmesine yönelik çeşitli çalışmalara rastlanmaktadır. Bu kapsamda, Tablo 11'de özetlenen araştırmalar; kullanılan ML yöntemleri, veri büyüklüğü ile niteliği ve elde edilen başarı oranları çerçevesinde değerlendirilmektedir. Söz konusu çalışmalar, geleneksel (manuel) program oluşturma ya da iş programı denetleme süreçlerine alternatif geliştirerek, geçmiş projelerin iş programlarından yararlanarak "aktivite sıralaması" ve "ilişki tahmini" alanına önemli katkılar sunmaktadır.

Tablo 11. Aktivite sıralama çalışmaları ve sonuçları

Çalışma	ML Türü	Veri Kaynağı & Miktarı	Hedef	Sonuç
Amer & Golparvar-Fard (2019)	LSTM	12 proje & 29.000 aktivite	Önceden verilen öncüllerin ardından ardıl tahmini	%92 tahmin doğruluğu
Amer & Golparvar-Fard (2021)	LSTM	32 proje & 78.704 aktivite	Ardıl/öncül tahmini (aktivite sıralaması)	Test kümesinde ~%63–74 ardıl tahmini & ~%66–72 öncül tahmini doğruluğu
Amer vd. (2022)	LSTM + NLP	3 ticari proje & 3.387 aktivite, ikili ilişkilerle yüzbinlerce etiketli çift	Her iki aktivite arasındaki ilişki tipini (FS, SS, FF, SF) otomatik saptama	%85,14'lük kesinlik (Precision), %95,12'lik geri çağırma (Recall) & %88,3'lük F1 skoru, yön tayininde ~%85 başarı
Prieto vd. (2023)	LLM (Large Language Model)	Küçük ölçekli senaryo (oda tadilatı)	Doğal dil girdisiyle taslak program (aktivite sıralaması + süre)	Mantıksal sıra genelde tutarlı ancak gereksiz aktiviteler bulunmakta. Uzman onayına ihtiyaç var.
Hong vd. (2023)	GAS (Graph-Based Automated Scheduling) & GCN	5.050 proje & 353 proje test için & 2 proje doğrulama	BIM'e düşük bağımlı otomatik iş programı + zaman/maliyet optimizasyon	Otomatik oluşturulan program, planlanan programa kıyasla ortalama %6,70 daha yakından gerçek sonuç elde etmiştir.

Amer & Golparvar-Fard (2021), büyük veri kümelerinden (32 proje ve 78.000'den fazla aktivite) yararlanarak sıklıkla tekrarlayan ardıl-öncül kalıplarını yakalamayı amaçlayan LSTM tabanlı bir yöntem geliştirilmiştir. Bu tür DL modellerinin, veri hacmi arttıkça ardıl-öncül ilişkileri eğitim setinde %76-98 ve test setinde yaklaşık %60–75 doğruluk aralığında tahmin edebildiği rapor edilmiştir. Aynı araştırma grubunun bir başka çalışmasında (Amer & Golparvar-Fard, 2019), 29.000 aktivite içeren 12 projeden elde edilen verilerle eğitilen bir LSTM modeli, önceden verilen öncülleri temel alarak ardıl faaliyeti %92 doğrulukla tahmin etmiştir. Benzer biçimde, Amer vd. (2022), her iki aktivite arasındaki ilişkiyi tanımlayan bir model sunmuş; bu modelin, yaklaşık %85 kesinlik (precision), %95 geri çağırma (recall) ve %88 F1 skoru elde ettiği belirtilmiştir.

Derin öğrenmeye alternatif olarak, Prieto vd. (2023), ChatGPT gibi büyük dil modellerini (Large Language Model, LLM) küçük ölçekli bir inşaat işinde (oda tadilatı) denemiş ve proje kapsamını metinsel olarak anlatarak, modelin taslak bir iş programı üretme potansiyelini incelemiştir. Elde edilen sonuçlar, ChatGPT'nin faaliyet sıralamasını genellikle mantıklı şekilde oluşturabildiğini ancak zaman zaman gereksiz aktiviteler eklediğini göstermiş, bu nedenle uzman onayının gerekli olduğu vurgulanmıştır. Büyük dil

modellerinin bu yaklaşımı, inşaat özelinde sistematik bir eğitim sürecinden geçmediği sürece daha çok "ilk taslak oluşturma" düzeyinde fayda sağlamaktadır.

Hong vd. (2023) tarafından önerilen Grafik Tabanlı Otomatik İş Programı (Graph-Based Automated Scheduling, GAS) yöntemi, inşaat projelerinin erken aşamalarında BIM'e bağımlılığı en aza indirerek otomatik program oluşturmayı hedeflemiştir. Bu yöntem, proje programlarındaki metinsel ve grafiksel verileri bütünleştiren bir DL yaklaşımıyla tipik aktivite dizilerini belirlemekte, ardından genetik algoritmalarla zaman ve maliyet bakımından etkin bir plan üretmektedir. Vaka çalışmaları, GAS yönteminin planlanan çizelgeye oranla ortalama %6,70 daha gerçeğe yakın maliyet ve süre sonuçları sunduğunu ortaya koymaktadır. Böylece, detaylı BIM modellemesine henüz sahip olmayan projelerde de bilgi birikimini yakalayıp değerlendirme olanağı sunmakta ve ilk aşama planlamayı kolaylaştırmaktadır.

Bu genel çerçevede incelendiğinde, tez çalışmasında ele alınan yöntemlerin de benzer şekilde aktivite sıralaması odaklı olduğuna ve farklı ölçekteki projelerde değişen veri büyüklükleriyle çalışmaya elverişli sonuçlar sunduğuna işaret edilmektedir. Özellikle daha küçük ölçekli projelerde ARM yaklaşımının, büyük projelerde ise node2vec ve DeepWalk yöntemlerinin LSTM ile benzer doğruluk oranları yakaladığı görülmüş, böylece farklı modelleme stratejilerinin proje boyutuna göre uyarlanmasının yararlı olabileceği anlaşılmıştır. Ayrıca, tezde kullanılan ML modellerinin, LSTM benzeri yüksek veri gereksinimine duyulan ihtiyacı önemli ölçüde azaltarak benzer doğruluk oranlarına ulaşması, daha düşük veri ortamlarında da aktivite sıralaması tahmininin başarıyla yapılabileceğine işaret etmektedir.

ML modelleri için belirlenen parametreler ve 24 proje içinden seçilen altı farklı test setinde gerçekleştirilen karşılaştırmalar sonucunda, tezde 3B olarak modellenen yapıya benzer özelliklere sahip bir test kümesi ve bu kümenin içerisinden bir proje tercih edilmiştir. Ardından, seçilen projeye ait aktivite sıralama dosyası elde edilmiştir. Bu küme, en büyük projesi bodrum, zemin ve bir normal kat olmak üzere üç kat barındıran örnek projeyle kat sayısı ve yapı karmaşıklığı bakımından en yakın benzerlik gösteren P1, P4, P9, P10 ve P15 numaralı projelerden oluşmaktadır. Bu test kümesi içerisinden ise aktivite sıralamasının elde edilmesi için en fazla kat ve aktivite sayısına sahip P9 numaralı proje seçilmiştir. Şekil 91'deki ML model karşılaştırması incelendiğinde, söz konusu test grubunda en yüksek doğruluk oranına %77 değeri ile ARM'nin ulaştığı görülmüştür. Bu nedenle, örnek projeye ait aktivite sıralama dosyasını üretmek üzere, ARM yöntemi temel

alınmıştır. Söz konusu analiz, EK 7'de sunulan Python tabanlı bir kod aracılığıyla yürütülmüş ve elde edilen çıktı dış ortama aktarılmak üzere hazırlanmıştır.

Bu doğrultuda, Tablo 12'de ARM yönteminin P9 projesindeki tahmin sonuçlarının bir kısmı verilmiştir. Önceki aşamalarda da belirtildiği üzere, her aktivite için üçer adet muhtemel ardıl tahmini yapılmış ve bu tahminlerden herhangi birinin doğru ardıla karşılık gelmesi durumunda ilgili aktivite "doğru" olarak sınıflandırılmıştır. Söz konusu çoklu tahmin yaklaşımı, inşaat projelerindeki her faaliyetin birden fazla ardıl ilişkisine sahip olabileceği gerçeğine dayanmaktadır.

Tablo 12. P9 projesi için gerçek ardıllar ve ARM modeli ardıl tahmini örnekleri

Aktivite Adı	Mevcut Ardıl (Ham Veri)	ARM Tahmin Ettiği Ardıl
Level -1-Duvar Boya İmalatı	Level 1-Duvar Boya İmalatı	Level 1-Duvar Boya İmalatı, Level 2-Asma Tavan İmalatı, Level 2-Duvar Seramik İmalatı
Level -1-Duvar Seramik İmalatı	Level 1-Duvar Seramik İmalatı	Level 1-Duvar Seramik İmalatı, Level 1-Duvar Boya İmalatı, Level 2-Duvar Boya İmalatı
Level -1-Duvar Sıva İmalatı	Level 1-Duvar Sıva İmalatı	Level 1-Duvar Sıva İmalatı, Level 2-Döşeme Laminant İmalatı, Level 2-Döşeme Seramik İmalatı
Level -1-Döşeme Seramik İmalatı	Level 1-Döşeme Seramik İmalatı	Level 2-Döşeme Yalıtım İmalatı, Level 1-Döşeme Seramik İmalatı, Level 1-Döşeme Yalıtım İmalatı
Level -1-Döşeme Yalıtım İmalatı	Level 1-Döşeme Yalıtım İmalatı	Level 1-Döşeme Yalıtım İmalatı, Level 2-Duvar Sıva İmalatı, Level 2-Duvar İmalatı
Level -1-Fore Kazık İmalatı	Level -1-İstinat Perdesi İmalatı, Level -1-Grobeton İmalatı	Level -1-İstinat Perdesi İmalatı, Level -1-Grobeton İmalatı, Level -1-Yol ve Sanat Yapıları
Level -1-Grobeton İmalatı	Level -1-Temel Kalıp İmalatı	Level -1-İstinat Perdesi İmalatı, Level -1-Fore Kazık İmalatı, Level -1-Temel Kalıp İmalatı
Level -1-Kapı İmalatı	Level 1-Kapı İmalatı	Level -2-Kapı İmalatı, Level -2-Özel İmalatlar, Level 1-Kapı İmalatı
Level -1-Kat Beton Dökümü	Level 1-Kat Kalıp İmalatı	Level 1-Kat Kalıp İmalatı, Level -3-Duvar İmalatı, Level -1-Kat Demir İmalatı
Level -1-Kat Demir İmalatı	Level -1-Kat Beton Dökümü	Level -1-Kat Beton Dökümü, Level -1-Kat Kalıp İmalatı
Level -1-Kat Kalıp İmalatı	Level -1-Kat Demir İmalatı	Level -1-Kat Demir İmalatı, Level -1-Temel Beton Dökümü, Level -2-Kat Beton Dökümü
Level -1-Temel Beton Dökümü	Level -1-Kat Kalıp İmalatı	Level -1-Kat Kalıp İmalatı, Level -1-Temel Demir İmalatı
Level -1-Temel Demir İmalatı	Level -1-Temel Beton Dökümü	Level -1-Temel Beton Dökümü, Level -1-Temel Kalıp İmalatı
Level -1-Temel Kalıp İmalatı	Level -1-Temel Demir İmalatı	Level -1-Grobeton İmalatı, Level -1-Temel Demir İmalatı
Level -1-Yol ve Sanat Yapıları	Level -1-Fore Kazık İmalatı	Level -1-Fore Kazık İmalatı
Level -1-İstinat Perdesi İmalatı	Level -1-Grobeton İmalatı	Level -1-Grobeton İmalatı, Level -1-Fore Kazık İmalatı

Tablo 12'nin devamı

Level 1-Duvar İmalatı	Level 2-Duvar İmalatı	Level -1-Döşeme Yalıtım İmalatı, Level 1-Duvar Sıva İmalatı, Level 1-Döşeme Yalıtım İmalatı
Level 1-Kat Beton Dökümü	Level 2-Kat Kalıp İmalatı	Level -1-Duvar İmalatı, Level 2-Kat Kalıp İmalatı, Level -2-Duvar İmalatı
Level 1-Kat Demir İmalatı	Level 1-Kat Beton Dökümü	Level 1-Kat Beton Dökümü, Level 1-Kat Kalıp İmalatı
Level 1-Kat Kalıp İmalatı	Level 1-Kat Demir İmalatı	Level -1-Kat Beton Dökümü, Level -3-Duvar İmalatı, Level 1-Kat Demir İmalatı
Level 2-Duvar Sıva İmalatı	Level -1-Duvar Boya İmalatı	Level -1-Duvar Boya İmalatı, Level 1-Duvar Boya İmalatı, Level 1-Asma Tavan İmalatı
Level 2-Duvar İmalatı	Level -1-Döşeme Yalıtım İmalatı	Level -1-Döşeme Yalıtım İmalatı, Level 1-Döşeme Yalıtım İmalatı, Level 2-Duvar Sıva İmalatı
Level 2-Döşeme Laminant İmalatı	Level -1-Tavan Kaplama İmalatı	Level -1-Duvar Sıva İmalatı, Level 1-Duvar Sıva İmalatı, Level 2-Döşeme Seramik İmalatı
Level 2-Döşeme Seramik İmalatı	Level -1-Pencere İmalatı	Level -1-Döşeme Laminant İmalatı, Level 3-Döşeme Yalıtım İmalatı, Level -1-Duvar Sıva İmalatı,
Level 2-Döşeme Yalıtım İmalatı	Level -1-Duvar Sıva İmalatı	Level -1-Duvar Sıva İmalatı, Level 1-Duvar Sıva İmalatı, Level 1-Döşeme Seramik İmalatı
Level 2-Döşeme Şap İmalatı	Level -1-Döşeme Seramik İmalatı	Level 1-Döşeme Seramik İmalatı, Level 2-Döşeme Yalıtım İmalatı, Level 1-Döşeme Yalıtım İmalatı
Level 2-Kat Beton Dökümü	Level 3-Çatı İmalatı, Level -1-Duvar İmalatı	Level 3-Çatı İmalatı, Level -1-Duvar İmalatı, Level 3-Kat Kalıp İmalatı
Level 2-Kat Demir İmalatı	Level 2-Kat Beton Dökümü	Level 2-Kat Beton Dökümü, Level 2-Kat Kalıp İmalatı
Level 2-Pencere İmalatı	Level -1-Döşeme Laminant İmalatı	Level 1-Döşeme Laminant İmalatı, Level 2-Döşeme Seramik İmalatı, Level 1-Pencere İmalatı
Level 2-Tavan Kaplama İmalatı	Level -1-Kapı İmalatı	Level 1-Kapı İmalatı, Level -1-Kapı İmalatı, Level 1-Tavan Kaplama İmalatı
Level 3-Çatı İmalatı	Level -1-Cephe İşleri, Level 0-Mekanik İşler, Level 0-Elektrik İşleri	Level 0-Elektrik İşleri, Level 1-Cephe İşleri, Level 2-Kat Beton Dökümü

Tablo 12'de herhangi bir aktivite sıralaması yapılmamış olup, yalnızca ilgili projenin aktiviteleri için bir ardıl tahmini çalışması yürütülmüştür. Daha öncede açıklandığı üzere, ARM'nin her aktivite için ürettiği üç tahminden en az birinin ham veriyle eşleşmesi hâlinde bu tahmin "doğru", hiçbir tahminin örtüşmemesi durumunda ise "yanlış" şeklinde kabul edilmiştir. Örneğin "Level -1-Grobeton İmalatı" için tahminlerden biri mevcut ardıla karşılık geldiğinden "doğru" kabul edilmiş, ancak "Level 2-Döşeme Laminant İmalatı" aktivitesi için ARM'nin sunduğu üç tahminden hiçbiri gerçek ardılla uyuşmadığı için "yanlış" değerlendirilmiştir. Ayrıca, test setine dâhil olmayan ya da projeye uygun düşmeyen kat düzeylerinde (örneğin, Level -3, Level -2, Level 3 veya daha yüksek katlar) yer alan aktivitelerin de tahmin edilebildiği görülmektedir. Bu durum, eğitim setinin daha yüksek veya farklı kat sayılarına sahip projeler içermesinden kaynaklanmaktadır.

BIM modeli ile bütünleştirmenin sağlıklı biçimde gerçekleşebilmesi amacıyla, aktivitelerin sıralanması aşaması için gerçekleştirilen Python kodunun çalıştırılması öncesinde "Başlangıç Aktivitesi", "Bodrum Kat Sayısı" ve "Normal Kat Sayısı" bilgileri kod içerisine tanımlanmıştır. Bu örnek projede, "Level -1-Grobeton İmalatı" başlangıç aktivitesi olarak belirlenmiş, bodrum kat sayısı 1, normal kat sayısı ise 2 şeklinde atanmıştır. Böylece, test projelerindeki kat sayıları ve ilgili aktiviteler, 3B BIM modelini destekleyecek şekilde oluşturularak modelde bulunmayan aktivitelerin bir kısmı tahmin sürecinde göz ardı edilerek daha tutarlı bir sıralama elde edilecektir. Örneğin, test kümesinde yer alan P15 projesi yalnızca zemin kattan oluştuğu için, aktiviteler içerisinde "Level 1-Grobeton İmalatı", "Level 1-Temel Kalıp İmalatı", "Level 2-Çatı İmalatı" gibi aktiviteler bulunmaktadır. P1 projesi ise bodrum ve zemin kattan oluştuğundan, aktiviteler arasında "Level -1-Grobeton İmalatı" ve "Level -1-Temel Kalıp İmalatı" gibi aktiviteler bulunmaktadır. Örnek projede ise bodrum kat yer aldığından grobeton ve temele ait imalatların Level 1'de değil de Level -1'de bulunması gerektiğinden, başlangıç aktivitesi ve kat sayısı gibi filtreler eklenmiş ve hem başlangıç aktivitesi hem de bodrum kat sayısı 1 girilerek örnek projenin "Level -1-Grobeton İmalatı" aktivitesi ile başlaması ve Level -1'de bulunan diğer aktiviteleri de tahmin ederek sıralama oluşturması sağlanmıştır. Bu sayede elde edilecek dosyada "Level 1-Grobeton İmalatı" ve "Level 1-Temel Kalıp İmalatı" gibi örnek projeye uygun olmayan aktiviteler yer almayacaktır. Böylelikle, kod içerisinde farklı projelerden aktarılan aktivite verilerinin modelin gerçek yapısıyla uyumlu hâle getirilmiştir.

Oluşturulan kod, başlangıç aktivitesini listeye ilk sırada yerleştirmekte ve bu aktivitenin ardılı (varsa) bir sonraki satıra eklenmektedir. Ardıl aktivitenin de kendi ardılını içermesi hâlinde, bu süreç yeni bir satırda tekrarlanarak ardılı kalmamış bir aktiviteye ulaşılana kadar devam etmektedir. Böylelikle, sahada izlenen iş akışını yansıtan bir aktivite sıralaması oluşturulmaktadır. Bu süreç içerisinde, bir önceki bölümde belirtildiği gibi kat filtrelemesi ile modelde bulunmayan aktiviteler sıralamaya dahil edilmemiştir. Örneğin, "Level -1 Kapı İmalatı" aktivitesi için sistemin "Level -2 Kapı İmalatı" şeklinde bir tahminde bulunabildiği dikkat çekmektedir. Ancak, bodrum kat sayısını 1 olarak tanımlamak suretiyle, söz konusu seviye (-2) aktiviteleri otomatik olarak ardıl sıralama listesine dahil edilmemekte ve bu yolla proje kapsamıyla örtüşmeyen fazladan imalatlar ayıklanmaktadır. Ayrıca, bazı aktiviteler sıralama sürecinde birden fazla kez ardıl olarak tahmin edilebilmektedir. Bu durumda, ardılın daha önce listelenip listelenmediği kontrol edilmekte ve sıralamaya tekrar eklenerek yinelenme yapılmamaktadır. Örneğin, "Level 1-Kat Demir İmalatı" için ardıl aktivite olarak "Level 1-Kat Beton Dökümü" ve "Level 1-Kat Kalıp İmalatı" tahmin edilmiştir. Fakat, "Level 1-Kat Kalıp İmalatı" zaten oluşturulmuş bir aktivite olduğundan, sıralamada yenilenmemiştir. Benzer bir durum, "Level -1-Kat Kalıp İmalatı" için önerilen ardıllar da görülmektedir. Tablo 12'de yer alan, "Level -1-Kat Kalıp İmalatı" için önerilen ardıllar incelendiğinde, "Level -1-Kat Demir İmalatı" henüz listelenmediği için sonraki aktivite olarak eklenmiş; "Level -1-Temel Beton Dökümü" ise daha önce yazıldığı gerekçesiyle atlanmıştır. Öte yandan, "Level -2-Kat Beton Dökümü" bodrum kat sayısının 1 olması nedeniyle BIM entegrasyonu çerçevesinde geçersiz kabul edilerek dahil edilmemiş, böylece ardıl sıralamanın veri bütünlüğüyle uyumlu biçimde oluşturulması sağlanmıştır.

Ayrıca, ML modelinin ardılını tahmin edemediği aktiviteler, "Tahmin Edilemeyen Aktiviteler" başlığı altında gruplanmakta ve bu sayede kullanıcıların ilgili işlemleri gözden kaçırmadan manuel şekilde düzenleyebilmeleri sağlanmaktadır. Tablo 13'te, incelenen test kümesi üzerinden P9 projesine ait ARM modelinden elde aktivite – ardıl tahmini ile elde edilen aktivite sıralaması ve sıralama aşamasında eşleştirilemeyen aktiviteler gösterilmektedir. Yukardaki bölümde açıklandığı üzere ARM modeline girdi olarak aktivite – ardıl tahmini listesi (Tablo 12'de bir kısmı gösterilmektedir) verilmekte ve çıktı olarak aktivite sıralaması (Tablo 13) elde edilmektedir. "Aktivite Adı" ve "Tahmin Edilemeyen Aktiviteler" adı altında verilen aktiviteler, .csv formatında dış ortama

aktarılmakta ve proje koşullarına göre dinamik biçimde güncellenebilecek bir temel çalışma dokümanı oluşturulmaktadır.

Tablo 13. Aktivite sıralama dosyası

Aktivite Adı	**Tahmin Edilemeyen Aktiviteler**
Level -1-Grobeton İmalatı	Level 1-Pencere İmalatı
Level -1-İstinat Perdesi İmalatı	Level -1-Kapı İmalatı
Level -1-Fore Kazık İmalatı	Level 2-Pencere İmalatı
Level -1-Temel Kalıp İmalatı	Level 2-Cephe İşleri
Level -1-Yol ve Sanat Yapıları	Level 2-Duvar Boya İmalatı
Level -1-Temel Demir İmalatı	Level -1-Döşeme Laminant İmalatı
Level -1-Temel Beton Dökümü	Level 1-Tavan Kaplama İmalatı
Level -1-Kat Kalıp İmalatı	Level -1-Duvar Sıva İmalatı
Level -1-Kat Demir İmalatı	Level -1-Yer Teslimi
Level -1-Kat Beton Dökümü	Level 2-Tavan Kaplama İmalatı
Level 1-Kat Kalıp İmalatı	Level -1-Duvar Seramik İmalatı
Level 1-Kat Demir İmalatı	Level -1-Pencere İmalatı
Level 1-Kat Beton Dökümü	Level -1-Tavan Kaplama İmalatı
Level -1-Duvar İmalatı	
Level 2-Kat Kalıp İmalatı	
Level 2-Kat Demir İmalatı	
Level 2-Kat Beton Dökümü	
Level 3-Çatı İmalatı	
Level 0-Elektrik İşleri	
Level -1-Cephe İşleri	
Level 1-Cephe İşleri	
Level 1-Çevre Düzenleme ve Peyzaj İşleri	
Level 1-Duvar İmalatı	
Level 2-Duvar İmalatı	
Level 0-Mekanik İşler	
Level 1-Kapı İmalatı	
Level 2-Kapı İmalatı	
Level -1-Döşeme Yalıtım İmalatı	
Level 1-Duvar Sıva İmalatı	
Level 2-Duvar Sıva İmalatı	
Level 1-Döşeme Yalıtım İmalatı	
Level 2-Döşeme Yalıtım İmalatı	

Tablo 13'ün devamı

Level -1-Döşeme Şap İmalatı	
Level 1-Döşeme Şap İmalatı	
Level 2-Döşeme Şap İmalatı	
Level 1-Duvar Seramik İmalatı	
Level 2-Duvar Seramik İmalatı	
Level -1-Duvar Boya İmalatı	
Level 1-Asma Tavan İmalatı	
Level 1-Duvar Boya İmalatı	
Level 2-Asma Tavan İmalatı	
Level -1-Döşeme Seramik İmalatı	
Level 1-Döşeme Seramik İmalatı	
Level 1-Döşeme Laminant İmalatı	
Level 2-Döşeme Laminant İmalatı	
Level 2-Döşeme Seramik İmalatı	

Tablo 13'te, tez kapsamında geliştirilen kodun çıktı dosyasını örnekleyecek biçimde, sıralanan aktiviteler ve tahmin edilemeyen aktiviteler başlıkları altında iki ayrı sütun hâlinde sunulmaktadır. Sol tarafta, kod tarafından ardıl ilişkileri belirlenip sıralamaya dâhil edilen aktiviteler yer alırken, sağ sütunda modelin çeşitli nedenlerle tanıyamadığı, mevcut proje yapısıyla eşleştiremediği ya da ardıl tahmininde bulunamadığı aktiviteler listelenmektedir. Böylelikle, hangi faaliyetlerin planlama sürecine dâhil edildiği ve hangi faaliyetlerin kullanıcı tarafından manuel inceleme veya ek düzenleme gerektirdiği kolaylıkla ayırt edilebilmekte; aynı zamanda 3B modele tam olarak yansıtılmayan veya veri setinde eksik kalmış ilişkilerin tespit edilmesi de mümkün olmaktadır. Tablo 13 incelendiğinde, 46 aktivitenin sıralandığı, 13 aktivitenin de tahmin edilemeyenler kısmında bulunduğu görülmektedir. 13 aktivite arasında, başlangıç aktivitesinin "Level -1-Grobeton İmalatı" ve bodrum kat sayısının 1 seçilmesinden dolayı grobeton imalatından önce bulunan "Yer Teslimi" aktivitesi "Tahmin Edilemeyen Aktiviteler" arasında yer almıştır. Ayrıca, Tablo 12'de verilen aktivite-ardıl ilişkilerinde ardıl olarak tahmin edilen "Level -2-Kat Beton Dökümü", "Level -2-Duvar İmalatı" gibi aktivitelerle, P9 projesinin aktiviteleri arasında yer almayan ancak eğitim setinden kaynaklı olarak tahmin listelerinde olma ihtimali bulunan "Level 1-Grobeton İmalatı" veya "Level 1-Temel Kalıp İmalatı" gibi aktivitelerin de gerçekleştirilen filtrelerle engellendiği ve iş programında yer almadığı görülmüştür.

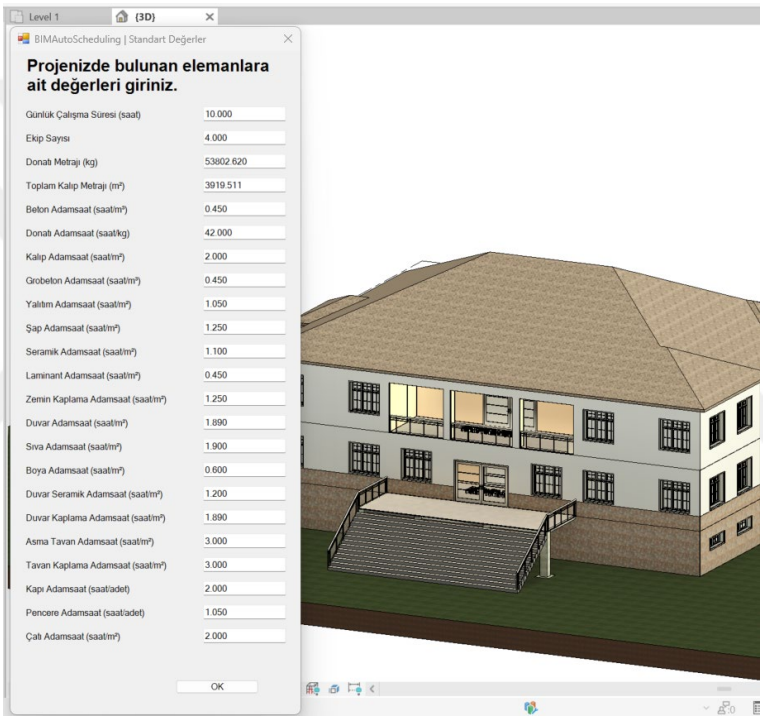
Literatürde ML modelleriyle aktivite sıralaması tahmini üzerine yapılan çalışmalara bakıldığında (Amer & Golparvar-Fard, 2019; Amer & Golparvar-Fard, 2021; Amer vd., 2022; Prieto vd., 2023; Hong vd., 2023), yalnızca öncül-ardıl ilişkilerindeki tahmin doğruluğunun raporlandığı, ancak sonuç olarak elde edilen aktiviteler diziliminin kullanıcıya bütüncül bir çizelge veya düzenlenebilir bir plan biçiminde sunulmadığı görülmektedir. Öte yandan, BIM modelleriyle aktiviteleri ilişkilendiren çalışmalarda ise (Wang & Rezazadeh Azar, 2019; ElMenshawy & Marzouk, 2021), araştırmacıların varsayımlarına dayalı sıralama kabulleri yapılmış olup, kullanıcı müdahalesine veya ayarlamasına imkân tanınmadığı anlaşılmaktadır. Bu bağlamda, tez kapsamında önerilen yaklaşım, ML modelleriyle elde edilen aktivite sıralamasını dış ortama aktarılabilecek bir dosya formatında sunmakta ve kullanıcının ihtiyaç duyduğu durumlarda bu sıralamayı değiştirebilmesine imkân veren etkileşimli bir yapı sağlamaktadır. Böylece, literatürde genellikle "tahmin doğruluğu" boyutuyla sınırlı kalan yaklaşımlara ek olarak, gerçek projelerdeki uyarlanabilirlik ve kullanıcı deneyimini geliştiren bütünleşik bir çizelgeleme çözümü sunulmaktadır. Bu yönüyle çalışma, yalnızca ardıl-öncül tahminlerini rapor etmek yerine, ortaya çıkan aktivitelerin dizilimini sahada geçerli olacak biçimde düzenleme fırsatı yaratarak literatüre katkı sunmaktadır.

3.2. Revit ve Navisworks Eklentilerinin Değerlendirilmesi

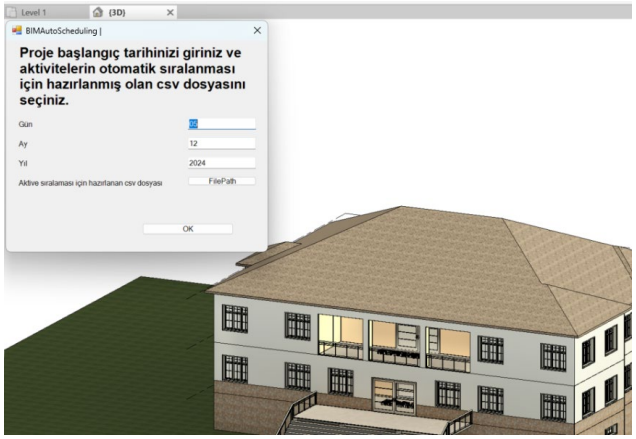
Bu aşamada, 3B olarak modellenen örnek proje üzerinde Revit ve Navisworks eklentilerinin çalışma süreçleri ile eklentilerden elde edilen verilerin ve çıktıların doğruluğu niteliksel olarak incelenmiştir. İlk olarak, "BIM Auto Scheduling" adıyla oluşturulan Revit eklentisine ilişkin çıktılar değerlendirilmiştir. ML modellerinde kullanılan nicel analiz yerine, önceki bölümlerde ayrıntıları aktarılan Dynamo kodlarının doğru çalışıp çalışmadığı ve Revit modeli üzerinde gerekli parametreleri ve bilgileri üretip üretmediği kontrol edilmiştir. Çalışma kapsamında kullanılan örnek proje, tüm yapı elemanlarını içerdiği için, eklentinin başarılı bir şekilde çalışması sonucunda bu elemanlara ait tüm gerekli bilgilerin doğru biçimde üretilmesi beklenmiştir.

"Yapılan Çalışmalar" kısmında da açıklandığı üzere, eklenti yürütüldüğünde Dynamo kodları sırasıyla çalışmakta ve ilk olarak kullanıcının varsayılan değerleri onaylayabileceği veya güncelleyebileceği Şekil 92'de gösterildiği gibi bir arayüz açılmaktadır. Bu pencerede, Dynamo aracılığıyla atanan adam-saat değerleri, ekip sayısı,

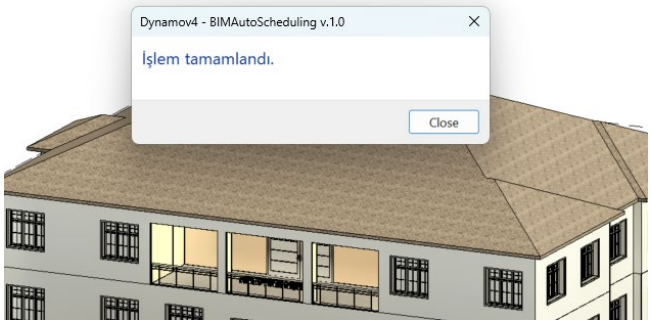
günlük çalışma süresi, yaklaşık donatı metrajı ve toplam kalıp metrajı gibi bilgilere ulaşılmakta ve bu bilgilerin Tablo 5'te verildiği gibi oluşturuldukları görülmektedir. Bu değerler kullanıcı tarafından gerektiğinde düzenlenebilmekte, ardından eklentinin çalışmasına devam edilmektedir. Kullanıcı, ilk aşamadaki kontrolleri tamamladığında, Şekil 93'te verilen ikinci bir arayüz penceresi açılarak iş programının başlangıç tarihi ile ML modelinden elde edilen aktivite sıralama dosyasının tanımlanması istenmektedir. Bu bilgiler girildikten sonra eklenti çalışmayı sürdürmekte ve tamamlandığında üçüncü bir bilgilendirme ekranı (Şekil 94) ile eklentinin tüm işlemleri bitirdiği görüntülenmektedir.



Şekil 92. BIM Auto Scheduling eklentisi varsayılan değerler ekranı



Şekil 93. BIM Auto Scheduling eklentisi iş programı oluşturma ekranı



Şekil 94. BIM Auto Scheduling eklentisi tamamlanma ekranı

Eklentinin bu aşamalarının sonucunda "Örnek Proje.csv" isimli bir iş programı dosyası ile "Örnek Proje SearchSets.xml" isimli bir Navisworks Search Sets dosyası üretilmektedir. Tablo 14'te elde edilen nihai iş programı dosyası gösterilmektedir. İş programı, Navisworks'te 4B zamanlama simülasyonu oluşturmak amacıyla aktivite adı, tipi, başlangıç ve bitiş tarihlerini içerecek biçimde yapılandırılmıştır.

Tablo 14. BIM Auto Scheduling eklentisi ile elde edilen iş programı

Task ID	Task Name	Task Type	Planned Start	Planned End
1	Level -1-Grobeton İmalatı	Construct	09/12/2024	09/12/2024
2	Level -1-Temel Kalıp İmalatı	Temporary	10/12/2024	10/12/2024
3	Level -1-Temel Demir İmalatı	Construct	11/12/2024	13/12/2024
4	Level -1-Temel Beton Dökümü	Construct	14/12/2024	16/12/2024
5	Level -1-Kat Kalıp İmalatı	Temporary	17/12/2024	14/02/2025
6	Level -1-Kat Demir İmalatı	Construct	15/02/2025	03/03/2025
7	Level -1-Kat Beton Dökümü	Construct	04/03/2025	06/03/2025
8	Level 1-Kat Kalıp İmalatı	Temporary	07/03/2025	14/05/2025
9	Level 1-Kat Demir İmalatı	Construct	15/05/2025	02/06/2025
10	Level 1-Kat Beton Dökümü	Construct	03/06/2025	04/06/2025
11	Level -1-Duvar İmalatı	Construct	05/06/2025	22/06/2025
12	Level 2-Kat Kalıp İmalatı	Temporary	23/06/2025	30/08/2025
13	Level 2-Kat Demir İmalatı	Construct	31/08/2025	18/09/2025
14	Level 2-Kat Beton Dökümü	Construct	19/09/2025	20/09/2025
15	Level 3-Çatı İmalatı	Construct	21/09/2025	18/10/2025
16	Level 1-Duvar İmalatı	Construct	19/10/2025	07/11/2025
17	Level 2-Duvar İmalatı	Construct	08/11/2025	24/11/2025
18	Level -1-Döşeme Yalıtım İmalatı	Construct	25/11/2025	25/11/2025

Tablo 14'ün devamı

19	Level 1-Döşeme Yalıtım İmalatı	Construct	26/11/2025	26/11/2025
20	Level 2-Döşeme Yalıtım İmalatı	Construct	27/11/2025	27/11/2025
21	Level -1-Duvar Sıva İmalatı	Construct	28/11/2025	25/01/2026
22	Level 1-Duvar Sıva İmalatı	Construct	26/01/2026	05/03/2026
23	Level 2-Duvar Sıva İmalatı	Construct	06/03/2026	17/04/2026
24	Level -1-Döşeme Şap İmalatı	Construct	18/04/2026	18/04/2026
25	Level 1-Döşeme Şap İmalatı	Construct	19/04/2026	19/04/2026
26	Level 2-Döşeme Şap İmalatı	Construct	20/04/2026	20/04/2026
27	Level -1-Duvar Seramik İmalatı	Construct	21/04/2026	24/04/2026
28	Level 1-Duvar Seramik İmalatı	Construct	25/04/2026	03/05/2026
29	Level 2-Duvar Seramik İmalatı	Construct	04/05/2026	06/05/2026
30	Level -1-Asma Tavan İmalatı	Construct	07/05/2026	08/05/2026
31	Level 1-Asma Tavan İmalatı	Construct	09/05/2026	11/05/2026
32	Level 2-Asma Tavan İmalatı	Construct	12/05/2026	13/05/2026
33	Level -1-Duvar Boya İmalatı	Construct	14/05/2026	28/05/2026
34	Level 1-Duvar Boya İmalatı	Construct	29/05/2026	05/06/2026
35	Level 2-Duvar Boya İmalatı	Construct	06/06/2026	15/06/2026
36	Level -1-Tavan Kaplama İmalatı	Construct	16/06/2026	11/07/2026
37	Level 1-Tavan Kaplama İmalatı	Construct	12/07/2026	05/08/2026
38	Level 2-Tavan Kaplama İmalatı	Construct	06/08/2026	01/09/2026
39	Level -1-Pencere İmalatı	Construct	02/09/2026	02/09/2026
40	Level 1-Pencere İmalatı	Construct	03/09/2026	03/09/2026
41	Level 2-Pencere İmalatı	Construct	04/09/2026	04/09/2026
42	Level -1-Döşeme Seramik İmalatı	Construct	05/09/2026	15/09/2026
43	Level 1-Döşeme Seramik İmalatı	Construct	16/09/2026	25/09/2026
44	Level 2-Döşeme Seramik İmalatı	Construct	26/09/2026	30/09/2026
45	Level 1-Döşeme Laminant İmalatı	Construct	01/10/2026	01/10/2026
46	Level 2-Döşeme Laminant İmalatı	Construct	02/10/2026	02/10/2026
47	Level 1-Zemin Kaplama İmalatı	Construct	03/10/2026	04/10/2026
48	Level 2-Zemin Kaplama İmalatı	Construct	05/10/2026	10/10/2026
49	Level -1-Duvar Kaplama İmalatı	Construct	11/10/2026	21/10/2026
50	Level 1-Duvar Kaplama İmalatı	Construct	22/10/2026	03/11/2026
51	Level 2-Duvar Kaplama İmalatı	Construct	04/11/2026	14/11/2026
52	Level -1-Kapı İmalatı	Construct	15/11/2026	15/11/2026
53	Level 1-Kapı İmalatı	Construct	16/11/2026	17/11/2026
54	Level 2-Kapı İmalatı	Construct	18/11/2026	18/11/2026

Tablo 14 ile verilen iş programı incelendiğinde, kalıp imalatı aktivitelerinin "Temporary" olarak işaretlendiği, başlangıç aktivitesi ve tarihinin doğru belirlendiği ve hesaplanan aktivite sürelerinin (İmalat Süresi parametreleri) başlangıç-bitiş tarihlerini oluşturmak üzere doğru aktarılmış olduğu anlaşılmaktadır.

İş programının yanı sıra eklenti sonucu oluşturulan ve Şekil 95'te kısa bir bölümü gösterilen Search Sets dosyasının, Navisworks ortamında Aktivite Adı, Aktivite Adı-Kalıp, Aktivite Adı-Demir gibi farklı imalat türlerini yansıtan parametrelere göre filtreleme yapacak biçimde düzenlendiği görülmektedir. Dosya incelendiğinde, bütün elemanların filtrelendiği ve bu nedenle de dosyanın doğru oluşturulduğu gözlenmiştir.

```
<?xml version="1.0" encoding="UTF-8"?>
<exchange xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://download.autodesk.com/us/navisworks/schemas/nw-
exchange-12.0.xsd">                                                                    1
<selectionsets>
```

```
  <selectionset name="Level -1-Temel Beton Dökümü" guid="114d567c-8d1f-4f38-be47-
e4bfece4145c">
    <findspec mode="all" disjoint="0">
     <conditions>
      <condition test="equals" flags="10">
       <category>
        <name internal="LcRevitData_Element">Custom</name>
       </category>
       <property>
        <name internal="lcldrevit_parameter_7556272">Aktivite Adı</name>          2
       </property>
       <value>
        <data type="wstring">Level  -1-Temel Beton Dökümü</data>
       </value>
      </condition>
     </conditions>
     <locator>/</locator>
    </findspec>
  </selectionset>
      :
      :
```

```
  <selectionset name="Level  -1-Temel Kalıp İmalatı" guid="114d567c-8d1f-4f38-be47-e4bfece4145c">
    <findspec mode="all" disjoint="0">
     <conditions>
      <condition test="equals" flags="10">
       <category>
        <name internal="LcRevitData_Element">Custom</name>
       </category>
       <property>
        <name internal="lcldrevit_parameter_7556272">Aktivite Adı-Kalıp</name>    3
       </property>
       <value>
        <data type="wstring">Level  -1-Temel Kalıp İmalatı</data>
       </value>
      </condition>
     </conditions>
     <locator>/</locator>
    </findspec>
  </selectionset>
      :
      :
```

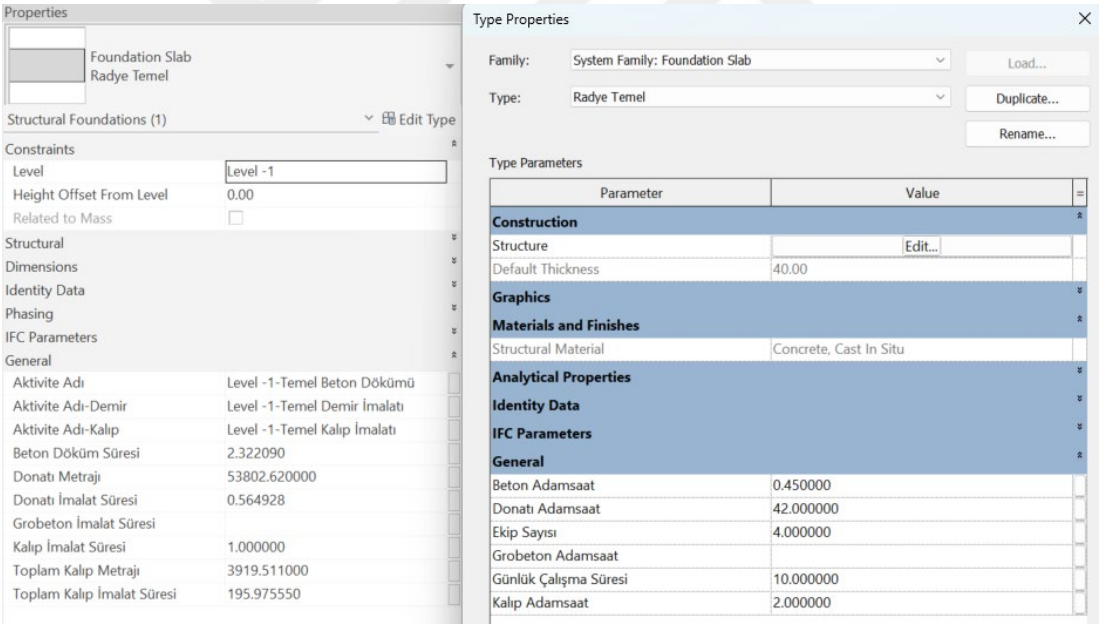
```
  <selectionset name="Level -1-Temel Demir İmalatı" guid="114d567c-8d1f-4f38-be47-
e4bfece4145c">
    <findspec mode="all" disjoint="0">
     <conditions>
      <condition test="equals" flags="10">
       <category>
        <name internal="LcRevitData_Element">Custom</name>
       </category>
       <property>
        <name internal="lcldrevit_parameter_7556272">Aktivite Adı-Demir</name>    4
       </property>
       <value>
        <data type="wstring">Level  -1-Temel Demir İmalatı</data>
       </value>
      </condition>
     </conditions>
     <locator>/</locator>
    </findspec>
  </selectionset>
      :
      :
```

Şekil 95. Örnek Proje SearchSets.xml dosyasından örnek bir bölüm

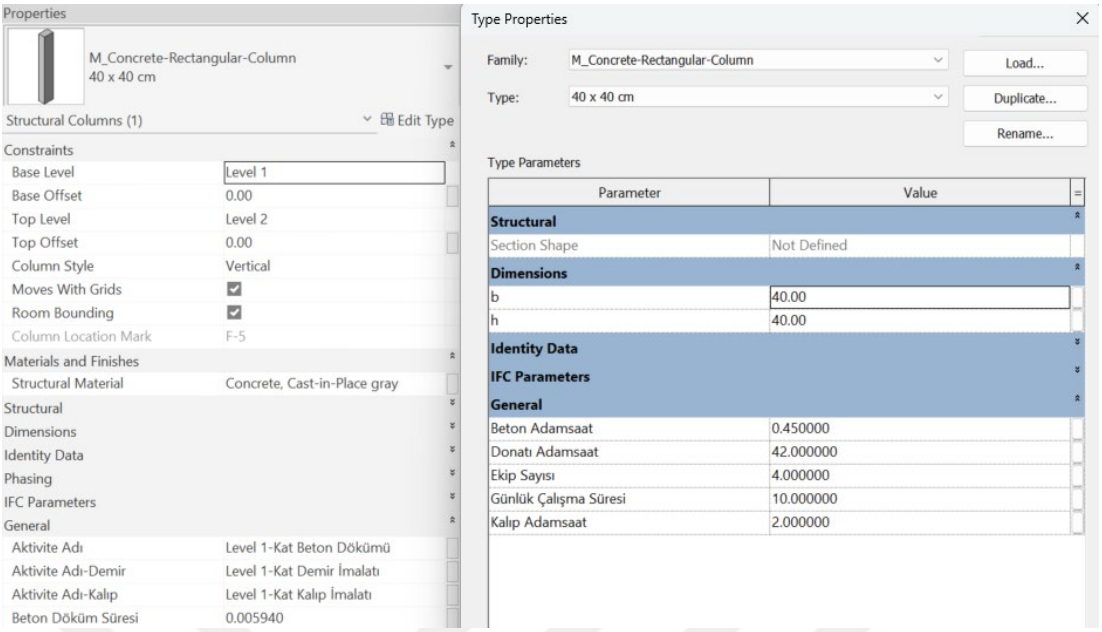
Şekil 95'te görüldüğü üzere kalıp ile ilgili aktiviteler Aktivite Adı-Kalıp parametresi, demir/donatı ile ilgili aktiviteler ise Aktivite Adı-Demir parametresi altında yer almaktadır. Ayrıca, Navisworks formatına uygun olacak şekilde başlıkların da eklendiği görülmektedir.

İş programı ve Search Sets dosyalarının doğru formatlarda oluşturulduğu görülmektedir ancak bu dosyaların oluşturulmasından önce eklentinin çalıştırdığı Dynamo kodlarının da doğru çalışıp çalışmadığı kontrol edilmelidir. Bunun için Dynamo kodlarından elde edilen çıktılar sırayla incelenmiş ve ilk olarak proje parametrelerinin oluşturulması aşaması kontrol edilmiştir.

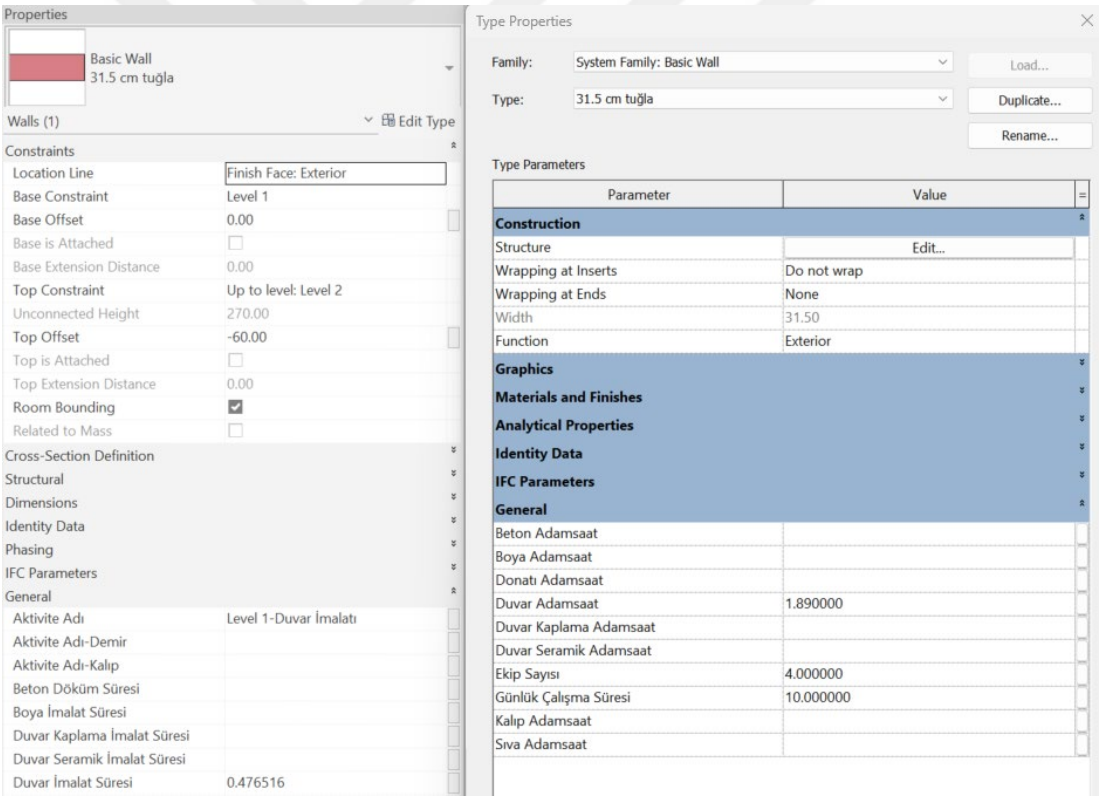
Bu kapsamda, Tablo 4'te listelenen proje parametrelerinin tamamının, belirtilen uygulama türünde, elemanlarda, türde ve grup altında doğru bir şekilde oluşturulduğu görülmüştür. Şekil 96'da farklı kategorilere ait elemanlarda oluşturulan parametrelere örnekler gösterilmektedir.



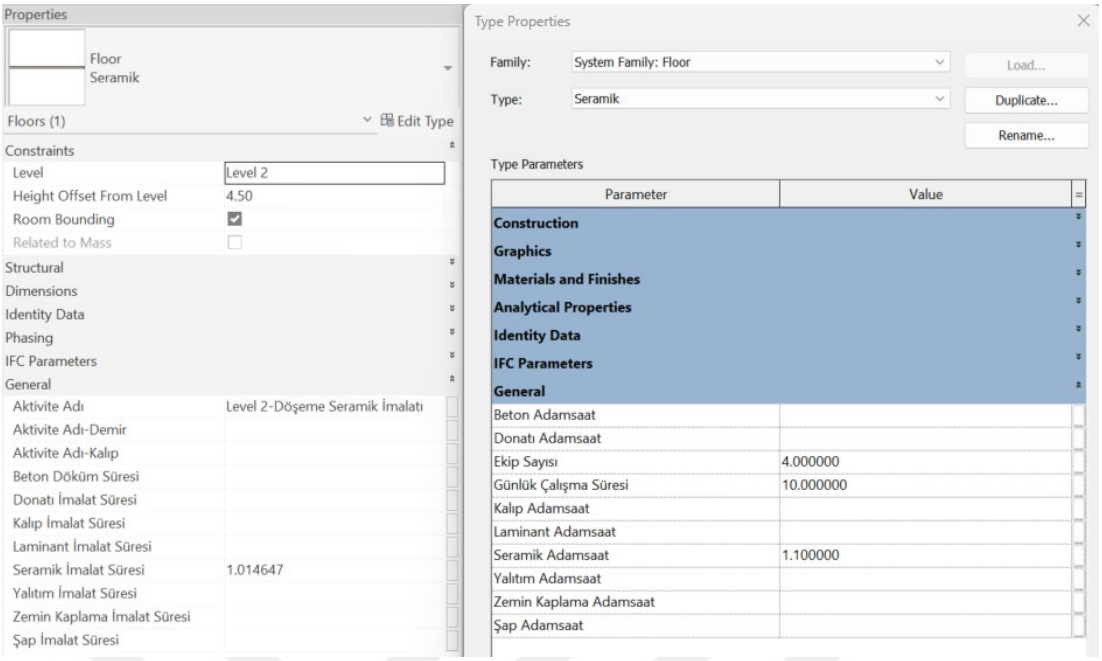
(a)


Properties
M_Concrete-Rectangular-Column
40 x 40 cm
Structural Columns (1)
Edit Type
Constraints
Base Level Level 1
Base Offset 0.00
Top Level Level 2
Top Offset 0.00
Column Style Vertical
Moves With Grids
Room Bounding
Column Location Mark F-5
Materials and Finishes
Structural Material Concrete, Cast-in-Place gray
Structural
Dimensions
Identity Data
Phasing
IFC Parameters
General
Aktivite Adı Level 1-Kat Beton Dökümü
Aktivite Adı-Demir Level 1-Kat Demir İmalatı
Aktivite Adı-Kalıp Level 1-Kat Kalıp İmalatı
Beton Döküm Süresi 0.005940
Type Properties
Family: M_Concrete-Rectangular-Column
Type: 40 x 40 cm
Load...
Duplicate...
Rename...
Type Parameters
Parameter
Value
Structural
Section Shape Not Defined
Dimensions
b 40.00
h 40.00
Identity Data
IFC Parameters
General
Beton Adamsaat 0.450000
Donatı Adamsaat 42.000000
Ekip Sayısı 4.000000
Günlük Çalışma Süresi 10.000000
Kalıp Adamsaat 2.000000

(b)


Properties
Basic Wall
31.5 cm tuğla
Walls (1)
Edit Type
Constraints
Location Line Finish Face: Exterior
Base Constraint Level 1
Base Offset 0.00
Base is Attached
Base Extension Distance 0.00
Top Constraint Up to level: Level 2
Unconnected Height 270.00
Top Offset -60.00
Top is Attached
Top Extension Distance 0.00
Room Bounding
Related to Mass
Cross-Section Definition
Structural
Dimensions
Identity Data
Phasing
IFC Parameters
General
Aktivite Adı Level 1-Duvar İmalatı
Aktivite Adı-Demir
Aktivite Adı-Kalıp
Beton Döküm Süresi
Boya İmalat Süresi
Duvar Kaplama İmalat Süresi
Duvar Seramik İmalat Süresi
Duvar İmalat Süresi 0.476516
Type Properties
Family: System Family: Basic Wall
Type: 31.5 cm tuğla
Load...
Duplicate...
Rename...
Type Parameters
Parameter
Value
Construction
Structure Edit...
Wrapping at Inserts Do not wrap
Wrapping at Ends None
Width 31.50
Function Exterior
Graphics
Materials and Finishes
Analytical Properties
Identity Data
IFC Parameters
General
Beton Adamsaat
Boya Adamsaat
Donatı Adamsaat
Duvar Adamsaat 1.890000
Duvar Kaplama Adamsaat
Duvar Seramik Adamsaat
Ekip Sayısı 4.000000
Günlük Çalışma Süresi 10.000000
Kalıp Adamsaat
Sıva Adamsaat

(c)



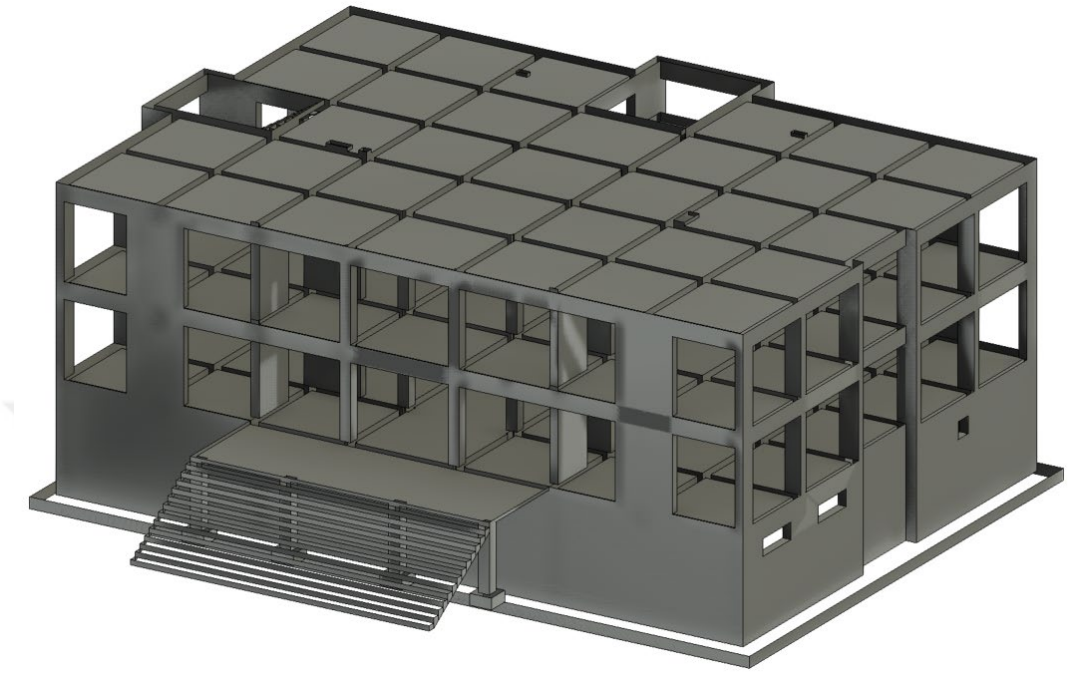
(d)

Şekil 96. Oluşturulan proje parametresi örnekleri

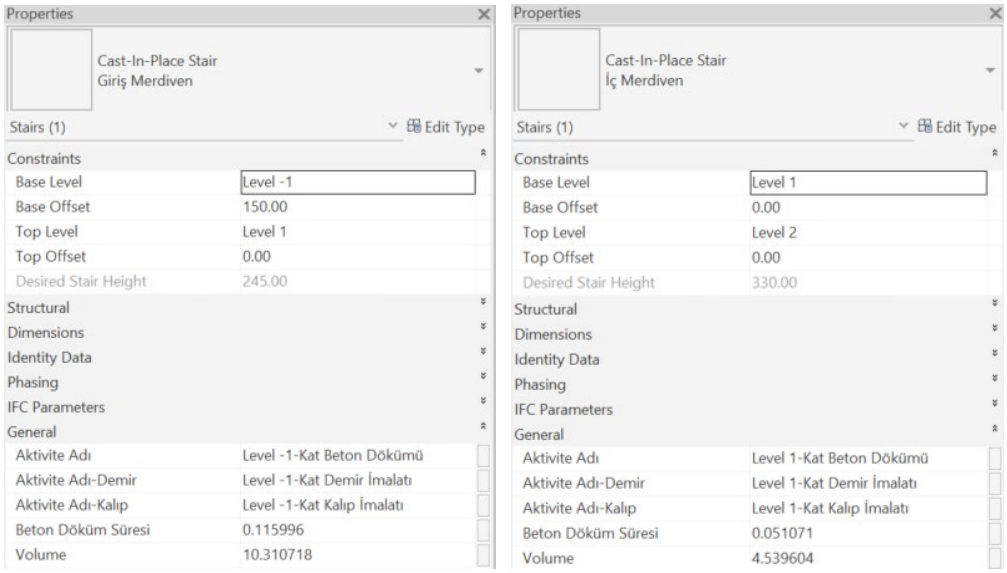
Şekil 96(a)'da temel elemanı için sol tarafta "instance" yani eleman bazlı ve sağ tarafta ise "type" yani elemanın türü tabanlı oluşturulan parametreler gösterilmektedir. Şekil 96(b), (c) ve (d)'de ise sırasıyla kolon, mimari duvar ve döşeme seramik elemanları için benzer şekilde oluşturulmuş olan proje parametreleri gösterilmektedir. Daha önce de belirtildiği üzere, parametre oluşturulması kategoriler üzerinden yapıldığı için aynı kategoride yer alan ancak diğer elemanlara ait parametreler de oluşturulmuştur. Ancak, Şekil 96'da görüldüğü üzere parametre oluşturulsa bile Dynamo kodlarında elemanlara sadece gerekli olan değerler atanmıştır.

Proje parametrelerinin oluşturulmasından sonra, Şekil 92'deki değerler ile ilgili olan arayüzün açılmasına kadar, adam-saat değerleri, günlük çalışma süresi, ekip sayısı ve yaklaşık donatı metrajının elemanlara atanması, kalıp metrajının ve merdiven beton hacminin hesaplanması ve gerekli parametrelere atanması işlemleri gerçekleştirilmiştir. Şekil 97'de ise kalıp metrajı hesabı için geliştirilen Dynamo kodu ile oluşturulan yüzeylerin 3B görünümü verilmektedir. Bu görünümden, döşemelerde şaft boşluklarının bırakıldığı ve yan yüzeylerinin kalıp ile kapatıldığı, merdiven boşluklarının bırakıldığı, kolon-kiriş-döşeme gibi eleman birleşimlerinde kalıp yapılmadığı, temel ve kolon gibi elemanların alt ve üst yüzeylerinin kalıp ile kapatılmadığı gibi durumların doğru şekilde oluşturulduğu görülmektedir. Örnek proje için donatı metrajı 53802.60 kg ve toplam kalıp

metrajı 3919.511 m^2 olarak elde edilmiştir. Ayrıca, merdivenler için de beton metrajları elde edilip Şekil 98'de gösterildiği gibi merdiven elemanlarına atanmıştır.



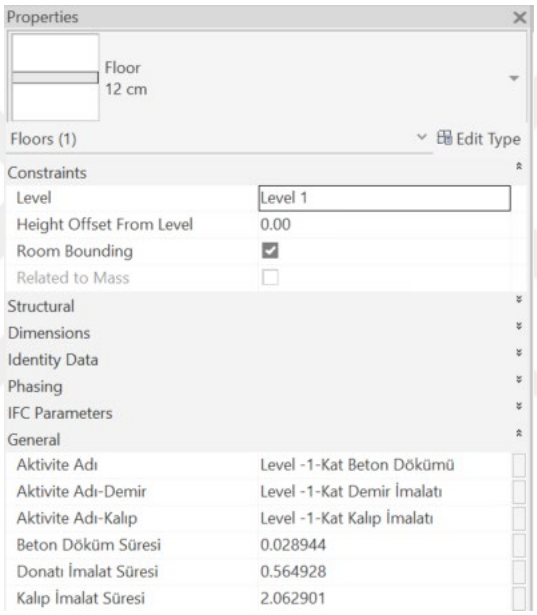
Şekil 97. 3B kalıp gösterimi



Şekil 98. Merdiven beton hacmi

Varsayılan değerlerin, donatı metrajının ve kalıp metrajının onaylanması veya düzenlenmesinden sonraki aşamada, Dynamo kodları ilk olarak "2.4.5. Aktivite Sürelerinin Hesaplanması" başlığında açıklanan yöntemle aktivite sürelerini hesaplamakta ve bu

süreleri elemanlara parametre olarak gün cinsinde kaydetmektedir. Şekil 96 ve 98'de "Beton Döküm Süresi", "Kalıp İmalat Süresi", "Donatı İmalat Süresi", "Duvar İmalat Süresi" ve "Seramik İmalat Süresi" için örnekler, Şekil 99'da ise döşemelere ait "Kalıp İmalat Süresi" ve "Donatı İmalat Süresi" parametreleri gösterilmektedir. Daha önce ayrıntıları açıklandığı üzere, bu iki aktivite için elde edilen metraj değerleri temel elemanlarına atanmıştır ancak aktivite süreleri kat sayısına bölünerek döşeme elemanlarına atanmıştır. Aktivite süreleri, Revit içinde ondalıklı sayı olarak tutulmasına karşın, iş programının dışa aktarımında tam sayı biçimine dönüştürülmektedir. Böylece bir katın tüm elemanlarına ait imalat süreleri toplanarak gerçek aktivite süresi hesaplanabilmektedir.



Şekil 99. Döşeme elemanları için değer atanan proje parametreleri

Aktivite süreleri elde edildikten sonra, metraj listeleri oluşturulmaktadır. Bu listeler Revit'ten iş programı oluşturulması için gerekli değildir ancak parametrelerin kontrolü, kullanıcıya hangi değerlerin esas alındığını doğrulama imkânı sunması bakımından önem taşımaktadır. Tablo 15-17'de sırasıyla kiriş, boya ve duvar kaplama elemanlarına ait örnek metraj listeleri yer almaktadır. Listelerin büyük olması nedeniyle, kat bazlı gruplama yapılarak gösterim sağlanmıştır. Listelerde hacim, alan, adam-saat, eleman sayısı, imalat süresi, beton döküm süresi gibi kritik parametrelerin toplamlarının otomatik olarak hesaplandığı görülmektedir.

Tablo 15. Kiriş metraj listesi

KİRİŞ METRAJI											
Aktivite Adı	**Aktivite Adı-Kalıp**	**Aktivite Adı-Demir**	**Count**	**Family and Type**	**Volume**	**Length**	**Reference Level**	**Günlük Çalışma Süresi**	**Ekip Sayısı**	**Beton Adamsaat**	**Beton Döküm Süresi**
Level -1-Kat Beton Dökümü	Level -1-Kat Kalıp İmalatı	Level -1-Kat Demir İmalatı	43	M_Concrete-Rectangular Beam: 30 x 60 cm	27.33 m³	<varies>	Level 1	10	4	19.35	0.307456
Level 1: 43					**27.33 m³**					**19.35**	**0.307456**
Level 1-Kat Beton Dökümü	Level 1-Kat Kalıp İmalatı	Level 1-Kat Demir İmalatı	70	M_Concrete-Rectangular Beam: 30 x 60 cm	41.33 m³	<varies>	Level 2	10	4	31.May	0.46493
Level 2: 70					**41.33 m³**					**31.May**	**0.46493**
Level 2-Kat Beton Dökümü	Level 2-Kat Kalıp İmalatı	Level 2-Kat Demir İmalatı	70	M_Concrete-Rectangular Beam: 30 x 60 cm	41.40 m³	<varies>	Level 3	10	4	31.May	0.46574
Level 3: 70					**41.40 m³**					**31.May**	**0.46574**
Grand total					**110.06 m³**					**82.35**	**1.238.125**

Tablo 16. Boya metraj listesi

BOYA METRAJI											
Aktivite Adı	**Count**	**Family and Type**	**Base Constraint**	**Width**	**Volume**	**Area**	**Length**	**Günlük Çalışma Süresi**	**Ekip Sayısı**	**Boya Adamsaat**	**Boya İmalat Süresi**
Level -1-Duvar Boya İmalatı	141	Basic Wall: Boya	Level -1	1	<varies>	975 m²	<varies>	10	4	84.6	14.624.269
Level -1: 141						**975 m²**				**84.6**	**14.624.269**
Level 1-Duvar Boya İmalatı	116	Basic Wall: Boya	Level 1	1	<varies>	525 m²	<varies>	10	4	69.6	7.879.776
Level 1: 116						**525 m²**				**69.6**	**7.879.776**
Level 2-Duvar Boya İmalatı	139	Basic Wall: Boya	Level 2	1	<varies>	637 m²	<varies>	10	4	83.4	95.494
Level 2: 139						**637 m²**				**83.4**	**95.494**
Grand total						**2137 m²**				**237.6**	**32.053.446**

Tablo 17. Duvar kaplama metraj listesi

DUVAR KAPLAMA METRAJI											
Aktivite Adı	**Count**	**Family and Type**	**Base Constraint**	**Width**	**Volume**	**Area**	**Length**	**Günlük Çalışma Süresi**	**Ekip Sayısı**	**Duvar Kaplama Adamsaat**	**Duvar Kaplama İmalat Süresi**
Level -1-Duvar Kaplama İmalatı	18	Basic Wall: Taş Kaplama	Level -1	3	<varies>	218 m²	<varies>	10	4	34.02	10.292.814
Level -1: 18						**218 m²**				**34.02**	**10.292.814**
Level 1-Duvar Kaplama İmalatı	42	<varies>	Level 1	3	<varies>	255 m²	<varies>	10	4	79.38	12.055.476
Level 1: 42						**255 m²**				**79.38**	**12.055.476**
Level 2-Duvar Kaplama İmalatı	22	Basic Wall: Giydirme Cephe Kaplama	Level 2	3	<varies>	230 m²	<varies>	10	4	41.58	1.087.159
Level 2: 22						**230 m²**				**41.58**	**1.087.159**
Grand total						**703 m²**				**154.98**	**3.321.988**

İlerleyen aşamalarda “Aktivite Adı”, “Aktivite Adı-Kalıp” ve “Aktivite Adı-Demir” gibi parametreler kullanılarak, elemanın bulunduğu kat seviyesi, eleman adı ve ilgili imalatın yer aldığı bir sistematikle aktivite isimleri oluşturulmaktadır. Tablo 14, söz konusu aktivite isimlerinin doğru kurgulandığını göstermektedir. Kiriş ve döşeme elemanlarının kat seviyelerinin yalnızca isimlendirme aşamasında bir seviye düşürülmesinin de doğru olarak çalıştığı ve bir sorunla karşılaşılmadığı tespit edilmiştir.

Bütün bu işlem adımlarının tamamlanmasının ardından, ikinci arayüzde ML modelinden elde edilen aktivite sıralama dosyası ve BIM modelindeki elemanlara ait Dynamo kodları ile oluşturulan bilgiler ilişkilendirilerek “Örnek Proje.csv” isimli iş programı dosyası dış ortama aktarılmaktadır. Modeldeki tüm elemanların iş programında yer aldığı, aktivite sürelerinin kat bazında tam sayıya dönüştürüldüğü ve öncül-ardıl aktiviteler arasında FS ilişkileriyle başlangıç-bitiş tarihlerinin otomatik biçimde hesaplandığı, nitel değerlendirmeyle doğrulanmıştır. Dolayısıyla, Revit için geliştirilen BIM Auto Scheduling eklentisinin, Dynamo kodlarını belirlenen sırayla ve sorunsuz çalıştırmış olması, “2.5. Revit Eklentisi Oluşturulması” başlığı ile anlatılan C# ile modelleme sürecinin de herhangi bir hata olmadan oluşturulmuş olduğunu göstermektedir.

Bu değerlendirmelere ek olarak, tez kapsamında 3B modellenen yapıyla, benzer kat sayısı ve kat alanına sahip başka bir anaokulunun Excel formatında iş programı ÇŞİDB’den elde edilmiş ve iş programları içerdikleri aktiviteler ve aktivite sıralamaları dikkate alınarak karşılaştırılmıştır. ÇŞİDB’den alınan iş programında, aktiviteler “İnşaat, Mekanik, Elektrik, Çevre ve Altyapı İşleri – İnşaat” olmak üzere dört iş kırılımı altında gruplandırılmıştır. Tez kapsamında inşaat işleri dikkate alındığından, iş programının İnşaat kısmının aktivite süresi, başlangıç ve bitiş tarihleri Tablo 18’de ve Gantt şeması ise Ek 12’de gösterilmektedir. İş programı incelendiğinde; FS dışında da aktivite ilişkilerinin kurulduğu ve toplam yapım süresinin 400 gün olduğu görülmektedir.

Tablo 18. ÇŞİDB'den elde edilen iş programı

İş Programı				Aktivite Süresi (gün)	Başlangıç Tarihi	Bitiş Tarihi
1	İnşaat	A-Su Basman Altı Grubu	Kazı Dolgu Grobeton ve Yalıtım İmalatları	16	23.09.2023	08.10.2023
2			Su Basman Altı Betonarme İmalatları	28	09.10.2023	06.11.2023
3		B-Kat Betonarme Grubu		54	07.11.2023	30.12.2023
4		C-Çatı Grubu		20	01.01.2024	20.01.2024
5		D-Duvar Grubu		68	07.01.2024	14.03.2024
6		E-Kaplamalar Grubu	Döşeme Kaplamaları	98	15.04.2024	22.07.2024
7			Duvar Kaplamaları	90	15.03.2024	14.06.2024
8			Tavan Kaplamaları	112	09.04.2024	30.07.2024
9		F-Cephe Grubu		110	19.03.2024	08.07.2024
10		G-Doğramalar		106	15.05.2024	30.08.2024
11		H-Muhtelif İmalatları		78	19.06.2024	06.09.2024

Çalışma kapsamında elde edilen iş programı ile ÇŞİDB'den alınan örnek iş programı karşılaştırıldığında, her iki planın temelde benzer imalat faaliyetlerini (temel, betonarme, duvar, kaplama vb.) içerdiği görülmektedir. Ancak planlama yaklaşımı, aktivite detaylandırma düzeyi ve faaliyetler arası ilişki türleri bakımından farklılıkları bulunmaktadır. Söz konusu farklılıklar, proje yönetiminde toplam süre tahminleri, kaynak planlaması ve kontrol süreçleri üzerinde doğrudan etkiye sahiptir.

Öncelikle, çalışma kapsamında elde edilen iş programının detaylandırma seviyesi örnek iş programına göre daha yüksektir. Bu programda her bir imalat kalemi (örneğin, temel kalıp, temel demir, kat kalıbı, kat demiri, beton dökümü, duvar ve döşeme kaplamaları gibi) ayrı ayrı ve kat bazında (Level -1, Level 1, Level 2, vb.) tanımlanmıştır. Böylesine ayrıntılı bir yapı, proje süresince yürütülecek işlerin planlanmasında ve sahada ilerlemenin takibinde avantaj sağlamaktadır. Buna karşılık, ÇŞİDB'den alınan örnek iş programında ise imalat kalemleri daha geniş başlıklar altında toplanmıştır. Örneğin, "Kat Betonarme Grubu" veya "Kaplamalar Grubu" gibi kavramlar üzerinden tanımlanan faaliyetler, bazı gruplar kendi içerisinde alt bileşenlere bölünmüş olsa da genel planda daha toplu bir görünüm sunmaktadır. Bu yaklaşım, proje yönetimine daha sade bir bakış sağlamakta; ancak sahada yürütülecek detaylı faaliyetlerin ayrı ayrı tanımlanması konusunda ilave alt planlamalara ihtiyaç duyulabilmektedir.

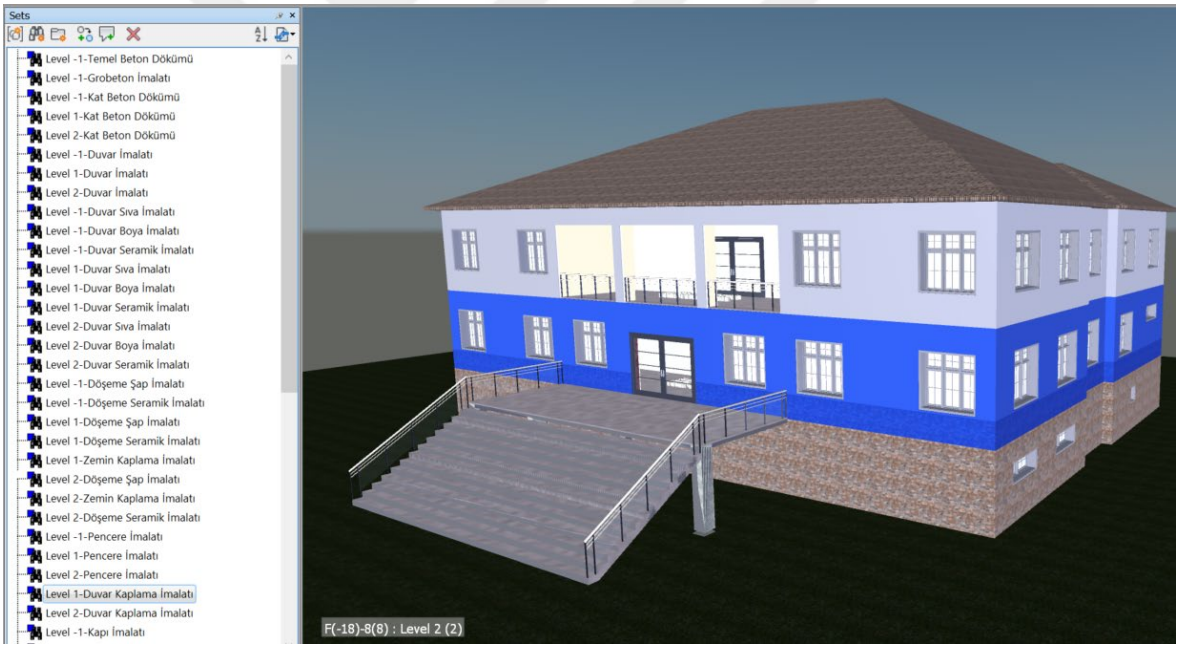
Aktivite sıralaması açısından incelendiğinde, ÇŞİDB'den elde edilen örnek iş programında kat betonarme grubu imalatlarının tamamlanmasının ardından çatı grubu çalışmalarına başlandığı ve bu başlangıcı takiben altı gün sonra duvar grubu imalatlarına geçildiği görülmektedir. Duvar grubu çalışmalarının sona ermesinin ardından duvar kaplama işlerine başlandığı, duvar kaplamalarının başlama tarihinden 24 gün sonra tavan kaplamaları ve 30 gün sonra da döşeme kaplamaları imalatlarına geçildiği tespit edilmiştir. Ayrıca, duvar kaplamalarının başlamasından dört gün sonra cephe grubu imalatlarına başlandığı ve cephe grubu işlerinin başlangıcını takiben 57 gün sonra ise doğrama imalatlarının (pencere, kapı vb.) devreye girdiği de görülmektedir.

Tez kapsamında geliştirilen iş programında da genel olarak sırasıyla betonarme imalatlar, çatı imalatları, duvar imalatları, sıva imalatı, şap imalatı, asma tavan uygulamalarının yer aldığı, bu uygulamaların ardından ise duvar–zemin–tavan kaplamaları (laminant, seramik, boya vb.), pencere imalatı ve kapı imalatları aktivitelerinin geldiği belirlenmiştir. Dolayısıyla, bu aktivite sıralamasının genel hatlarıyla ÇŞİDB'den alınan örnek iş programındaki süreçle büyük ölçüde örtüştüğü gözlemlenmektedir. Bu benzerlik, sahada izlenecek iş adımlarının her iki planda da tutarlı bir şekilde tanımlandığını göstermekte ve planlama sürecinde faaliyetlerin ardışık ya da kısmen paralel olarak yürütülmesine imkân sağlamaktadır.

Tez kapsamında yalnızca FS ilişki türü kullanılmış olması nedeniyle normalde sahada kısmen veya tamamen eş zamanlı yürütülebilecek bazı imalatlar birbirini beklemek zorunda kalmaktadır. Bu durum, programın toplam süresini 700 güne kadar uzatan temel etmenlerden biri olarak öne çıkmaktadır. Öte yandan, örnek iş programında SS ve FF gibi farklı ilişki türleri de kullanıldığı için belirli işler paralel yürütülebilmekte, böylece toplam proje süresi 400 güne kadar kısaltılabilmektedir. Örnek iş programında da aktivitelerin yalnızca FS olarak ilişkilendirildiği kabul edilseydi, inşaat aşamasının 780 gün süreceği görülmektedir. Bu durum tez kapsamında yapılan aktivite süresi hesaplarının da doğruluğunu göstermektedir.

Sonraki aşamada, Revit'te oluşturulan 3B BIM modeli, iş programı ve Search Sets dosyaları Navisworks kullanılarak 4B simülasyona dönüştürülmüştür. Bu amaçla, "BIM Auto Scheduling – Import CSV" isimli bir Navisworks eklentisi geliştirilerek hazırlanan iş programının Navisworks'teki elemanlarla otomatik eşleştirilmesi sağlanmıştır. İlk olarak, Revit'ten aktarılan 3B model Navisworks'te açıldıktan sonra "Örnek Proje SearchSets.xml" dosyası yazılıma dâhil edilmiş ve 54 adet Search Set'in doğru şekilde

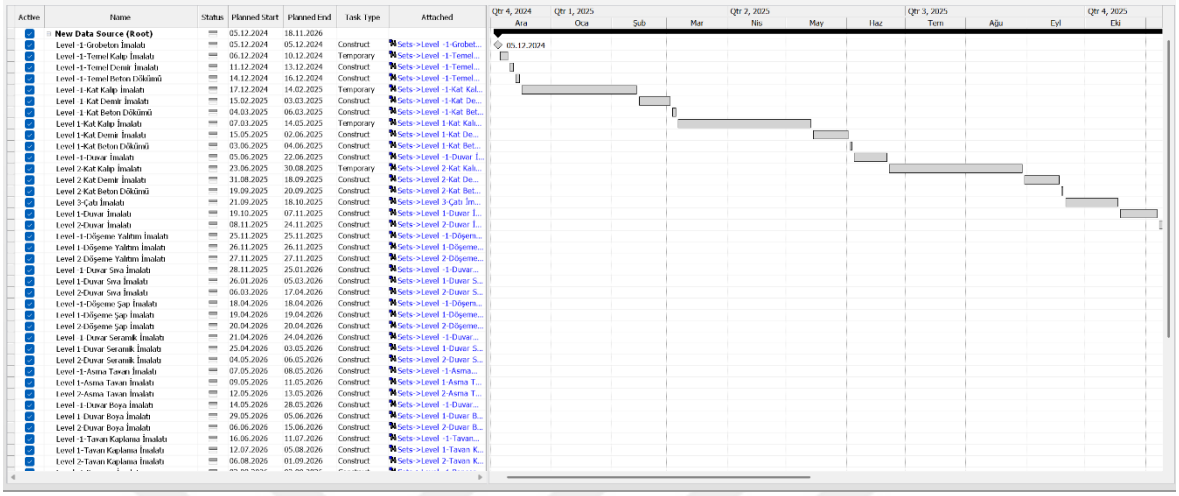
eşleştirildiği gözlemlenmiştir. Şekil 100'de görüldüğü gibi, elemanları "Aktivite Adı", "Aktivite Adı-Kalıp" ve "Aktivite Adı-Demir" gibi parametrelere göre filtreleyebilme olanağı, 4B programlamada elemanların gruplandırılması için büyük kolaylık sunmaktadır. Ayrıca, Şekil 100'de örnek olması için Level 1'de bulunan duvar kaplama elemanına ait, "Aktivite Adı" parametresinin değeri ile aynı isimde oluşturulan Search Sets'in seçimi gösterilmektedir. Search Sets listesinin sol tarafta sıralandığı ve "Level 1-Duvar Kaplama İmalatı" isimli Search Sets'in seçildiği görülmektedir. Sağ tarafta bulunan 3B modelde de zemin katta (model için Level 1'i ifade etmektedir) bulunan dış cephe kaplama elemanları mavi renkle belirginleştirilmektedir. Benzer şekilde tüm elemanlara ait Search Sets'lerin oluşturulduğu ve elemanlarla doğru bir şekilde eşleştiği görülmüştür. Bu sayede, elemanların doğru aktivitelerle eşleştirilmesi sağlanmış ve herhangi bir imalat grubunun planlama adımlarında eksik veya hatalı ilişkilendirilmesinin önüne geçilmiştir.



Şekil 100. Navisworks Search Sets

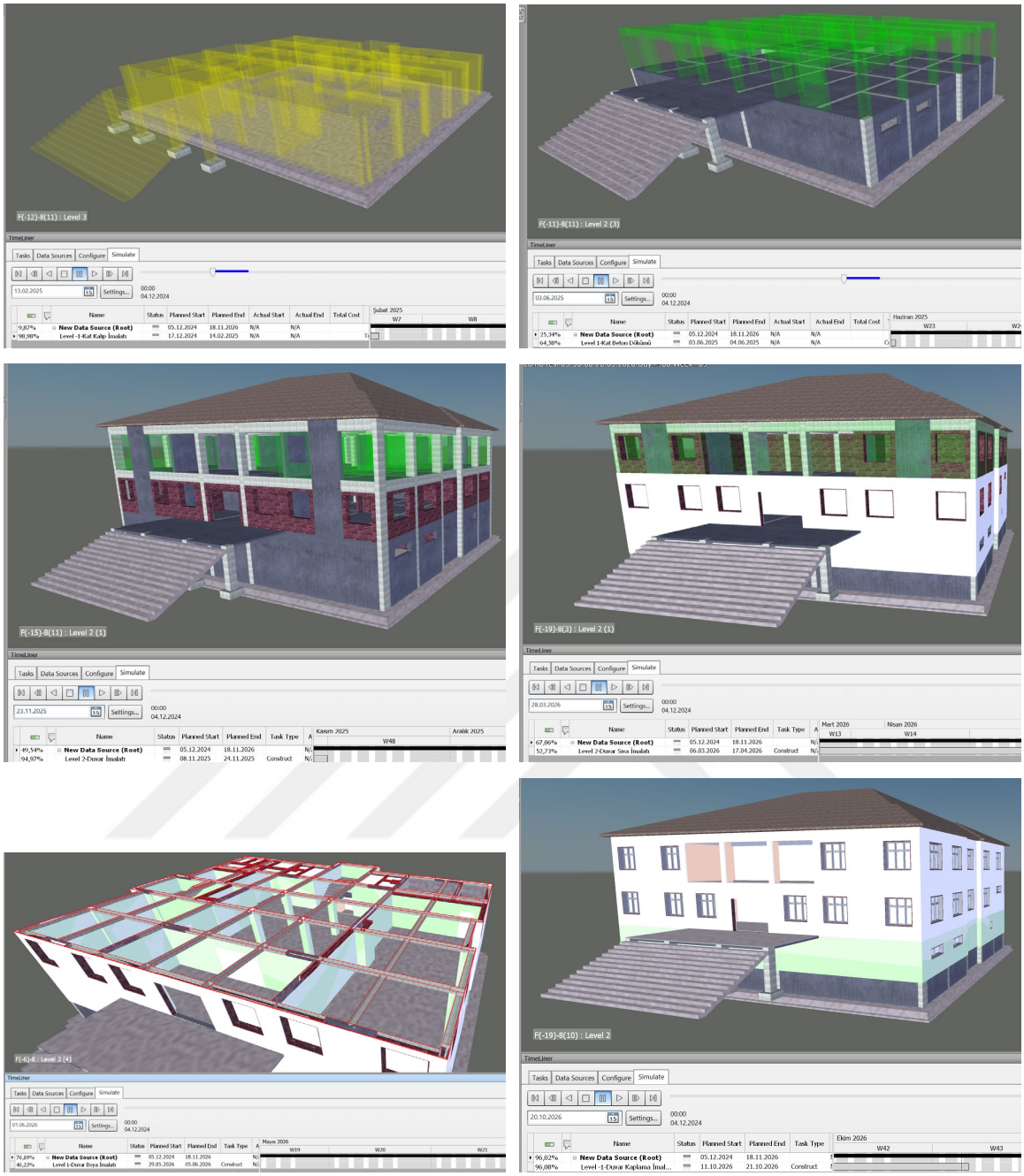
İkinci adımda, Navisworks Timeliner modülü aracılığıyla oluşturulan iş programı modele entegre edilmiştir. Timeliner > Data Sources ekranında tez kapsamında geliştirilen "BIM Auto Scheduling – Import CSV" eklentisi çalıştırılarak, "Örnek Proje.csv" dosyası herhangi bir ek düzenleme yapılmadan içe aktarılmış ve Search Sets'ler üzerinden modeldeki elemanlarla otomatik olarak eşleştirilmiştir. Burada, bütün aktivitelerin Navisworks ortamına başarıyla aktarıldığı ve ilgili yapı elemanlarıyla doğru biçimde

ilişkilendirildiği tespit edilmiştir. Şekil 101'de aktivitelerin Navisworks'e aktarılmış ve elemanlarla ilişkilendirilmiş (attach) olduğu gösterilmektedir.



Şekil 101. Navisworks Timeliner ile iş programının gösterimi

Son aşamada Timeliner > Simulate sekmesi kullanılarak 4B simülasyon videosu oluşturulmuştur. Videodan elde edilen görünümler Şekil 102'de gösterilmektedir. Sarı renk ile Level -1-Kat Kalıp İmalatı aktivitesinin yani geçici işlerin temsil edildiği, daha sonra da sırasıyla Level 1-Kat Beton Dökümü, Level 2-Duvar İmalatı, Level 2-Duvar Sıva İmalatı, Level 1-Duvar Boya İmalatı ve Level 1-Duvar Kaplama İmalatı aktivitelerine ait görseller verilmektedir. 54 aktiviteden oluşan sürecin tamamının 4B simülasyon videosu içinde görselleştirilebildiği anlaşılmıştır.



Şekil 102. 4B simülasyon videosundan görseller

Tüm bu değerlendirmeler, Revit için geliştirilen “BIM Auto Scheduling” eklentisi ile Navisworks ortamı için oluşturulan “BIM Auto Scheduling – Import CSV” eklentisinin hedeflendiği doğrultuda işlediğini ve eklentiler üzerinden elde edilen iş programı verilerinin 4B simülasyon hâline getirilmesinde herhangi bir sorun yaşanmadığını göstermektedir. Dynamo kodlarının öngörüldüğü şekilde parametreleri oluşturup güncellediği ve gerekli hesaplamaları yaptığı, ML’den elde edilen aktivite sıralaması ile elemanları birleştirerek iş programını elde ettiği, C# ile geliştirilen arayüz ve otomasyon

araçlarının sorunsuz çalıştığı ve böylece örnek projeye ait tüm aktivite adımlarının doğru zamanlamayla birleştirildiği anlaşılmaktadır. 3B BIM modeliyle ilişkilendirilen aktivite bilgileri ve ML aktivite sıralaması sayesinde 4B simülasyon oluşturulması, tez kapsamında planlanan yöntem ve amaçlar doğrultusunda beklenen çıktıların başarıyla elde edildiğini doğrulamaktadır.

İnşaat projelerinde BIM tabanlı otomasyon ve aktivitelerin sıralanması üzerine gerçekleştirilen araştırmaların ortak paydası, BIM verilerini esas alarak yapı elemanlarına ait nicel ve nitel bilgileri otomatik veya yarı otomatik biçimde işleyip, aktivite sürelerini hesaplamak ve iş programını oluşturmaktır. Ancak literatürde incelenen çalışmalar arasında belirgin bazı farklılıklar göze çarpmaktadır. Bu farklılıklar; kullanılan yazılımlar, odaklanılan yapı elemanları, ele alınan aktivite ilişkisi türleri ve tercih edilen analiz yöntemleri etrafında yoğunlaşmaktadır. Bu çalışmaların bir kısmı Tablo 19'da özetlenmiştir.

Tablo 19. BIM tabanlı otomatik iş programı üretimine yönelik farklı yaklaşımlar

Yazar(lar) / Yıl	Çalışmanın Amacı / Kullanılan Yöntem	Yapı Elemanları	Aktivite Sıralaması / İlişkileri	Kullanılan Program(lar)	Temel Sonuçlar
Kim vd. (2013)	ifcXML verilerini ayrıştırma, kural tabanlı süre hesabı	Duvar, döşeme, tavan, çatı, kapı, pencere	Kullanıcı bazlı kurallar	ArchiCAD, RSMeans, MS Project	Otomatik program oluşturarak programlama hızını artırdı, taslak niteliğinde çıktı elde edildi.
ElMenshawy & Marzouk (2021)	BIM modelinden metraj listeleri elde edilerek Excel VBA kodu aracılığıyla aktiviteler oluşturulması ve GA ile optimizasyon	Temel, kolon, döşeme, duvar, sıva, tavan kaplama, zemin kaplama, boya	Kullanıcı tanımlı sıralama + FS, SS, FF, SF aktivite arası ilişki	Revit, Excel VBA, NSGA-II, Primavera P6	Süre–maliyet dengesi kurdu, 47 aktiviteli bir konut binasında test edildi.
Fazeli vd. (2024)	Revit API ve optimizasyon algoritmaları ile 4B BIM	Duvar (sıva, kaplama), döşeme, tavan	FS, temel mantık kuralları	Revit, Navisworks, MATLAB (GA, PSO, ACO vb.)	69 aktiviteli bir konut binasında test edildi. GA en başarılı sonuçları verdi. 4B entegrasyon ile otomatik süre tahmini mümkün olduğu belirtildi.
Aljebory & QaisIssam (2019)	Yapay zeka tabanlı uzman sistem (KBES) ile Revit–Primavera entegrasyonu	Kolon, duvar, kapı ve pencere (basit konut)	FS	Revit, C# ile geliştirilen yazılım, Primavera P6	Küçük projede yarı otomasyon, büyük projeler için daha fazla geliştirme ihtiyacı gözlemlendi.
Wang & Rezazadeh Azar (2019)	BIM verilerinden Dynamo, RSMeans veritabını, vaka ve kural temelli akıl yürütme ile taslak iş programı elde edilmesi	Betonarme karkas (temel, kolon, kiriş, döşeme, duvar)	FS +paralel senaryolar	Revit, Dynamo, Excel, MS Project	Dört ve beş katlı binalarda, ardışık ve kısmen örtüşen senaryolarla %86-95 benzerlikte iş programı üretildi. Son kullanıcı revizyonunun önemi vurgulandı.
Wefki vd. (2024)	BIM tabanlı iş programı, GA ile optimizasyon	Betonarme temel, kolon, kiriş, döşeme	FS + ek kaynak senaryolar (GA)	Revit, SQL, GA, Navisworks, Power BI	Süre–maliyet optimizasyonu ve 5B gösterge tablolarıyla entegrasyon sağlandı.
Faghihi vd. (2014)	IFC formatlı BIM dosyası ve GA ile otomatik iş programı oluşturma	Çelik veya betonarme kolon ve kiriş	FS	IFC, GA tabanlı prototip	Farklı yapı boyutlarında ve çeşitli GA parametre ayarlarında %100 yapısal uyumlu sıralama elde edilmiştir.
Altun & Akçamete (2019)	3B model ve iş programı arasındaki bağlantıyı yarı veya tam otomatik hâle getiren iki aşamalı bir yöntem ile 4B modelleme	Çeşitli model öğeleri (Revit aileleri, montaj kodu vb.)	FS (Navisworks'te ekleniyor)	Revit, Dynamo, MS Project, Navisworks	Ofis binası ile test edildi. Çok katmanlı işler incelenmedi ve Navisworks'te işlemler manuel olarak gerçekleştirildi.
Mikulakova vd. (2010)	İş planlarının bilgi tabanlı olarak üretilmesi ve farklı alternatiflerin çok ölçütlü değerlendirmesi	Duvar, döşeme, kolon, kiriş, merdiven	FS, SS (kural tabanlı), çoklu senaryo	IFC, CBR veritabanı, çok ölçütlü karar verme	Eski projelerden benzer vakaları yeniden kullanarak otomatik iş programı üretilmektedir.

Tablo 19'dan da görüldüğü üzere, bazı yazarlar ifcXML veya IFC formatını doğrudan ayrıştırıp RSMeans vb. veri tabanlarıyla üretkenlik oranlarını ilişkilendirerek FS odaklı programlama tercih ederken, diğerleri Revit–Primavera veya Revit–MS Project entegrasyonuna odaklanmış ve SS, FF gibi daha gelişmiş ilişki tiplerini de dâhil eden senaryolar geliştirmiştir. Benzer şekilde, taşıyıcı sistemi (temel, kolon, kiriş, döşeme) önceleyen yaklaşımlar çoğunlukta olsa da duvar, kapı, pencere ya da döşeme kaplaması gibi farklı yapı elemanlarını da içeren çalışmalar mevcuttur.

Bu karşılaştırmalardan görülebileceği gibi, BIM tabanlı otomatik veya yarı otomatik inşaat programı üretimi konusunda farklı yazarlar farklı yazılımlar, yöntemler ve aktivite ilişkileri benimsemiş olsa da tüm çalışmalardaki ortak amaç; model verilerini kullanarak planlama sürecini hızlandırmak, hata payını minimuma indirmek ve proje paydaşlarına daha etkin bir zaman-maliyet entegrasyonu sağlamaktır. Çalışmalarda genelde sınırlı türde yapı elemanlarına odaklanıldığı, aktivite sıralamasının ve ilişkilerin (çoğunlukla FS) ya kullanıcı tarafından belirlendiği ya da belirli basit kurallarla sabitlendiği dikkat çekmektedir. Ayrıca, inşaat sektöründeki mevcut proje verilerinden yararlanma oranının düşük olduğu, AI yöntemlerinin ise daha çok optimizasyon aşamasında kullanıldığı göze çarpmaktadır. Dolayısıyla, geliştirilen prototiplerin pratik uygulamalarda yaygınlaşabilmesi için (i) daha çok yapı elemanını dikkate alması, (ii) kalıp imalatı, donatı imalatı, sıva, seramik, laminant gibi bir elemanın çok disiplinli bileşenlerinin sisteme dâhil edilmesi, (iii) modelleme ve veri aktarım süreçlerinde standardizasyonun artması, (iv) şantiye kısıtları, lojistik, iş güvenliği gibi gerçek koşulların programa daha iyi yansıtılması, (v) büyük ölçekli projelerde kapsamlı performans testlerinin yapılması, (vi) literatürdeki ve sektördeki mevcut inşaat verilerinden daha fazla yararlanılması ve (vii) ML veya DL gibi AI yöntemlerinin iş programı oluşturulması sürecinin farklı aşamalarına entegre edilmesi gibi öneriler yazarlar tarafından vurgulanmaktadır.

Tez kapsamında gerçekleştirilen çalışmaların, literatürde yer alan benzer çalışmalarla hem amaç hem de elde edilen prototipin uygunluğu açısından karşılaştırıldığında, BIM tabanlı otomatik iş programı üretilmesi, aktivitelerin sıralamasının inşaat sektörüne uygunluğu ve 4B modelleme süreçleri bakımından genel bir uyum sağladığı görülmektedir. Bununla birlikte, tez çalışmamızın daha geniş kapsam, artan otomasyon ve AI destekli planlama yaklaşımlarıyla bu çalışmalardan bir adım öne çıktığı görülmektedir. Örneğin, Kim vd. (2013) ve Fazeli vd. (2024) çalışmalarında sınırlı sayıda yapı elemanı incelenirken, çalışmamızda çok daha fazla yapı elemanı ele alınmış, otomatik aktivite

seçim mekanizması geliştirilmiş ve 4B simülasyon süreciyle entegre edilmiştir. ElMenshawy & Marzouk (2021) ile karşılaştırıldığında ise çalışmamızda aynı katta birden fazla kolon veya döşeme türünü içeren karmaşık projeler ve daha büyük ölçekli planlamalar da AI destekli aktivite sıralamasıyla yönetilmekte, böylece belirli bir sıra yerine dinamik ve veriye dayalı bir yaklaşım benimsenmektedir. Wang & Rezazadeh Azar (2019) çalışmasından farklı olarak incelenen yapı elemanı sayısı artırılmış ve kural/durum tabanlı yöntemler yerine AI tabanlı iş programları kullanılarak 4B simülasyon süreci daha ileri bir seviyeye taşınmıştır. Aljebory & QaisIssam (2019) AI tabanlı kurallarla aktivite sıralaması yaparken, tez kapsamında bu yaklaşım daha da otomatikleştirilmiş, veri seti büyütülmüş ve Revit ile diğer yazılımlar arasında veri alışverişini kolaylaştıran eklentiler geliştirilmiştir. Wefki vd. (2024) ve Faghihi vd. (2014) çalışmalarından farklı olarak merdiven ve mimari elemanlar gibi daha fazla öğeye yer verilmiş ve ML desteğiyle manuel müdahaleler azaltılmıştır. Ayrıca, Altun & Akçamete (2019) araştırmasına ek olarak, aynı model öğesine (örneğin, duvar) birden fazla yapım katmanı (örneğin, tuğla, yalıtım, sıva, boya vb.) atanarak 4B modellemede ayrıntı seviyesi artırılmış ve yine ML yaklaşımlarıyla daha doğru aktivite sıralaması elde edilmiştir. Son olarak, Mikulakova vd. (2010) çalışmasında büyük projelerden elde edilen IFC veri tabanlarını kullanmada yaşanan zorluklar öne çıkarken, çalışmamızda Revit ve Navisworks yazılımlarına entegre eklentiler sayesinde veriye erişim ve işleme sürecini kolaylaştırarak planlama otomasyonu ileri bir seviyeye taşınmıştır. Genel olarak bakıldığında, ele alınan yapı elemanı çeşitliliğinin artırılması, mevcut iş programı verilerine dayanarak ML algoritmalarından yararlanılması yoluyla aktivite sıralamasının otomatik üretilmesi ve BIM ile ilişkilendirilmesi, Revit ve Navisworks eklentileri ile iş programı oluşturulması ve 4B modelleme sürecinde otomasyonun artırılması gibi aşamalar, literatürdeki yaklaşımları bir adım ileri taşımakta ve gelecekteki çalışmalara rehberlik etme potansiyeli barındırmaktadır.

4. SONUÇLAR VE ÖNERİLER

İnşaat projelerinin kapsamı genişledikçe, zaman, maliyet ve kalite unsurlarını eşzamanlı olarak etkin bir biçimde yönetmek giderek zorlaşmaktadır. Manuel yöntemlerle oluşturulan iş programlarının, büyük ölçüde uzman müdahalesine ihtiyaç duyması verimlilik kayıplarını artırırken, hata olasılığını da yükseltmekte ve sonuç olarak iş programlarının doğruluğu ve inşaat projelerinin başarısı olumsuz etkilenmektedir. Bu durum, geleneksel iş programı hazırlama yöntemlerinin yetersiz kalmasına neden olmaktadır.

Tez çalışmasında, büyük oranda manuel ve hataya açık yöntemlerle yürütülen inşaat projelerinde iş programı hazırlama süreçlerinin otomasyona uygun ve BIM odaklı bir çerçevede ele alınması amaçlanmış ve ML yöntemleriyle aktivite sıralaması tahmininden 4B simülasyona uzanan bütüncül bir yaklaşım geliştirilmiştir. Toplam 24 farklı projeden derlenen veri seti (1390 aktivite ve 1824 aktivite-ardıl ilişkisi) üzerinde altı farklı ML modeli (ARM, node2vec, DeepWalk, GCN, GAT, GraphSAGE) ayrıntılı biçimde test edilmiş ve parametre optimizasyonu gerçekleştirilmiştir. Daha sonra söz konusu modellerden elde edilen aktivite sıralama bilgileri, Revit ve Navisworks yazılımlarıyla bütünleşik şekilde kullanılabilecek bir iş programı ve 4B model oluşturma süreciyle desteklenmiştir. Geliştirilen "BIM Auto Scheduling" eklentisi, Revit platformuna gömülü bir biçimde Dynamo kodlarını otomatik olarak çalıştırarak iş programı verilerinin oluşturulmasını sağlarken, "BIM Auto Scheduling – Import CSV" eklentisi de Navisworks içerisinde aktivite-eleman eşleşmelerini otomatikleştirerek 4B simülasyonun daha hızlı ve hatasız biçimde oluşturulmasına katkıda bulunmuştur.

Tüm bu adımlar, proje paydaşlarının daha hızlı, hatasız ve veri odaklı şekilde iş programı hazırlamasına imkân tanımakta; geleneksel yöntemlerden kaynaklanan zaman ve emek kayıplarını azaltma potansiyeli taşımaktadır. Ayrıca, çalışmada farklı AI ve ML algoritmalarının aktivite sıralaması tahminlerindeki etkinliği detaylı biçimde incelenerek literatüre kıyaslayıcı ve bütüncül bir bakış sunulmuştur. Elde edilen bulgular, manuel planlama yöntemlerinin zaman alıcı ve hataya açık yapısını gidermeye yönelik özgün bir otomasyon fırsatı sunmuştur.

Çalışmada gerçekleştirilen değerlendirmeler sonucunda elde edilen temel sonuçlar aşağıdaki gibidir:

- GCN, GAT, GraphSAGE algoritmaları için farklı boyutlarda (16, 32, 64, 128, 256) gizli katman ve yerleştirme boyutları denenmiştir. GAT, 256×256 boyutunda %67,19 doğruluk oranı ile en yüksek isabete ulaşmıştır. GraphSAGE ise 64×64 parametre setinde %54,69'luk bir doğruluk değeri elde etmiştir. GCN, 64×64 ve 256×256 boyutlarında %29,69'luk orana kadar yükselse de genel tabloda diğer modellerin gerisinde kalmıştır. Bu veriler ışığında, GCN için 64×64 ve 256×256, GAT için 256×256, GraphSAGE için 64×64 parametre setleri en uygun konfigürasyonlar olarak belirlenmiştir.
- Node2vec üzerinde yürütülen 32 farklı deneme, düşük/orta düzey yerleştirme boyutlarının (64–128) doğruluk performansını artırdığını ve p–q parametrelerinin model kararlılığında önemli rol oynadığını göstermiştir. Yerleştirme boyutunun 128, yürüyüş uzunluğunun 80, yürüyüş sayısının 5, iş parçacığı sayısının 2, p değerinin 0.5 ve q değerinin 2 olarak belirlendiği parametre kombinasyonu, test edilen tüm varyasyonlar arasında %73 doğruluk oranı ile en yüksek oranı sağlamıştır.
- DeepWalk algoritmasında gerçekleştirilen 19 farklı parametre denemesinde, 64–128 yerleştirme boyutlarında ve orta seviyede (20–40) yürüyüş uzunluğunda istikrarlı tahmin sonuçları elde edilmiştir. 64 boyutlu yerleştirme, 20 yürüyüş uzunluğu, 5 yürüyüş sayısı ve 2 iş parçacığı parametreleri, incelenen tüm kombinasyonlar arasında en yüksek doğruluk oranını elde etmiştir.
- ARM algoritmasında, minimum destek ve güven eşiklerinin aşırı yüksek belirlenmesi, kural sayısını önemli ölçüde düşürüp doğruluk oranlarını azaltmaktadır. Destek değerinin 0.001 ve güven değerinin 0.1 olduğu durumda, %78 doğruluk oranına ulaşılarak 1993 kural çıkarılması, proje ilişkilerinin geniş kapsamlı olarak tespit edilebildiğini göstermiştir.
- ML modellerinin performansı; eğitim ve test setindeki proje ölçeği, aktivite sayısı ve veri karmaşıklığına göre değişiklik göstermektedir. 24 proje üzerinden, her aktivite yoğunluk kategorisinden (düşük, orta ve yüksek) ikişer proje seçmek suretiyle altı farklı eğitim-test senaryosu kurgulanarak modellerin performansları test edilmiştir. Bu inceleme sonucunda, ARM yönteminin küçük ölçekli veya az aktivite içeren test projelerinde %78'e varan doğruluk oranı ile daha başarılı olduğu gözlemlenirken; node2vec ve DeepWalk algoritmaları test projelerindeki tahmin edilecek aktivite sayısı arttığında ARM'ı geride bırakarak daha yüksek doğruluk oranlarına ulaşmıştır.

GAT; GCN ve GraphSAGE'e kıyasla daha yüksek doğru tahmin sunmakla birlikte, diğer üç yöntemin bazı durumlarda gerisinde kalmaktadır.

- ML çıktıları, "Aktivite Adı" ve "Tahmin Edilemeyen Aktiviteler" başlıklarıyla .csv formatında dış ortama aktarılmakta ve proje gereksinimlerine göre dinamik biçimde güncellenebilen bir temel doküman oluşturulmaktadır. Bu sayede projede eksik kalmış ilişkilerin ve kullanıcı müdahalesi gereken faaliyetlerin tespiti kolaylaşmaktadır. Dolayısıyla, yalnızca öncül-ardıl tahminlerini rapor etmekle sınırlı kalmayan bu yaklaşım, sahadaki gerçek süreçlere uygun biçimde aktivitelerin yeniden düzenlenmesine olanak tanımaktadır.
- Farklı ölçeklerdeki projelerden derlenen veri setinin kullanılması, modelin genellenebilirliğini artırmış ve büyük hacimli veri gerektirmeyen yöntemlerle de yüksek performanslı ardıl tahminin mümkün olduğunu ortaya koymuştur. İnşaat sektöründeki projeler ölçek, tipoloji ve karmaşıklık açısından oldukça çeşitlilik gösterdiğinden, genellenebilir modeller oluşturmak, zaman ve maliyet tahminlerinde daha geniş kitlelere uygulanabilir sonuçlar elde edilmesi açısından kritik önem taşımaktadır.
- Revit programı üzerinde geliştirilen "BIM Auto Scheduling" eklentisi ile ML modellerinden elde edilen aktivite sıralama bilgileri ile 3B BIM modelinde bulunan elemanlar ilişkilendirilerek iş programı elde edilmiştir. Bu süreçte geliştirilen Dynamo kodları aracılığıyla çeşitli proje parametrelerinin oluşturulması, kullanıcı bazlı değerlerin tanımlanması ve kontrolü, metraj hesaplamaları ve listelerinin elde edilmesi, aktivitelerin oluşturulması ve aktivite süresinin hesaplanması, 4B simülasyon için verilerin oluşturulması ve ML çıktısı ile ilişkilendirme adımları gerçekleştirilmiştir. Esnek yapılandırılabilir arayüzü sayesinde, kullanıcılar standart değerleri ve metraj verilerini güncelleyerek iş programını proje gereksinimlerine uygun şekilde düzenleyebilmektedir. Bu AI destekli sistem, manuel planlama hatalarını minimize ederek verimlilik artışı sağlamakta, aynı zamanda proje yönetiminde standartlaşmayı ve koordinasyonu güçlendirerek sektörde zaman ve maliyet tasarrufu sunmaktadır.
- Elde edilen iş programı, ÇŞB'den alınmış benzer ölçekteki bir projenin iş programıyla planlama yöntemi ve aktivite detay seviyesi açısından kıyaslandığında geliştirilen iş programının daha yüksek bir detay seviyesine sahip olduğu gözlemlenmiştir. İmalatların ayrıntılı olarak tanımlanması, saha uygulamalarının

etkin takibini ve iş yükünün daha net bir biçimde planlanmasını sağlar. Böylece proje yöneticileri, olası gecikmeler veya maliyet artışlarıyla ilgili hızlı ve doğru önlemler alabilmektedir.

- İki iş programı aktivitelerin sıralanması açısından karşılaştırıldığında, betonarme, çatı, duvar, kaplama, pencere ve kapı gibi imalat sıralamalarının büyük ölçüde örtüştüğü görülmektedir. Aktivite sıralamalarındaki paralellik, sektörde yaygın kullanılan yöntemlerle tutarlılığı vurgulayarak, hem saha operasyonlarının kolay koordine edilmesi hem de benzer projelere uygulanabilirliğin arttırılması açısından değerli bir rehber niteliği taşımaktadır. Sahada iş akışının doğru sıralamayla planlanması, verimlilik artışı sağlamanın yanı sıra kaynak kullanımında optimizasyon yaratarak maliyetleri kontrol altında tutmayı desteklemektedir.
- Navisworks için geliştirilen "BIM Auto Scheduling – Import CSV" eklentisi, Revit'te oluşturulan iş programı ve ML modeli çıktılarıyla entegre edilerek 4B simülasyon sürecini otomatikleştirmektedir. Revit eklentisi aracılığıyla üretilen "Search Sets" dosyası, Navisworks içinde her aktivite türüne ait elemanları otomatik filtreleyerek doğru ve hızlı bir eşleştirme sağlamaktadır. İş programı, ek düzenlemeye gerek duymadan Navisworks Timeliner modülüne aktarılmış ve ilgili yapı elemanlarıyla doğrudan ilişkilendirilmiştir.
- Oluşturulan 4B simülasyon, kalıp ve betonarme süreçlerinden iç ve dış kaplamalara kadar tüm imalat adımlarının zamanla ilişkilendirilerek takip edilmesine imkân tanımış, böylece proje sürecinin detaylı bir şekilde analiz edilmesini sağlamıştır. Zaman bazlı görsel modelleme, olası gecikme ve çakışmaların önceden tespit edilmesini kolaylaştırarak sahada daha etkin bir yönetim sunmaktadır. Bu sistem, çok katlı ve karmaşık projelerde aktivite takibini güçlendirerek, proje ekibi ve paydaşlar için karar alma süreçlerini hızlandıran kapsamlı bir belge ve simülasyon çıktısı üretmektedir. AI destekli bu entegrasyon, iş programlarının dinamik bir şekilde modellenmesine ve sektörde zaman-maliyet yönetiminin iyileştirilmesine önemli bir katkı sağlamaktadır.

Tez kapsamında benimsenen yöntemin hem 3B BIM modelinin aktivite bilgileriyle bütünleşmesini hem de ML tabanlı aktivite sıralamasıyla 4B simülasyona geçişi başarılı biçimde desteklediği gözlenmiştir. Bu sonuçlar, inşaat projelerinde hızlı, doğru ve entegre planlama süreçlerinin uygulanabilirliğini artırarak sektörde verimlilik ve kalite hedeflerine ulaşılmasında önemli bir katkı sunmaktadır.

4.1. Katkılar

Bu tez çalışması, inşaat sektöründe iş programı ve 4B simülasyon süreçlerini ML ve BIM entegrasyonu aracılığıyla otomasyon odaklı bir yaklaşımla ele almaktadır. Farklı ölçek ve niteliklere sahip projelerde derlenen veri setiyle test edilen yöntem, sektördeki manuel, zaman alıcı ve hataya açık planlama pratiklerine karşı yenilikçi ve dönüştürücü bir çözüm sunmaktadır. Çalışmanın potansiyel katkıları şu başlıklarda özetlenebilir:

- Sunulan modelleme ve otomasyon yaklaşımı, akademik anlamda mevcut ML–BIM çalışmalarına hem metodolojik (farklı ML yöntemlerinin kıyaslamalı kullanımı) hem de uygulama boyutunda (3B modele entegre iş programı ve 4B simülasyon üretimi) özgün katkılar sunmaktadır.
- Büyük veri gerektiren LSTM benzeri yöntemlere kıyasla, GCN, GAT, GraphSAGE, node2vec, DeepWalk ve ARM gibi algoritmalarla daha düşük veri ortamlarında da etkin ardıl tahmini yapılabileceği gösterilmiştir.
- GCN, GAT, GraphSAGE, node2vec, DeepWalk ve ARM gibi farklı algoritmalar aynı veri seti üzerinde denenmiş ve performansları proje ölçeği ve karmaşıklığına göre irdelenmiştir. Literatürde çoğu çalışmada tek bir model incelenirken, bu tez farklı yöntemlerin artı ve eksi yönlerini kıyaslayarak uygulayıcılara daha esnek bir seçim yapma olanağı sunmuştur.
- Çoğunlukla ML modellerinin sadece öncül-ardıl tahmin doğruluğunu vurguladığı ya da BIM temelli çalışmalarda aktivitelerin sabit kabullerle sıralandığı gözlenmektedir. Bu tezde sunulan yaklaşım ise, ML modellerinden elde edilen aktivite sıralamasını dışarı aktarılabilir bir formatta kullanıma sunarak, kullanıcıya gerek duyduğunda sıralamayı değiştirme olanağı tanıyan etkileşimli bir yapı geliştirmektedir. Böylece, genellikle ML modellerinin performans değerlendirmesiyle sınırlı kalan çalışmaların ötesine geçilerek, gerçek projelerde uyarlanabilir ve kullanıcı dostu bir çizelgeleme çözümü oluşturulmakta; ortaya çıkan aktivitelerin sahadaki uygulamaya uygun biçimde düzenlenebilmesi yoluyla literatüre özgün bir katkı sağlanmaktadır.
- Pek çok çalışmada yalnızca taşıyıcı sistem (kolon, kiriş, döşeme vb.) veya belirli yapısal öğeler ele alınırken, bu tezde mimari aşamalardan (duvar, sıva, kaplama, boya, asma tavan vb.) kalıp ve donatı imalatına kadar çok sayıda eleman ve imalat kalemi dâhil edilmiştir. Böylelikle kullanıcıya daha detaylı ve sahaya uyarlanabilir bir planlama modeli oluşturulabilmektedir.

- Geliştirilen “BIM Auto Scheduling” ve “BIM Auto Scheduling – Import CSV” eklentileri, proje paydaşlarının veriyi farklı yazılımlarda tekrarlı biçimde işlemesini engelleyip tek tuşla iş programı ve 4B simülasyon oluşturma imkânı tanımaktadır. Bu sayede, süreçteki manuel işlemleri azaltıp hata olasılığını en aza indiren özgün bir katkı sunulmaktadır.
- Tezde tasarlanan kullanıcı arayüzleri (adam-saat, ekip sayısı, donatı metrajı vb.) gerçek şantiye koşullarına göre hızla güncellenebilecek biçimde yapılandırılmıştır. Literatürdeki pek çok prototip çalışmanın aksine, saha verilerinin anlık değişimine hızlı tepki verme özelliği, geliştirilen modelin pratik kullanıma uygunluğunu artırmaktadır.
- Tezde sunulan yöntem, sektörde “İnşaat 4.0” konseptiyle hedeflenen dijital dönüşüm adımlarını destekleyen yeni bir otomasyon bakış açısı kazandırarak, proje yönetimi ve planlama süreçlerinde daha ileri düzeyde veri odaklı teknolojilerin benimsenmesi için bir örnek oluşturmaktadır.
- Kurumsal ölçekte yaygınlaştırıldığında, sektördeki dijital dönüşümün hızlanmasına, maliyet ve zaman tasarrufunun artmasına ve şantiye yönetiminde sürdürülebilir bir otomasyon kültürünün gelişmesine katkı sağlayacaktır.

4.2. Araştırma Önerileri

Bu tezde geliştirilen BIM tabanlı iş programı otomasyon modeli ve 4B simülasyon yaklaşımı, farklı ölçek ve nitelikteki inşaat projelerinde hız ve doğruluk anlamında önemli avantajlar sunmaktadır. Ancak bu kapsamın genişletilmesi ve daha fazla senaryonun ele alınması için aşağıdaki araştırma önerileri öne çıkmaktadır:

- Mevcut FS ilişkisinin yanı sıra SS, FF ve SF gibi ilişkileri ve aktiviteler arası bekleme sürelerini içeren daha esnek ve gerçekçi iş programı modelleri oluşturulabilir.
- Altyapı, tünel, yol, köprü gibi farklı proje türlerinden veri setleri toplanarak ML modelleri üzerinde daha kapsamlı testler yapılabilir ve genellenebilirlik düzeyi artırılabilir.
- Veri setleri büyütülerek LSTM ve RNN gibi DL yöntemleri kullanılarak ardıl tahminlerindeki doğruluk seviyesinin artırılması hedeflenebilir. Özellikle büyük

hacimli ve karmaşık projelerde daha yüksek doğruluk oranlarına ulaşılması sağlanabilir.

- Aktivite bazında maliyet ve karbon ayak izi gibi verilerin de BIM modeliyle ilişkilendirilmesi, sürdürülebilirlik ve ekonomik analiz açısından daha kapsamlı bir iş programı oluşturulmasına ve 5B/6B perspektifinin geliştirilmesine imkân tanıyabilecektir.
- BIM modeline maliyet verileri eklenerek, süre-maliyet optimizasyonu gerçekleştirilebilir ve böylece tam entegre bir "planlama ve optimizasyon" modülü geliştirilebilir.
- Gerçek zamanlı şantiye verilerinin (sensör, drone veya IoT tabanlı sistemlerle) entegrasyonu sayesinde, iş programının sahadaki güncel koşullara göre dinamik biçimde revize edilebildiği, kesintisiz bir planlama süreci kurgulanabilir.
- Primavera P6 ve MS Project gibi farklı proje yönetimi yazılımlarıyla birlikte çalışabilecek eklentiler ve arayüzler geliştirilerek sektörde yaygın kullanım kolaylaştırılabilir.
- Siber güvenlik, veri gizliliği ve yetkilendirme gibi konuların entegrasyonuyla, özellikle bulut tabanlı veri paylaşımının yaygınlaştığı projelerde modelin daha güvenli ve kurumsal düzeyde kabul edilebilir bir altyapıya kavuşması mümkün olabilir.

Bu öneriler doğrultusunda, gelecekte daha çeşitli proje tipleri, farklı veri toplama yöntemleri ve gelişmiş aktivite ilişkilerinin de iş akışına entegre edildiği araştırmaların inşaat sektörü planlama süreçlerini dijitalleştirme çabalarına hız kazandıracağı düşünülmektedir.

5. KAYNAKLAR

Abdulqader, D. M, Abdulazeez, A. M., & Zeebaree, D. Q. (2020). Machine learning supervised algorithms of gene selection: A review. *Machine Learning*, *62*(3), 233-244.

Abioye, S. O., Oyedele, L. O., Akanbi, L., Ajayi, A., Delgado, J. M. D., Bilal, M., ... & Ahmed, A. (2021). Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges. *Journal of Building Engineering*, *44*, 103299.

Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2019). N-GCN: Multi-scale graph convolution for semi-supervised node classification. *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*, s. 1-9.

Ahmed, S. (2018). Barriers to implementation of building information modeling (BIM) to the construction industry: a review. *Journal of civil engineering and construction*, *7*(2), 107-113.

Ahmed, S. A. A. (2023). *Ağ Yapılı Veriler için Öznitelik Tabanlı Özellik Öğrenimi*. Çankırı Karatekin Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Çankırı.

Ahmed, S. M., Emam, H. H., & Farrell, P. (2014). Barriers to BIM/4D implementation in Qatar. *Smart, Sustainable and Healthy Cities*, 533.

AIA. (2022). *Understanding Level of Development (LOD) in Building Information Modeling (BIM)*: https://learn.aiacontracts.com/articles/6469008-so-what-is-an-lod-anyway/ 5 Ocak 2025.

Akinci, B., Fischen, M., Levitt, R., & Carlson, R. (2002). Formalization and automation of time-space conflict analysis. *Journal of Computing in Civil Engineering*, *16*(2), 124-134.

Alaca, Y. (2023). *Graf ve Karekod Yöntemleriyle Dönüştürülmüş Log Kayıtları Üzerinde Derin Öğrenme Tabanlı Siber Saldırı Tespiti*. Karabük Üniversitesi, Lisansüstü Eğitim Enstitüsü, Doktora Tezi, Karabük.

Alathamneh, S., Collins, W., & Azhar, S. (2024). BIM-based quantity takeoff: Current state and future opportunities. *Automation in Construction*, *165*, 105549.

Ali, K. N., Alhajlah, H. H., & Kassem, M. A. (2022). Collaboration and risk in building information modelling (BIM): A systematic literature review. *Buildings*, *12*(5), 571.

Alişan, Y. (2024). *Dinamik Ağlarda Hastalık Yayılma Analizi*. Harran Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Şanlıurfa.

Aljebory, K. M., & QaisIssam, M. (2019). Developing AI based scheme for project planning by expert merging Revit and Primavera software. *16th International Multi-Conference on Systems, Signals & Devices (SSD)*, s. 404-412.

Aljuaid, A., & Anwar, M. (2022). Survey of supervised learning for medical image processing. *SN Computer Science*, *3*(4), 292.

Alkhaleel, B. A. (2024). Machine learning applications in the resilience of interdependent critical infrastructure systems - A systematic literature review. *International Journal of Critical Infrastructure Protection*, *44*, 100646.

Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., & Aljaaf, A. J. (2020). A systematic review on supervised and unsupervised machine learning algorithms for data science. *Supervised and unsupervised learning for data science*, 3-21.

Alom, M. Z., & Taha, T. M. (2017). Network intrusion detection for cyber security using unsupervised deep learning approaches. *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, s. 63-69.

Althuwaynee, O. F., Aydda, A., Hwang, I. T., Lee, Y. K., Kim, S. W., Park, H. J., ... & Park, Y. (2021). Uncertainty reduction of unlabeled features in landslide inventory using machine learning t-SNE clustering and data mining apriori association rule algorithms. *Applied Sciences*, *11*(2), 556.

Alshamali, H. (2020). *Yapı Bilgi Modellemesinin 3D ve 4D Fonksiyonlarının Örnek Bir İnşaat Projesi Üzerinde İncelenmesi*. Çukurova Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Adana.

Alshammari, K., Beach, T., & Rezgui, Y. (2021). Cybersecurity for digital twins in the built environment: Current research and future directions. *Journal of Information Technology in Construction*, *26*, 159-173.

Al-Sinan, M. A., Bubshait, A. A., & Aljaroudi, Z. (2024). Generation of Construction Scheduling through Machine Learning and BIM: A Blueprint. *Buildings*, *14*(4), 934.

Al-Sinan, M., Bubshait, A., & Al-Dossary, S. (2023). A Framework for Embracing Construction 4.0. *1st Int'l Conference on Challenges in Engineering, Medical, Economics and Education: Research & Solutions (CEMEERS-23),* s. 21-22. Lisbon, Portugal.

Altınok, D. (2020). *Yapı Bilgi Modelleme Yazılımları ile Kaba Yapı Elemanları Metrajı: Sorunlar ve Çözüm Önerileri*. İstanbul Üniversitesi-Cerrahpaşa, Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Altun, M., & Akcamete, A. (2018). A method for facilitating 4D modeling by automating task information generation and mapping. *Advances in Informatics and Computing in Civil and Construction Engineering: Proceedings of the 35th CIB W78 2018 Conference: IT in Design, Construction, and Management*, s. 479-486, Cham: Springer International Publishing.

Amer, F., & Golparvar-Fard, M. (2019). Formalizing Construction Sequencing Knowledge and Mining Company-Specific Best Practices from Past Project Schedules. *Computing in Civil Engineering 2019: Visualization, Information Modeling, and Simulation - Selected Papers from the ASCE International Conference on Computing in Civil Engineering 2019*, s. 215–223.

Amer, F., & Golparvar-Fard, M. (2021). Modeling dynamic construction work template from existing scheduling records via sequential machine learning. *Advanced Engineering Informatics*, *47*, 101198.

Amer, F., Hockenmaier, J., & Golparvar-Fard, M. (2022). Learning and critiquing pairwise activity relationships for schedule quality control via deep learning-based natural language processing. *Automation in Construction*, *134*, 104036.

Amer, F., Jung, Y., & Golparvar-Fard, M. (2023). Construction schedule augmentation with implicit dependency constraints and automated generation of lookahead plan revisions. *Automation in Construction*, *152*, 104896.

Amer, F., Koh, H. Y., & Golparvar-Fard, M. (2021). Automated methods and systems for construction planning and scheduling: Critical review of three decades of research. *Journal of Construction Engineering and Management*, *147*(7), 03121002.

Anowar, F., Sadaoui, S., & Selim, B. (2021). Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, *40*, 100378.

Arabameri, A., Arora, A., Pal, S. C., Mitra, S., Saha, A., Nalivan, O. A., ... & Moayedi, H. (2021). K-fold and state-of-the-art metaheuristic machine learning approaches for groundwater potential modelling. *Water Resources Management*, *35*, 1837-1869.

Arayici, Y., Coates, P., Koskela, L., Kagioglou, M., Usher, C., & O'reilly, K. J. S. S. (2011). BIM adoption and implementation for architectural practices. *Structural survey*, *29*(1), 7-25.

Arya, D., Gupta, D. K., Rudinac, S., & Worring, M. (2020). Hypersage: Generalizing inductive representation learning on hypergraphs. https://arxiv.org/abs/2010.04558

Asadi, K., Ramshankar, H., Noghabaei, M., & Han, K. (2019). Real-time image localization and registration with BIM using perspective alignment for indoor monitoring of construction. *Journal of Computing in Civil Engineering*, *33*(5), 04019031.

Aubin, P. (2020). Revit families: A step-by-step introduction. *https://www. autodesk. com/autodesk-university/article/Revit-Families-Step-Step-Introduction-2018* adresinden alınmıştır.

Ay, T. (2023). *A Comparative Study on Node Classification Methods for Undirected Social Networks*. Bahçeşehir Üniversitesi, Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Ayhan, B. U., Doğan, N. B., & Tokdemir, O. B. (2020). An association rule mining model for the assessment of the correlations between the attributes of severe accidents. *Journal of civil engineering and management*, *26*(4), 315-330.

Azhar, S., Khalfan, M., & Maqsood, T. (2012). Building information modeling (BIM): Now and beyond. *Australasian Journal of Construction Economics and Building, 12*(4), 15-28.

Aziz, N. D., Nawawi, A. H., & Ariff, N. R. M. (2016). Building information modelling (BIM) in facilities management: Opportunities to be considered by facility managers. *Procedia-Social and Behavioral Sciences*, *234*, 353-362.

Babatunde, S. O., Ekundayo, D., Adekunle, A. O., & Bello, W. (2020). Comparative analysis of drivers to BIM adoption among AEC firms in developing countries: A case of Nigeria. *Journal of Engineering, Design and Technology*, *18*(6), 1425-1447.

Bahadır, Ü. (2018). *Yenileme Projelerinin Yönetim Süreçlerinde Yapı Bilgi Modellemesinin Kullanımı: Vaka Çalışması*. K.T.Ü., Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Trabzon.

Balakina, A., Simankina, T., & Lukinov, V. (2018). 4D modeling in high-rise construction. *E3S Web of Conferences,* Vol. 33, s. 03044, EDP Sciences.

Bandyopadhyay, S., Kara, H., Biswas, A., & Murty, M. N. (2018). Sac2vec: Information network representation with structure and content. https://arxiv.org/abs/1804.10363

Banihashemi, S., & Sohi, S. Z. (2022). *Data-Centric Regenerative Built Environment: Big Data for Sustainable Regeneration*. Routledge.

Banihashemi, S., Tabadkani, A., & Hosseini, M. R. (2018). Integration of parametric design into modular coordination: A construction waste reduction workflow. *Automation in Construction*, *88*, 1-12.

Bankar, R. S., & Lihitkar, S. R. (2019). Science Mapping and Visualization Tools Used for Bibliometric and Scientometric Studies: A Comparative Study. *Journal of Advancements in Library Sciences*, *6*(1), 382-394.

Bansal, V. K. (2017). Integrated CAD and GIS–based framework to support construction planning: case study. *Journal of Architectural Engineering*, *23*(3), 05017005.

Bargstädt, H. J. (2015). Challenges of BIM for construction site operations. *Procedia Engineering*, *117*, 52-59.

Bhavsar, H., & Ganatra, A. (2012). A comparative study of training algorithms for supervised machine learning. *International Journal of Soft Computing and Engineering (IJSCE)*, *2*(4), 2231-2307.

BIMForum (2024, Aralık 20). *LOD Specification*. https://bimforum.global/lod-2/ adresinden alınmıştır.

Biswas, A., Ghosh, A., Kar, A., Mondal, T., Ghosh, B., & Bardhan, P. K. (2021). The impact of COVID-19 in the construction sector and its remedial measures. *Journal of Physics: Conference Series*, *1797*(1), 012054.

Blanco, J. L., Rockhill, D., Sanghvi, A., & Torres, A. (2023). *From start-up to scale-up: accelerating growth in construction technology*. McKinsey & Company.

Bonaccorso, G. (2018). *Machine Learning Algorithms: Popular algorithms for data science and machine learning*. Packt Publishing Ltd.

Borrmann, A., König, M., Koch, C., & Beetz, J. (2018). Building information modeling: Why? what? how?, s. 1-24. Springer International Publishing.

Boton, C., Kubicki, S., & Halin, G. (2015). The challenge of level of development in 4D/BIM simulation across AEC project lifecyle. A case study. *Procedia Engineering*, *123*, 59-67.

Bozorgi, E., Alqaiidi, S. K., Shams, A., Arabnia, H. R., & Kochut, K. (2025). A Survey on Recent Random Walk-based Methods for Embedding Knowledge Graphs. *The Journal of Supercomputing, 81*, 619.

Butkovic, B., Heesom, D., & Oloke, D. (2019). The Need for Multi-LOD 4D Simulations in Construction Projects. *Journal of Information Technology in Construction*, *24*, 256.

Büchmann-Slorup, R., & Andersson, N. (2010). BIM-based scheduling of Construction–A comparative analysis of prevailing and BIM-based scheduling processes. *27 th Int. Conf. of the CIB W78,* s. 113-123.

Candelario-Garrido, A., García-Sanz-Calcedo, J., & Rodríguez, A. M. R. (2017). A quantitative analysis on the feasibility of 4D planning graphic systems versus conventional systems in building projects. *Sustainable Cities and Society*, *35*, 378-384.

Canpolat, F. (2023). *Küçük Ölçekli Yapılar İçin Yapı Bilgi Modelleme (YBM) Tabanlı Yapım Süreci (4D) ve Maliyeti (5D) Tahmini: Bir Vaka Çalışması*. Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Elazığ.

Cao, D., Wang, G., Li, H., Skitmore, M., Huang, T., & Zhang, W. (2015). Practices and effectiveness of building information modelling in construction projects in China. *Automation in construction*, *49*, 113-122.

Changali, S., Mohammad, A., & van Nieuwland, M. (2015). The construction productivity imperative. *McKinsey Quarterly*, s. 1-10.

Charef, R., Alaka, H. & Emmitt, S., 2018. Beyond the Third Dimension of BIM: a Systematic Review of Literature and Assessment of Professional Views. *Journal of Building Engineering*, *19*, 242–257.

Chen, H., Deng, Z., Xu, Y., & Li, Z. (2021). Non-recursive graph convolutional networks. *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*, s. 3205-3209.

Chen, L., Li, Y., Deng, X., Liu, Z., Lv, M., & He, T. (2022). Semantic-aware network embedding via optimized random walk and paragaraph2vec. *Journal of Computational Science*, *63*, 101825.

Chen, L., Xie, Y., Zheng, Z., Zheng, H., & Xie, J. (2020). Friend Recommendation Based on Multi-Social Graph Convolutional Network. *IEEE Access, 8*, 43618–43629.

Cheng, Y., Yu, W. & Li, Q. (2015). GA-based multi-level association rule mining approach for defect analysis in the construction industry. *Automation in Construction*, *51*, 78-91.

Chien, C. F., Dauzère-Pérès, S., Huh, W. T., Jang, Y. J., & Morrison, J. R. (2020). Artificial intelligence in manufacturing and logistics systems: algorithms, applications, and case studies. *International Journal of Production Research*, *58*(9), 2730-2731.

Choudhary, R., & Gianey, H. K. (2017). Comprehensive review on supervised machine learning algorithms. *2017 International Conference on Machine Learning and Data Science (MLDS)*, s. 37-43.

Ciribini, A. L. C., Ventura, S. M., & Paneroni, M. (2016). Implementation of an interoperable process to optimise design and construction phases of a residential building: A BIM Pilot Project. *Automation in construction*, *71*, 62-73.

Collier, E., & Fischer, M. (1995). Four-dimensional modeling in design and construction. *CIFE Tech. Rep.*, Stanford University, Stanford, Calif.

Collins, W., & Redden, L. (2022). Assessing industry estimating software utilization practices to improve construction education. *Proceedings of the 58th Annual Associated Schools of Construction International Conference, EPiC Series in Built Environmet*, s. 552-559.

Crisci, C., Ghattas, B., & Perera, G. (2012). A review of supervised machine learning algorithms and their applications to ecological data. *Ecological Modelling*, *240*, 113-122.

Crowther, J., & Ajayi, S. O. (2021). Impacts of 4D BIM on construction project performance. *International Journal of Construction Management*, *21*(7), 724-737.

Cui, L., Bai, L., Bai, X., Wang, Y., & Hancock, E. R. (2024). Learning aligned vertex convolutional networks for graph classification. *IEEE Transactions on Neural Networks and Learning Systems*, *35*(4), 4423–4437.

Czmoch, I., & Pękala, A. (2014). Traditional design versus BIM based design. *Procedia Engineering, 91*, 210-215.

Çapraz, M. (2024). *Industrial Plant Construction Projects Scheduling for Elimination and Mitigation of Delay Risks*. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Çepni, Y. (2021). *BIM-Based Formwork and Cladding Quantity Take-Off Using Visual Programing*. Orta Doğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Ankara.

Çıracıoğlu Saraç, A. (2021). *İnşaat Sektöründe Planlama ve Kontrol İş Akış Süreçlerinin Yapı Enformasyonu Modellemesi (BIM) Kullanılarak Etkinleştirilmesi: Kavramsal Bir Model Önerisi*. İstanbul Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Doktora Tezi, İstanbul.

Das, A. (2024). *The BIM in Practice in the Real World*. University of Washington, Master Thesis, Washington.

Datta, S. D., Islam, M., Sobuz, M. H. R., Ahmed, S., & Kar, M. (2024). Artificial intelligence and machine learning applications in the project lifecycle of the construction industry: A comprehensive review. *Heliyon*, *10*(5).

Dawood, N., & Sikka, S. (2009). Development of 4D based performance indicators in construction industry. *Engineering, Construction and Architectural Management*, *16*(5), 438-458.

Desgagné-Lebeuf, A., Lehoux, N., Beauregard, R., & Desgagné-Lebeuf, G. (2019). Computer-assisted scheduling tools in the construction industry: A systematic literature review. *IFAC-PapersOnLine, 52*(13), 1843-1848.

DeWitt, S. (2016). *Design of a 4D Deliverable for Construction Schedule Updates*, The University of North Carolina at Charlotte, Doktora Tezi, Charlotte, ABD.

Doan, T., & Kalita, J. (2015). Selecting machine learning algorithms using regression models. *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, s. 1498-1505.

Dong, S., Zhong, J., Hao, P., Zhang, W., Chen, J., Lei, Y., & Schneider, A. (2018). Mining multiple association rules in LTPP database: An analysis of asphalt pavement thermal cracking distress. *Construction and Building Materials*, *191*, 837-852.

Dou, B., Zhu, Z., Merkurjev, E., Ke, L., Chen, L., Jiang, J., ... & Wei, G. W. (2023). Machine learning methods for small data challenges in molecular science. *Chemical Reviews, 123*(13), 8736-8780.

Doukari, O., Seck, B., & Greenwood, D. (2022). The creation of construction schedules in 4D BIM: A comparison of conventional and automated approaches. *Buildings, 12*(8), 1145.

Eastman, C., M., Teicholz, P. & Sacks, R. (2008). *BIM Handbook: A Guide to Building Information Modelling for Owners, Manager, Designers, Engineers, and Contractors*. 1st Edition, John Wiley & Sons, New Jersey.

Eastman, C., Teicholz, P., Sacks, R., Liston, K. (2011). *BIM Handbook (2nd Edition) A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. John Wiley & Sons, New Jersey.

Eckhardt, C. M., Madjarova, S. J., Williams, R. J., Ollivier, M., Karlsson, J., Pareek, A., & Nwachukwu, B. U. (2023). Unsupervised machine learning methods and emerging applications in healthcare. *Knee Surgery, Sports Traumatology, Arthroscopy*, *31*(2), 376-381.

Edirisinghe, R., & London, K. (2015). Comparative analysis of international and national level BIM standardization efforts and BIM adoption. *Proceedings of the 32nd CIB W78 Conference*, s. 149-158.

Elghaish, F., & Abrishami, S. (2020). Developing a framework to revolutionise the 4D BIM process: IPD-based solution. *Construction Innovation, 20*(3), 401-420.

Elmalı, Ö., & Bayram, S. (2022). Adoption of BIM concept in the Turkish AEC industry. *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, 1-18.

ElMenshawy, M., & Marzouk, M. (2021). Automated BIM schedule generation approach for solving time–cost trade-off problems. *Engineering, Construction and Architectural Management, 28*(10), 3346-3367.

El-Sabek, L. M., & McCabe, B. Y. (2018). Coordination challenges of production planning in the construction of international mega-projects in the Middle East. *International Journal of Construction Education and Research*, *14*(2), 118-140.

Enegbuma, W. I., Dodo, Y. A., & Ali, K. N. (2014). Building information modelling penetration factors in Malaysia. *International Journal of Advances in Applied Sciences (IJAAS)*, *3*(1), 47-56.

Faghihi, V., Nejat, A., Reinschmidt, K. F., & Kang, J. H. (2015). Automation in construction scheduling: a review of the literature. *The International Journal of Advanced Manufacturing Technology*, *81*, 1845-1856.

Faghihi, V., Reinschmidt, K. F., & Kang, J. H. (2014). Construction scheduling using genetic algorithm based on building information model. *Expert Systems with Applications*, *41*(16), 7565-7578.

Fan, C. L. (2020). Defect risk assessment using a hybrid machine learning method. *Journal of Construction Engineering and Management*, *146*(9), 04020102.

Fazeli, A., Banihashemi, S., Hajirasouli, A., & Mohandes, S. R. (2024). Automated 4D BIM development: The resource specification and optimization approach. *Engineering, Construction and Architectural Management, 31*(5), 1896-1922.

Fleming, W. S. (2016). *BIM Modelling for Structural Analysis*. Poznan University of Technology, Faculty of Civil and Environmental Engineering, Master Thesis, Finland.

Forestier, G., & Wemmert, C. (2016). Semi-supervised learning using multiple clusterings with limited labeled data. *Information Sciences*, *361*, 48-65.

Froese, T. M. (2010). The impact of emerging information technology on project management for construction. *Automation in construction*, *19*(5), 531-538.

Fu, C., Liu, C., Ishi, C. T., & Ishiguro, H. (2020). Multi-modality emotion recognition model with GAT-based multi-head inter-modality attention. *Sensors*, *20*(17), 4894.

Garcia, J., Villavicencio, G., Altimiras, F., Crawford, B., Soto, R., Minatogawa, V., ... & Yepes, V. (2022). Machine learning techniques applied to construction: A hybrid bibliometric analysis of advances and future directions. *Automation in Construction*, *142*, 104532.

Garcia-Lopez, N. P. (2017). *An Activity and Flow-Based Construction Model for Managing On-Site Work*. Stanford University, PhD Thesis, California.

Gledson, B. J., & Greenwood, D. (2017). The adoption of 4D BIM in the UK construction industry: An innovation diffusion approach. *Engineering, Construction and Architectural Management*, *24*(6), 950-967.

Goswell, Y. M. (2020). *Tavsiye Sistemlerinde Node2vec Gömme ve Sinirsel İşbirlikçi Filtreleme Kullanımı*. Atatürk Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Erzurum.

Grohe, M. (2020). Word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, s. 1-16.

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, s. 855-864.

Gu, X., Deligianni, F., Han, J., Liu, X., Chen, W., Yang, G. Z., & Lo, B. (2023). Beyond supervised learning for pervasive healthcare. *IEEE Reviews in Biomedical Engineering, 17*, 42-62.

Gupta, S., Saluja, K., Goyal, A., Vajpayee, A., & Tiwari, V. (2022). Comparing the performance of machine learning algorithms using estimated accuracy. *Measurement: Sensors*, *24*, 100432.

Gülerses, F. (2018). *Yapı Bilgi Modellemesi (5D) ile Maliyet Yönetiminin Avantaj ve Dezavantajlarının Tespiti*. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Günaydın Ünsal, A. (2022). *Kamu Bina Yapım İşlerinde Süre ve Maliyet Sapmalarının Erken Tespiti için Kazanılmış Değer Analizine Dayalı Bir İzleme Sisteminin Geliştirilmesi*. Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, Trabzon.

Hagberg, E., Hagerman, D., Johansson, R., Hosseini, N., Liu, J., Björnsson, E., ... & Hjelmgren, O. (2022). Semi-supervised learning with natural language processing for right ventricle classification in echocardiography—A scalable approach. *Computers in Biology and Medicine*, *143*, 105282.

Hajirasouli, A., Banihashemi, S., Drogemuller, R., Fazeli, A., & Mohandes, S. R. (2022). Augmented reality in design and construction: thematic analysis and conceptual frameworks. *Construction Innovation*, *22*(3), 412-443.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, *30*.

Hamledari, H., McCabe, B., Davari, S., & Shahi, A. (2017). Automated schedule and progress updating of IFC-based 4D BIMs. *Journal of Computing in Civil Engineering*, *31*(4), 04017012.

Han, K. K., & Golparvar-Fard, M. (2015). Appearance-based material classification for monitoring of operation-level construction progress using 4D BIM and site photologs. *Automation in construction*, *53*, 44-57.

Harris, J. (2010). *Integration of BIM and Business Strategy*. Northwestern University, McCormick School of Engineering and Applied Science, Master Thesis, Evanston.

Hartmann, T., Gao, J., & Fischer, M. (2008). Areas of application for 3D and 4D models on construction projects. *Journal of Construction Engineering and management*, *134*(10), 776-785.

Hartmann, V., Beucke, K. E., Shapir, K., & König, M. (2012). Model-based scheduling for construction planning. *14th International Conference on Computing in Civil and Building Engineering*, s. 27-29.

He, Q., Wang, G., Luo, L., Shi, Q., Xie, J., & Meng, X. (2017). Mapping the managerial areas of Building Information Modeling (BIM) using scientometric analysis. *International journal of project management*, *35*(4), 670-685.

Hoang, D., & Wiegratz, K. (2023). Machine learning methods in finance: Recent applications and prospects. *European Financial Management*, *29*(5), 1657-1701.

Hochscheid, E., & Halin, G. (2020). Generic and SME-specific factors that influence the BIM adoption process: An overview that highlights gaps in the literature. *Frontiers of Engineering Management*, *7*(1), 119-130.

Hong, Y., Xie, H., Agapaki, E., & Brilakis, I. (2023). Graph-Based Automated Construction Scheduling without the use of BIM. *Journal of Construction Engineering and Management, 149*(2), 05022020.

Hosseini, M. R., Chileshe, N., Zuo, J., & Baroudi, B. (2015). Adopting global virtual engineering teams in AEC Projects: A qualitative meta-analysis of innovation diffusion studies. *Construction Innovation*, *15*(2), 151-179.

Huang, J., & Bhat, R. (2017). Navisworks Hacks for Efficient Workflows. *Autodesk University*.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., & Levine, S. (2021). How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, *40*(4-5), 698-721.

Jafari, F., Moradi, K., & Shafiee, Q. (2024). Shallow learning vs. deep learning in engineering applications. *Shallow Learning vs. Deep Learning: A Practical Guide for Machine Learning Solutions,* Cham: Springer Nature Switzerland.

Jiang, T., Gradus, J. L., & Rosellini, A. J. (2020). Supervised machine learning: a brief primer. *Behavior therapy*, *51*(5), 675-687.

Jin, Z., Gambatese, J., Liu, D., & Dharmapalan, V. (2019). Using 4D BIM to assess construction risks during the design phase. *Engineering, Construction and Architectural Management, 26*(11), 2637-2654.

Jung, Y., Hockenmaier, J., & Golparvar-Fard, M. (2024). Transformer language model for mapping construction schedule activities to uniformat categories. *Automation in Construction, 157*, 105183.

Kaczmarek, I. (2023). Spatial objects classification using machine learning and spatial walk algorithm. *Open Geosciences*, *15*(1), 20220542.

Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. *Artificial intelligence review*, *52*(1), 273-292.

Kamsu-Foguem, B., Rigal, F., & Mauget, F. (2013). Mining association rules for the quality improvement of the production process. *Expert systems with applications*, *40*(4), 1034-1045.

Kapelko, M., & Abbott, M. (2017). Productivity growth and business cycles: Case study of the Spanish construction industry. *Journal of Construction Engineering and Management*, *143*(5), 05016026.

Karakurumer, B. (2024). *Proje Yönetiminde Zaman ve Maliyet Etkilerinin Yapı Bilgi Modellemesi (BIM) ile İncelenmesi*. Osmaniye Korkut Ata Üniversitesi, Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi, Osmaniye.

Karakuş, C. 2023. Makine Öğrenmesi Ders Notları.

Kassem, M., Dawood, N., & Chavada, R. (2015). Construction workspace management within an Industry Foundation Class-Compliant 4D tool. *Automation in construction*, *52*, 42-58.

Kataoka, M. (2008). Automated generation of construction plans from primitive geometries. *Journal of Construction Engineering and Management*, *134*(8), 592-600.

Kaşıkçı, K. (2024). *Feature Generation for SME Behavioral Credit Scoring Model Using Graph Embeddings*. Koç Üniversitesi, Fen Bilimleri ve Mühendislik Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Kaya, U., & Özener, O. Ö. (2024). A strategic evaluation of BIM-driven information management in the context of ISO 19650-2 standard. *Engineering, Construction and Architectural Management*.

Kayhan, B. M., & Yildiz, G. (2023). Reinforcement learning applications to machine scheduling problems: a comprehensive literature review. *Journal of Intelligent Manufacturing*, *34*(3), 905-929.

Khemani, B., Patil, S., Kotecha, K., & Tanwar, S. (2024). A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data*, *11*(1), 18.

Kıvırcık, İ. (2016). *An Investigation into the Building Information Modeling Applications in the Construction Project Management*. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, İstanbul.

Kim, H., Anderson, K., Lee, S., & Hildreth, J. (2013). Generating construction schedules through automatic data extraction using open BIM (building information modeling) technology. *Automation in Construction, 35*, 285-295.

Kim, K., & Cho, Y. K. (2015). Construction-specific spatial information reasoning in building information models. *Advanced Engineering Informatics*, *29*(4), 1013-1027.

Kim, M., Baek, S. H., & Song, M. (2018). Relation extraction for biological pathway construction using node2vec. *BMC bioinformatics, 19*, 75-84.

King, P. L. (2019). *Lean for the process industries: dealing with complexity*. Productivity Press, New York.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. https://arxiv.org/abs/1609.02907

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 4909-4926.

Koo, B., & Fischer, M. (2000). Feasibility study of 4D CAD in commercial construction. *Journal of construction engineering and management*, *126*(4), 251-260.

Kropp, C., Koch, C., & König, M. (2018). Interior construction state recognition with 4D BIM registered image sequences. *Automation in construction*, *86*, 11-32.

Kumaş, O. (2023). *Fraud Detection in Blockchain Cryptocurrencies*. Bahçeşehir Üniversitesi, Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Kuruoğlu, M. (2007). *İnşaat Proje Yönetimi Temel İlkeleri – 1*, İnşaat Mühendisleri Odası, İstanbul.

Lampert, M., & Scholtes, I. (2023). The self-loop paradox: Investigating the impact of self-loops on graph neural networks. https://arxiv.org/abs/2312.01721

Lee, S., Cha, Y., Han, S., & Hyun, C. (2019). Application of association rule mining and social network analysis for understanding causality of construction defects. *Sustainability*, *11*(3), 618.

Lee, S., Yu, J., & Jeong, D. (2015). BIM acceptance model in construction organizations. *Journal of management in engineering*, *31*(3), 04014048.

Lei, X., Mohamad, U. H., Sarlan, A., Shutaywi, M., Daradkeh, Y. I., & Mohammed, H. O. (2022). Development of an intelligent information system for financial analysis depend on supervised machine learning algorithms. *Information Processing & Management*, *59*(5), 103036.

Liao, P. C., Chen, H., & Luo, X. (2019). Fusion model for hazard association network development: A case in elevator installation and maintenance. *KSCE Journal of Civil Engineering*, *23*, 1451-1465.

Lin, C. L., & Fan, C. L. (2018). Examining association between construction inspection grades and critical defects using data mining and fuzzy logic. *Journal of Civil Engineering and Management*, *24*(4), 301-317.

Lin, C. Y., Su, H. T., Tung, S. L., & Hsu, W. H. (2021). Multivariate and propagation graph attention network for spatial-temporal prediction with outdoor cellular traffic. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management,* s. 3248-3252.

Lindblad, H., & Vass, S. (2015). BIM implementation and organisational change: A case study of a large Swedish public client. *Procedia Economics and Finance, 21*, 178-184.

Liu, J., Shi, D., Li, G., Xie, Y., Li, K., Liu, B., & Ru, Z. (2020). Data-driven and association rule mining-based fault diagnosis and action mechanism analysis for building chillers. *Energy and Buildings*, *216*, 109957.

Liu, Q. H. & Cao, Q. (2024). Dynamic control method of construction cost based on fuzzy neural network. *Automatika: Journal for Control, Measurement, Electronics, Computing and Communications*, *65*(4), 1339-1349.

Lu, Q., Parlikad, A. K., Woodall, P., Don Ranasinghe, G., Xie, X., Liang, Z., ... & Schooling, J. (2020). Developing a digital twin at building and city levels: Case study of West Cambridge campus. *Journal of Management in Engineering, 36*(3), 05020004.

Ma, N., Mazumder, S., Wang, H., & Liu, B. (2020). Entity-aware dependency-based deep graph attention network for comparative preference classification. *58th annual meeting of the association for computational linguistics*, s. 5782-5788.

Maçka Kalfa, S. (2018). Building information modeling (BIM) systems and their applications in Turkey. *Journal of Construction Engineering, Management & Innovation*, *1*(1), 55-66.

Mahadevkar, S. V., Khemani, B., Patil, S., Kotecha, K., Vora, D. R., Abraham, A., & Gabralla, L. A. (2022). A review on machine learning styles in computer vision—techniques and future directions. *Ieee Access*, *10*, 107293-107329.

Mahalingam, A., Kashyap, R., & Mahajan, C. (2010). An evaluation of the applicability of 4D CAD on construction projects. *Automation in construction*, *19*(2), 148-159.

Martínez-Aires, M. D., López-Alonso, M., & Martínez-Rojas, M. (2018). Building information modeling and safety management: A systematic review. *Safety science*, *101*, 11-18.

Martín-Martín, A., Orduna-Malea, E., Thelwall, M., & López-Cózar, E. D. (2018). Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories. *Journal of informetrics*, *12*(4), 1160-1177.

Mazuera-Rozo, A., Mojica-Hanke, A., Linares-Vásquez, M., & Bavota, G. (2021). Shallow or deep? an empirical study on detecting vulnerabilities using deep learning. *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, s. 276-287).

McGraw Hill Construction. (2014). *The Business Value of BIM for Construction in Global Markets*. McGraw Hill Construction, Bedford MA.

Meng, X. (2012). The effect of relationship management on project performance in construction. *International journal of project management*, *30*(2), 188-198.

Miao, Y., Xu, Y., & Mandic, D. (2022). Hyper-gst: Predict metro passenger flow incorporating graphsage, hypergraph, social-meaningful edge weights and temporal exploitation. *https://arxiv.org/abs/2211.04988.*

Miettinen, R., & Paavola, S. (2014). Beyond the BIM utopia: Approaches to the development and implementation of building information modeling. *Automation in construction*, *43*, 84-91.

Mikulakova, E., König, M., Tauscher, E., & Beucke, K. (2010). Knowledge-based schedule generation and evaluation. *Advanced Engineering Informatics*, *24*(4), 389-403.

Mittal, S., Sharma, I., & Kumar, A. (2023). Comparative Analysis of Shallow Learning and Deep Learning. *2023 International Conference on Next Generation Electronics (NEleX)*, s. 1-8.

Mostofi, F. (2024). *Mekânsal-Zamansal Makine Öğrenimi Modeli ile Risk Bilgili İnşaat İlerleme Tahmini*. Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, Trabzon.

Mostofi, F., & Toğan, V. (2023). Construction safety predictions with multi-head attention graph and sparse accident networks. *Automation in Construction, 156*, 105102.

Mostofi, F., & Toğan, V. (2024). Predicting construction accident outcomes using graph convolutional and dual-edge safety networks. *Arabian Journal for Science and Engineering*, *49*(10), 13315-13332.

Mostofi, F., Toğan, V., Başağa, H. B., Çıtıpıtıoğlu, A., & Tokdemir, O. B. (2023). Multiedge graph convolutional network for house price prediction. *Journal of Construction Engineering and Management*, *149*(11), 04023112.

Mostofi, F., Toğan, V., Ayözen, Y. E., & Tokdemir, O. B. (2022). Construction safety risk model with construction accident network: A graph convolutional network approach. *Sustainability (Switzerland), 14*(23).

Mostofi, F., Tokdemir, O. B., Bahadır, Ü., & Toğan, V. (2025). Performance-driven contractor recommendation system using a weighted activity–contractor network. *Computer-Aided Civil and Infrastructure Engineering, 40*(3), 409-424.

Mostofi, F., Tokdemir, O. B., & Toğan, V. (2023a). Comprehensive root cause analysis of construction defects using semisupervised graph representation learning. *Journal of Construction Engineering and Management*, *149*(9), 04023079.

Mostofi, F., Tokdemir, O. B., Toğan, V., & Arditi, D. (2024). Predicting the Cost of Rework in High-Rise Buildings Using Graph Convolutional Networks. *Journal of Construction Engineering and Management*, *150*(8).

Mou, N., Wang, H., Zhang, H., & Fu, X. (2020). Association rule mining method based on the similarity metric of tuple-relation in indoor environment. *IEEE Access*, *8*, 52041-52051.

Mrosla, L., Koch, V., & Both, P. V. (2019). Quo vadis AI in Architecture? Survey of the current possibilities of AI in the architectural practice. *Des. Artif. Intell*, *2*, 45-54.

Nagarhalli, T. P., Vaze, V., & Rana, N. K. (2021). Impact of machine learning in natural language processing: A review. *2021 third international conference on intelligent communication technologies and virtual mobile networks (ICICV),* s. 1529-1534).

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons*, *4*(51-62), 56.

Naticchia, B., Carbonari, A., Vaccarini, M., & Giorgi, R. (2019). Holonic execution system for real-time construction management. *Automation in construction*, *104*, 179-196.

NBIMS. (2012). *National BIM Standard-United States Version 2*. National Institute of Building Sciences.

Olde Scholtenhuis, L. L., Hartmann, T. and Dorée, A. G. (2016). 4D CAD based method for supporting coordination of urban subsurface utility projects. *Automation in construction*, *62*, 66-77.

Olsen, D., & Taylor, J. M. (2017). Quantity take-off using building information modeling (BIM), and its limiting factors. *Procedia engineering*, *196*, 1098-1105.

Ozcan Deniz, G. (2018). Emerging cad and bim trends in the aec education: An analysis from students' perspective. *Journal of Information Technology in Construction*, *23*, 138-156.

Özcan, S. (2021). *Yapı Bilgi Modeli ve İş Programı Entegrasyonunun (4.Boyut) İş Güvenliğine Katkılarının Değerlendirilmesi*. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Özorhon, B. (2018). *BIM (Yapı Bilgi Modellemesi) İBB Anadolu Yakası Raylı Sistem Projeleri*. Abaküs Kitap, İstanbul.

Palumbo, E., Rizzo, G., & Troncy, R. (2017). Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. T*he eleventh ACM conference on recommender systems*, s. 32-36.

Pan, X., Zhong, B., Wang, Y., & Shen, L. (2022). Identification of accident-injury type and bodypart factors from construction accident reports: A graph-based deep learning framework. *Advanced Engineering Informatics*, *54*.

Pan, Y., Zhang, L., & Skibniewski, M. J. (2020). Clustering of designers based on building information modeling event logs. *Computer-Aided Civil and Infrastructure Engineering, 35*(7), 701-718.

Pan, Y., & Zhang, L. (2021). Roles of artificial intelligence in construction engineering and management: A critical review and future trends. *Automation in Construction*, *122*, 103517.

Paneru, S., Ghimire, P., Kandel, A., Thapa, S., Koirala, N., & Karki, M. (2023). An exploratory investigation of implementation of building information modeling in Nepalese architecture–engineering–construction industry. *Buildings, 13*(2), 552.

Park, J., & Cai, H. (2015). Automatic construction schedule generation method through BIM model creation. *Computing in Civil Engineering Conference*, s. 620-627.

Pasupa, K., & Sunhem, W. (2016). A comparison between shallow and deep architecture classifiers on small dataset. *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, s. 1-6.

Patel, A. (2025, Ocak 5). *BIM Level of Development | LOD 100, 200, 300, 350, 400, 500*. https://www.united-bim.com/bim-level-of-development-lod-100-200-300-350-400-500/ adresinden alınmıştır.

Patel, D. (2023). Revolutionizing Project Management with Generative AI. *Int. Sci. J. Eng. Manag*, *2*, 1-8.

Patil, D. A., Kelkar, A., & Malawade, R. (2017). Application of Building Information Modelling Software's for Planning and Scheduling of Multi-Storeyed Building. *International Research Journal of Engineering and Technology, 4*, 2775-2782.

Pearl, J. (2009). *Causality*. Cambridge University Press, New York.

Pekşen, M. F. (2024). *Elektrik Pano Yangınlarına Karşı Nesnelerin İnterneti Destekli Algılama Sisteminin Geliştirilmesi*. Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, Sakarya.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. *20th ACM SIGKDD international conference on Knowledge discovery and data mining*, s. 701-710.

Pittard, S., & Sell, P. (2017). *BIM and quantity surveying*. Routledge.

Politi, R. R. (2018). *Project Planning and Management Using Building Information Modeling (BIM)*. İzmir Yüksek Teknoloji Enstitüsü, İnşaat Mühendisliği, Yüksek Lisans Tezi, İzmir.

Project Management Institute. (2021). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. Seventh Edition and The Standard for Project Management (7th ed.).

Quirk, V. (2012). *A Brief History of BIM, ArchDaily*. https://www.archdaily.com/302490/a-brief-history-of-bim adresinden alınmıştır.

Rahman, M. K., & Azad, A. (2021). A comprehensive analytical survey on unsupervised and semi-supervised graph representation learning methods. https://arxiv.org/abs/2112.10372

Raiman, A. (2022). Comparison of Autodesk Revit & Graphisoft ArchiCad in OpenBIM workflow.

Rajabi, M. S., Rezaeiashtiani, M., Radzi, A. R., Famili, A., Rezaeiashtiani, A., & Rahman, R. A. (2022). Underlying factors and strategies for organizational BIM capabilities: The case of Iran. *Applied System Innovation*, *5*(6), 109.

Ryu, H. G., & Lee, H. S. (2004). Schedule management process model based on CSDM. In *ISARC 2004, 21st Int. Symp. on Automation and Robotics in Construction* (pp. 188-193).

Sacks, R., Girolami, M., & Brilakis, I. (2020). Building information modelling, artificial intelligence and construction tech. *Developments in the Built Environment*, *4*, 100011.

Sacks, R., Treckmann, M., & Rozenfeld, O. (2009). Visualization of work flow to support lean construction. *Journal of construction engineering and management*, *135*(12), 1307-1315.

Saini, H., Singh, K., & Malik, U. (2017). Project management in construction using primavera. *International Journal of Civil Engineering and Technology*, *8*(8), 538-549.

Salama, T., & Moselhi, O. (2019). Multi-objective optimization for repetitive scheduling under uncertainty. *Engineering, Construction and Architectural Management, 26*(7), 1294-1320.

Salamat, A., Luo, X., & Jafari, A. (2020). Balnode2vec: Balanced random walk based versatile feature learning for networks. *2020 International Joint Conference on Neural Networks (IJCNN)*, s. 1-8.

Sampaio, A. Z., Sequeira, P., Gomes, A. M., & Sanchez-Lite, A. (2023). BIM methodology in structural design: A practical case of collaboration, coordination, and integration. *Buildings*, *13*(1), 31.

Sánchez-Hernández, G., Chiclana, F., Agell, N., & Aguado, J. C. (2013). Ranking and selection of unsupervised learning marketing segmentation. *Knowledge-based systems*, *44*, 20-33.

Sarmah, B., Nair, N., Jain, R., Mehta, D., & Pasquali, S. (2024). Learning embedded representation of the stock correlation matrix using graph machine learning. *2024 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*, s. 1-9.

Schmarje, L., Santarossa, M., Schröder, S. M., & Koch, R. (2021). A survey on semi-, self- and unsupervised learning for image classification. *IEEE Access*, *9*, 82146-82168.

Shan, Y., Zheng, B., Chen, L., Chen, L., & Chen, D. (2020). A reinforcement learning-based adaptive path tracking approach for autonomous driving. *IEEE Transactions on Vehicular Technology*, *69*(10), 10581-10595.

Sharpe, C., Wiest, T., Wang, P., & Seepersad, C. C. (2019). A comparative evaluation of supervised machine learning classification techniques for engineering design applications. *Journal of Mechanical Design*, *141*(12), 121404.

Shaygan, M., Meese, C., Li, W., Zhao, X. G., & Nejad, M. (2022). Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities. *Transportation research part C: emerging technologies, 145*, 103921.

Sheikhkhoshkar, M., Rahimian, F. P., Kaveh, M. H., Hosseini, M. R., & Edwards, D. J. (2019). Automated planning of concrete joint layouts with 4D-BIM. *Automation in construction, 107*, 102943.

Shourangiz, E., Mohamad, M. I., Hassanabadi, M. S., Banihashemi, S. S., Bakhtiari, M., & Torabi, M. (2011). Flexibility of BIM towards design change. *2nd International Conference on Construction and Project Management, IPEDR*.

Sigalov, K., & König, M. (2017). Recognition of process patterns for BIM-based construction schedules. *Advanced Engineering Informatics*, *33*, 456-472.

Singh, A. K., Pal, A., Kumar, P., Lin, J. J., & Hsieh, S. H. (2023). Prospects of integrating BIM and NLP for automatic construction schedule management. *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, s. 238-245.

Singh, V., Gu, N., & Wang, X. (2011). A theoretical framework of a BIM-based multi-disciplinary collaboration platform. *Automation in construction, 20*(2), 134-144.

Smith, P. (2014). BIM implementation–global strategies. *Procedia engineering*, *85*, 482-492.

Sompolgrunk, A., Banihashemi, S., Hosseini, M. R., Golzad, H., & Hajirasouli, A. (2023). An integrated model of BIM return on investment for Australian small-and medium-sized enterprises (SMEs). *Engineering, Construction and Architectural Management*, *30*(5), 2048-2074.

Son, H., Kim, C., & Kwon Cho, Y. (2017). Automated schedule updates using as-built data and a 4D building information model. *Journal of Management in Engineering*, *33*(4), 04017012.

Song, Z., Baur, B., & Roy, S. (2023). Benchmarking graph representation learning algorithms for detecting modules in molecular networks. *F1000Research*, *12*, 941.

Soni, J., Prabakar, N., & Upadhyay, H. (2019). Feature extraction through deepwalk on weighted graph. *15th international conference on data science (ICDATA'19),* Las Vegas, NV.

Stanley, R., Thurnell, D. (2014). The benefits of, and barriers to, implementation of 5D BIM for quantity surveying in New Zealand. *Australasian Journal of Construction Economics and Building, 14*(1), 105-117.

Sun, C., Li, C., Lin, X., Zheng, T., Meng, F., Rui, X., & Wang, Z. (2023). Attention-based graph neural networks: a survey. *Artificial Intelligence Review*, *56*(2), 2263-2310.

Şahin, E. (2023). *Boru Hatlarında Arıza Durumlarının Grafik Evrişimli Sinir Ağları (GCN) ile Tespiti*. Marmara Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, İstanbul.

Şenol, B. (2023). *Yapı Bilgi Modellemesinin Proje Yönetim Süreçlerine Etkisi*. Bahçeşehir Üniversitesi, Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi, İstanbul.

Tahmasebi, P., Kamrava, S., Bai, T., & Sahimi, M. (2020). Machine learning in geo-and environmental sciences: From small to large scale. *Advances in Water Resources*, *142*, 103619.

Tauscher, E., Mikulakova, E., Beucke, K., & König, M. (2009). Automated generation of construction schedules based on the IFC object model. In *Computing in Civil Engineering (2009)* (pp. 666-675).

Temel, B. A. (2022). *BIM Tabanlı Çizilen Projelerin Mevzuata Uygunluk Denetimini Gerçekleştiren Sistem Geliştirilmesi*. Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, Trabzon.

Topal, N. (2019). *İnşaat Projelerinde Yapı Bilgi Modellemesi Uygulamaları: 5D Modelleme İle Örnek Vaka Çalışması*. Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Eskişehir.

Torres-Calderon, W., Chi, Y., Amer, F. and Golparvar-Fard, M. (2019). Automated mining of construction schedules for easy and quick assembly of 4D BIM simulations. *ASCE International Conference on Computing in Civil Engineering 2019: Visualization, Information Modeling, and Simulation, American Society of Civil Engineers (ASCE)*, s. 432-438.

Trebbe, M., Hartmann, T., & Dorée, A. (2015). 4D CAD models to support the coordination of construction activities between contractors. *Automation in construction*, *49*, 83-91.

Tripathi, A. M., & Mishra, A. (2021). Self-supervised learning for environmental sound classification. *Applied Acoustics*, *182*, 108183.

Tulenheimo, R. (2015). Challenges of implementing new technologies in the world of BIM–Case study from construction engineering industry in Finland. *Procedia Economics and Finance*, *21*, 469-477.

URL-1. (2024, Aralık 18). About Parameters. https://help.autodesk.com/view/RVT/2025/ENU/?guid=GUID-AEBA08ED-BDF1-4E59-825A-BF9E4A871CF5 adresinden alınmıştır.

URL-2. (2024, Aralık 19). Differences Between Navisworks Freedom, Simulate and Manage. https://globaletraining.na1.teamsupport.com/knowledgeBase/33733362 adresinden alınmıştır.

URL-3. (2024, Aralık 19). Structural Dynam(o)ite: Optimized Design and Fabrication Workflows with Dynamo. https://www.autodesk.com/autodesk-university/article/Structural-Dynamoite-Optimized-Design-and-Fabrication-Workflows-Dynamo-2019 adresinden alınmıştır.

URL-4. (2024, Eylül 18). Yapı Yaklaşık Maliyeti Hesabı: http://www.ikolyapidenetim.com.tr/pratik-bilgiler-yapi-yaklasik-maliyeti-hesabi.html adresinden alınmıştır.

Van Eck, N. J., & Waltman, L. (2011). Text mining and visualization using VOSviewer. https://arxiv.org/abs/1109.2058

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, *109*(2), 373-440.

Veličković, P., Casanova, A., Liò, P., Cucurull, G., Romero, A., & Bengio, Y. (2018). Graph attention networks. *6th International Conference on Learning Representations, ICLR 2018*, s. 39–41.

Verma, A., Khan, S. D., Maiti, J., & Krishna, O. B. (2014). Identifying patterns of safety related incidents in a steel plant using association rule mining of incident investigation reports. *Safety science*, *70*, 89-98.

Vidalakis, C., Abanda, F. H., & Oti, A. H. (2020). BIM adoption and implementation: focusing on SMEs. *Construction Innovation, 20*(1), 128-147.

Volk, R. Stengel, J., Schultmann, F. (2013). Building Information Modeling (BIM) for existing buildings — Literature review and future needs. *Automation in Construction, 38*, 19.

Volk, R., Luu, T. H., Mueller-Roemer, J. S., Sevilmis, N., & Schultmann, F. (2018). Deconstruction project planning of existing buildings based on automated acquisition and reconstruction of building information. *Automation in construction*, *91*, 226-245.

Vrahatis, A. G., Lazaros, K., & Kotsiantis, S. (2024). Graph attention networks: a comprehensive review of methods and applications. *Future Internet*, *16*(9), 318.

Vysotskiy, A. M., S.; Zolotova, J.; Tuchkevich, E. (2015). Features of BIM Implementation Using Autodesk Software. *Procedia Engineering, 117*, 10.

Wang, G., Ai, J., Mo, L., Yi, X., Wu, P., Wu, X., & Kong, L. (2023). Anomaly detection for data from unmanned systems via improved graph neural networks with attention mechanism. *Drones*, *7*(5), 326.

Wang, T., & Chen, H. M. (2023). Integration of building information modeling and project management in construction project life cycle. *Automation in Construction*, *150*, 104832.

Wang, X., Huang, X., Luo, Y., Pei, J., & Xu, M. (2018). Improving workplace hazard identification performance using data mining. *Journal of Construction Engineering and Management*, *144*(8), 04018068.

Wang, Y., Wang, X., & Liu, W. (2016). Unsupervised local deep feature for image recognition. *Information Sciences, 351*, 67-75.

Wang, Z., & Rezazadeh Azar, E. (2019). BIM-based draft schedule generation in reinforced concrete-framed buildings. *Construction Innovation, 19*(2), 280-294.

Wefki, H., Elnahla, M., & Elbeltagi, E. (2024). BIM-based schedule generation and optimization using genetic algorithms. *Automation in Construction, 164*, 105476.

Weldu, Y. W., & Knapp, G. M. (2012). Automated generation of 4D building information models through spatial reasoning. *Construction Research Congress 2012: Construction Challenges in a Flat World*, s. 612-621.

Wu, L., Sun, P., Hong, R., Fu, Y., Wang, X., & Wang, M. (2018). Socialgcn: An efficient graph convolutional network based model for social recommendation. https://arxiv.org/abs/1811.02815

Wu, Z., Chen, C., Cai, Y., Lu, C., Wang, H., & Yu, T. (2019). BIM-based visualization research in the construction industry: A network analysis. *International journal of environmental research and public health*, *16*(18), 3473.

Wuest, T., Irgens, C., & Thoben, K. D. (2014). An approach to monitoring quality in manufacturing using supervised machine learning on product state data. *Journal of Intelligent Manufacturing*, *25*, 1167-1180.

Xin, X., Tu, Y., Stojanovic, V., Wang, H., Shi, K., He, S., & Pan, T. (2022). Online reinforcement learning multiplayer non-zero sum games of continuous-time Markov jump linear systems. *Applied Mathematics and Computation*, *412*, 126537.

Xu, K., & Muñoz-Avila, H. (2008). CaBMA: a case-based reasoning system for capturing, refining, and reusing project plans. *Knowledge and Information Systems*, *15*, 215-232.

Xu, X., Du, H., & Lian, Z. (2022). Discussion on regression analysis with small determination coefficient in human-environment researches. *Indoor air*, *32*(10), e13117.

Xu, Y., Zhou, Y., Sekula, P., & Ding, L. (2021). Machine learning in construction: From shallow to deep learning. *Developments in the built environment*, *6*, 100045.

Xu, Z., Yuan, Y., Wei, H., & Wan, L. (2019). A serendipity-biased Deepwalk for collaborators recommendation. *PeerJ Computer Science*, *5*, e178.

Xu, Z., & Saleh, J. H. (2021). Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *Reliability Engineering & System Safety*, *211*, 107530.

Yarnold, J., Banihashemi, S., Lemckert, C., & Golizadeh, H. (2023). Building and construction quality: systematic literature review, thematic and gap analysis. *International Journal of Building Pathology and Adaptation*, *41*(5), 942-964.

Yavan, F. (2023). *Intelligent Optimization of Steel Structures Through BIM-Based Visual Programming Platform and Tools*. Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Trabzon.

Yeşilyurt, M. (2017). *4D/5D Modelleme Araçlarının Performanslarının Değerlendirilmesi*. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul.

You, X., Ma, Y., Liu, Z., Liu, J., & Zhang, M. (2021). Representation method of cooperative social network features based on Node2Vec model. *Computer Communications, 173*, 21-26.

Zangari, L., Interdonato, R., Calió, A., & Tagarelli, A. (2021). Graph convolutional and attention models for entity classification in multilayer networks. *Applied Network Science, 6*(1), 87.

Zhang, J. P., & Hu, Z. Z. (2011). BIM-and 4D-based integrated solution of analysis and management for conflicts and structural safety problems during construction: 1. Principles and methodologies. *Automation in construction*, *20*(2), 155-166.

Zhao, X., Yeoh, K. W., & Chua, D. K. H. (2020). Extracting construction knowledge from project schedules using natural language processing. *The 10th international conference on engineering, project, and production management*, s. 197-211.

Zheng, Z., Li, J., Zhu, L., Li, H., Petzold, F., & Tan, P. (2022). GAT-CADNet: graph attention network for panoptic symbol spotting in CAD drawings. *IEEE/CVF conference on computer vision and pattern recognition*, s. 11747-11756.

6. EKLER

6.1. Ek 1. GCN Modeli Python Kodu

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class GCNLayer(nn.Module):
    """
    A single Graph Convolutional Network (GCN) layer.
    """
    def __init__(self, in_features, out_features):
        """
        Args:
            in_features (int): Number of features for each input node.
            out_features (int): Number of output features for each node.
        """
        super(GCNLayer, self).__init__()
        self.linear = nn.Linear(in_features, out_features)

    def forward(self, X, A_hat):
        """
        Forward pass for the GCN layer.

        Args:
            X (torch.Tensor): Input node features of shape (N, in_features),
                              where N is the number of nodes.
            A_hat (torch.Tensor): Normalized adjacency matrix with self-loops of shape (N, N).

        Returns:
            torch.Tensor: Output node features of shape (N, out_features).
        """
        # Multiply the adjacency matrix with the input features.
        support = torch.mm(A_hat, X)
        # Apply the linear transformation.
        output = self.linear(support)
        return output

class GCN(nn.Module):
    """
    Graph Convolutional Network composed of two GCN layers.
    """
    def __init__(self, num_features, hidden_size, embedding_size):
        """
        Args:
            num_features (int): Number of input features per node.
            hidden_size (int): Number of features in the hidden layer.
            embedding_size (int): Number of output features (embeddings).
        """
        super(GCN, self).__init__()
        self.gcn1 = GCNLayer(num_features, hidden_size)
        self.relu = nn.ReLU()
        self.gcn2 = GCNLayer(hidden_size, embedding_size)
```

```
def forward(self, X, A_hat):
    """
    Forward pass for the GCN network.

    Args:
        X (torch.Tensor): Input node features of shape (N, num_features).
        A_hat (torch.Tensor): Normalized adjacency matrix with self-loops of shape (N, N).

    Returns:
        torch.Tensor: Node embeddings of shape (N, embedding_size).
    """
    h = self.gcn1(X, A_hat)
    h = self.relu(h)
    h = self.gcn2(h, A_hat)
    return h
```

6.2. Ek 2. GAT Modeli Python Kodu

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class GATLayer(nn.Module):
    """
    Graph Attention Layer as described in the GAT paper.
    """
    def __init__(self, in_features, out_features):
        """
        Args:
            in_features (int): Number of input features per node.
            out_features (int): Number of output features per node.
        """
        super(GATLayer, self).__init__()
        self.linear = nn.Linear(in_features, out_features, bias=False)
        self.att_weight = nn.Parameter(torch.empty(2 * out_features, 1))

        # Initialize weights using Xavier uniform initialization.
        nn.init.xavier_uniform_(self.linear.weight, gain=1.414)
        nn.init.xavier_uniform_(self.att_weight, gain=1.414)

    def forward(self, h, adj):
        """
        Forward pass of the GAT layer.

        Args:
            h (torch.Tensor): Input node features of shape (N, in_features),
                              where N is the number of nodes.
            adj (torch.Tensor): Adjacency matrix of shape (N, N).

        Returns:
            torch.Tensor: Output node features of shape (N, out_features).
        """
        # Apply linear transformation.
        Wh = self.linear(h)  # (N, out_features)
        N = Wh.size(0)

        # Prepare tensors for attention computation.
        # Wh_i: (N, N, out_features) where each row is repeated across columns.
        Wh_i = Wh.unsqueeze(1).expand(-1, N, -1)
        # Wh_j: (N, N, out_features) where each column is repeated across rows.
        Wh_j = Wh.unsqueeze(0).expand(N, -1, -1)

        # Concatenate Wh_i and Wh_j along the feature dimension.
        a_input = torch.cat([Wh_i, Wh_j], dim=-1)  # (N, N, 2*out_features)

        # Compute attention scores.
        e = torch.matmul(a_input, self.att_weight).squeeze(-1)  # (N, N)

        # Mask out non-existent edges by assigning -infinity.
        zero_vec = torch.full_like(e, float('-inf'))
        attention = torch.where(adj > 0, e, zero_vec)

        # Normalize attention scores.
        attention = F.softmax(attention, dim=1)
```

```
        # Aggregate the features using the attention coefficients.
        h_prime = torch.matmul(attention, Wh)  # (N, out_features)
        return h_prime


class GAT(nn.Module):
    """
    Graph Attention Network composed of two GAT layers.
    """
    def __init__(self, num_features, hidden_size, embedding_size):
        """
        Args:
            num_features (int): Number of features in the input.
            hidden_size (int): Number of features in the hidden layer.
            embedding_size (int): Number of features in the output layer (embeddings).
        """
        super(GAT, self).__init__()
        self.gat1 = GATLayer(num_features, hidden_size)
        self.elu = nn.ELU()
        self.gat2 = GATLayer(hidden_size, embedding_size)

    def forward(self, h, adj):
        """
        Forward pass of the GAT network.

        Args:
            h (torch.Tensor): Input node features of shape (N, num_features).
            adj (torch.Tensor): Adjacency matrix of shape (N, N).

        Returns:
            torch.Tensor: Node embeddings of shape (N, embedding_size).
        """
        h = self.gat1(h, adj)
        h = self.elu(h)
        h = self.gat2(h, adj)
        return h
```

6.3. Ek 3. GraphSAGE Modeli Python Kodu

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class GraphSAGELayer(nn.Module):
    """
    A single GraphSAGE layer that aggregates neighbor features,
    concatenates them with the node's own features, and applies a linear transformation.
    """
    def __init__(self, in_features, out_features):
        """
        Args:
            in_features (int): Number of features for each input node.
            out_features (int): Number of output features for each node.
        """
        super(GraphSAGELayer, self).__init__()
        # Since we concatenate the original features with the aggregated features,
        # the input dimension for the linear layer is doubled.
        self.linear = nn.Linear(in_features * 2, out_features)

    def forward(self, h, adj):
        """
        Forward pass for the GraphSAGE layer.

        Args:
            h (torch.Tensor): Input node features of shape (N, in_features), where N is the number of nodes.
            adj (torch.Tensor): Adjacency matrix of shape (N, N).

        Returns:
            torch.Tensor: Output node features of shape (N, out_features).
        """
        # Aggregate neighbor features using the adjacency matrix.
        agg_neighbors = torch.matmul(adj, h)

        # Concatenate the node's own features with its neighbors' aggregated features.
        h_concat = torch.cat([h, agg_neighbors], dim=1)

        # Apply the linear transformation and then a ReLU activation.
        h_out = self.linear(h_concat)
        h_out = F.relu(h_out)
        return h_out

class GraphSAGE(nn.Module):
    """
    GraphSAGE network composed of two GraphSAGE layers.
    """
    def __init__(self, num_features, hidden_size, embedding_size):
        """
        Args:
            num_features (int): Number of features in the input.
            hidden_size (int): Number of features in the hidden layer.
            embedding_size (int): Number of features in the output layer (embeddings).
        """
        super(GraphSAGE, self).__init__()
        self.sage1 = GraphSAGELayer(num_features, hidden_size)
        self.sage2 = GraphSAGELayer(hidden_size, embedding_size)
```

```
def forward(self, h, adj):
    """
    Forward pass for the GraphSAGE network.

    Args:
        h (torch.Tensor): Input node features of shape (N, num_features).
        adj (torch.Tensor): Adjacency matrix of shape (N, N).

    Returns:
        torch.Tensor: Node embeddings of shape (N, embedding_size).
    """
    h = self.sage1(h, adj)
    h = self.sage2(h, adj)
    return h
```

6.4. Ek 4. Node2vec Modeli Python Kodu

```
import networkx as nx
import pandas as pd
from node2vec import Node2Vec

def prepare_graph(df):
    """
    Prepare a directed graph from the input DataFrame.

    Parameters:
        df (pd.DataFrame): DataFrame with columns 'Activity Name' and 'Successors'.
                           Rows with '-' in 'Successors' are excluded.

    Returns:
        tuple: A tuple containing:
            - G (nx.DiGraph): Directed graph constructed from the DataFrame.
            - df_exploded (pd.DataFrame): DataFrame with each edge (activity-successor) per row.
    """
    df = df[['Activity Name', 'Successors']].copy()
    df = df[df['Successors'] != '-'].reset_index(drop=True)
    # Split the 'Successors' string into a list and explode it to create one edge per row
    df['Successors'] = df['Successors'].str.split(', ')
    df_exploded = df.explode('Successors').reset_index(drop=True)

    # Create a directed graph and add edges
    G = nx.DiGraph()
    for _, row in df_exploded.iterrows():
        G.add_edge(row['Activity Name'], row['Successors'])
    return G, df_exploded

def train_node2vec(G, dimensions=128, walk_length=20, num_walks=150, workers=2, window=5,
min_count=1):
    """
    Train a Node2Vec model on the given graph.

    Parameters:
        G (nx.DiGraph): Directed graph.
        dimensions (int): Dimensionality of the node embeddings.
        walk_length (int): Length of each random walk.
        num_walks (int): Number of random walks per node.
        workers (int): Number of worker threads.
        window (int): Window size for the skip-gram model.
        min_count (int): Minimum count threshold for nodes.

    Returns:
        model: Trained Node2Vec model.
    """
    node2vec_model = Node2Vec(G, dimensions=dimensions, walk_length=walk_length,
                              num_walks=num_walks, workers=workers)
    model = node2vec_model.fit(window=window, min_count=min_count)
    return model

def predict_successors_node2vec(activity, model, top_n=3):
    """
    Predict successors for a given activity using the Node2Vec model.

    Parameters:
```

```
    activity (str): The activity name.
    model: Trained Node2Vec model.
    top_n (int): Number of top similar nodes to return.

Returns:
    List[str]: Predicted successor activity names.
"""
if activity not in model.wv:
    return []
similar_nodes = model.wv.most_similar(activity, topn=top_n)
predicted = [node for node, _ in similar_nodes]
return predicted
```

6.5. Ek 5. DeepWalk Modeli Python Kodu

```
import networkx as nx
import pandas as pd
from node2vec import Node2Vec

def train_deepwalk(G, dimensions=64, walk_length=40, num_walks=10,
                   workers=2, window=5, min_count=1, p=1, q=1):
    """
    Train a DeepWalk model using Node2Vec with parameters similar to DeepWalk.

    Parameters:
        G (nx.DiGraph): Directed graph.
        dimensions (int): Dimensionality of the node embeddings.
        walk_length (int): Length of each random walk.
        num_walks (int): Number of random walks per node.
        workers (int): Number of worker threads.
        window (int): Window size for the skip-gram model.
        min_count (int): Minimum count threshold for nodes.
        p (float): Return hyperparameter.
        q (float): Inout hyperparameter.

    Returns:
        model: Trained DeepWalk model.
    """
    deepwalk_model = Node2Vec(G, dimensions=dimensions, walk_length=walk_length,
                              num_walks=num_walks, workers=workers, p=p, q=q)
    model_dw = deepwalk_model.fit(window=window, min_count=min_count)
    return model_dw

def predict_successors_deepwalk(activity, model, top_n=3):
    """
    Predict successors for a given activity using the DeepWalk model.

    Parameters:
        activity (str): The activity name.
        model: Trained DeepWalk model.
        top_n (int): Number of top similar nodes to return.

    Returns:
        List[str]: Predicted successor activity names.
    """
    if activity not in model.wv:
        return []
    similar_nodes = model.wv.most_similar(activity, topn=top_n)
    predicted = [node for node, _ in similar_nodes]
    return predicted
```

6.6. Ek 6. ARM Modeli Python kodu

```
import pandas as pd
from apyori import apriori  # Ensure you have installed the 'apyori' package

def prepare_transactions(df):
    """
    Prepare transactions by grouping the DataFrame by 'Activity Name' and aggregating
    the 'Successors' into a list.

    Parameters:
        df (pd.DataFrame): DataFrame containing at least 'Activity Name' and 'Successors' columns.

    Returns:
        List[List[str]]: A list of transactions, where each transaction is a list starting
                         with the activity name followed by its successors.
    """
    # Group by activity and collect successors
    grouped = df.groupby(['Activity Name'])['Successors'].apply(list).reset_index()
    # Create transactions by including the activity name as the first item
    transactions = [[row['Activity Name']] + row['Successors'] for _, row in grouped.iterrows()]
    return transactions

def generate_association_rules(transactions, min_support=0.001, min_confidence=0.1):
    """
    Generate association rules using the Apriori algorithm.

    Parameters:
        transactions (List[List[str]]): List of transactions.
        min_support (float): Minimum support threshold.
        min_confidence (float): Minimum confidence threshold.

    Returns:
        pd.DataFrame: DataFrame containing association rules with columns for Antecedent,
                      Consequent, Support, Confidence, and Lift.
    """
    association_results = list(apriori(transactions, min_support=min_support,
min_confidence=min_confidence))
    print(f"Total association rules generated: {len(association_results)}")

    rules_list = []
    for rule in association_results:
        for ordered_stat in rule.ordered_statistics:
            if ordered_stat.items_base and ordered_stat.items_add:
                rules_list.append({
                    'Antecedent': list(ordered_stat.items_base),
                    'Consequent': list(ordered_stat.items_add),
                    'Support': rule.support,
                    'Confidence': ordered_stat.confidence,
                    'Lift': ordered_stat.lift
                })
    rules_df = pd.DataFrame(rules_list)
    print("Sample Association Rules:")
    print(rules_df.head())
    return rules_df

def filter_rules(rules_df, df_train_exploded):
    """
```

```
    Filter the generated rules to include only valid activity names and successors.

    Parameters:
        rules_df (pd.DataFrame): DataFrame containing association rules.
        df_train_exploded (pd.DataFrame): Training DataFrame with columns 'Activity Name' and 'Successors'.

    Returns:
        pd.DataFrame: Filtered association rules DataFrame.
    """
    # Extract unique valid activity names and successors from training data
    activity_names_train = df_train_exploded['Activity Name'].unique()
    successors_train = df_train_exploded['Successors'].unique()

    def is_activity_name(items):
        return all(item in activity_names_train for item in items)

    def is_successor(items):
        return all(item in successors_train for item in items)

    filtered_rules_df = rules_df[
        rules_df['Antecedent'].apply(is_activity_name) &
        rules_df['Consequent'].apply(is_successor)
    ]
    print(f"Filtered rules count: {filtered_rules_df.shape[0]}")
    print(filtered_rules_df.head())
    return filtered_rules_df

def predict_successors(activity, rules_df, top_n=3):
    """
    Predict successors for a given activity based on association rules.

    Parameters:
        activity (str): The activity name.
        rules_df (pd.DataFrame): DataFrame containing association rules.
        top_n (int): Number of top predictions to return.

    Returns:
        List[str]: List of predicted successor activity names.
    """
    # Find rules where the given activity is in the antecedent
    relevant_rules = rules_df[rules_df['Antecedent'].apply(lambda x: activity in x)]
    if relevant_rules.empty:
        return []

    # Sort the rules by confidence in descending order
    relevant_rules = relevant_rules.sort_values(by='Confidence', ascending=False)
    consequents = relevant_rules['Consequent'].tolist()

    # Flatten the list and remove duplicates while preserving order
    predicted = []
    for consequent in consequents:
        for item in consequent:
            if item not in predicted:
                predicted.append(item)
            if len(predicted) == top_n:
                break
        if len(predicted) == top_n:
            break
```

```
    return predicted

def display_predictions(pred_df):
    """
    Display prediction results with the corresponding activity names.

    Parameters:
        pred_df (pd.DataFrame): DataFrame containing predictions with columns 'Activity Name',
                                'Actual Successors', and 'Predicted Successors'.
    """
    for index, row in pred_df.iterrows():
        activity = row['Activity Name']
        actual = row['Actual Successors']
        predicted = row['Predicted Successors']
        print(f"\nActivity: {activity}")
        print(f"Actual Successors: {actual}")
        print(f"Predicted Successors: {predicted}")
```

6.7. Ek 7. Aktivite Sıralamasının Dış Ortama Aktarılması Python Kodu

```
import ast
import re
import pandas as pd

# Example floor counts provided by the user
basements_count = 1  # e.g., 1 basement floor allowed (Level -1)
normal_count = 2     # e.g., 2 normal floors allowed (Level 1 and Level 2)
                     # Additionally, Level 3 is allowed only for "Çatı İmalatı" activities.


def get_level_number(activity_name):
    """
    Extracts the level number from the activity name.

    Example activity names:
        "Level -1-Grobeton İmalatı" or "Level 3- Çatı İmalatı"

    Args:
        activity_name (str): The activity name.

    Returns:
        int or None: The extracted level number, or None if not found.
    """
    match = re.search(r'Level\s*([\-]?\d+)', activity_name)
    if match:
        return int(match.group(1))
    return None


def is_activity_allowed(activity_name, basements_count, normal_count):
    """
    Determines whether an activity is allowed based on its floor level.

    Basement floors: allowed if level is in the range [-basements_count, -1].
    Normal floors: allowed if level is in the range [1, normal_count].
    Level 0 is always allowed.
    Special exception: if level is normal_count + 1 and the activity name contains "Çatı İmalatı",
    the activity is allowed.

    Args:
        activity_name (str): The name of the activity.
        basements_count (int): The number of allowed basement floors.
        normal_count (int): The number of allowed normal floors.

    Returns:
        bool: True if the activity is allowed, False otherwise.
    """
    level_num = get_level_number(activity_name)
    if level_num is None:
        # If no level number is found, the activity is not allowed.
        return False

    if level_num < 0:
        return -basements_count <= level_num < 0
    elif level_num == 0:
        return True
```

```
    elif level_num > 0:
        if level_num <= normal_count:
            return True
        elif level_num == normal_count + 1 and "Çatı İmalatı" in activity_name:
            return True
        else:
            return False


def build_work_plan(prediction_df, start_activity):
    """
    Builds a work plan (an ordered list of activity names) starting from the specified activity,
    following the predicted successors using breadth-first search.

    Args:
        prediction_df (pd.DataFrame): A DataFrame containing 'Activity Name' and 'Predicted Successors' columns.
        start_activity (str): The starting activity.

    Returns:
        list: A list of activity names representing the work plan.
    """
    work_plan = []
    visited = set()
    queue = [start_activity]

    while queue:
        current_activity = queue.pop(0)
        if current_activity not in visited:
            work_plan.append(current_activity)
            visited.add(current_activity)
            # Get predicted successors for the current activity
            row = prediction_df[prediction_df["Activity Name"] == current_activity]
            if not row.empty:
                successors = row["Predicted Successors"].values[0]
                if isinstance(successors, list):
                    queue.extend(successors)
                else:
                    queue.append(successors)
    return work_plan


def main():
    """
    Main function that generates a work plan based on prediction results.

    Note:
        The following variables/functions must be defined externally:
            - test_data: A DataFrame containing at least 'Activity Name' and 'Successors' columns.
            - filtered_rules_df: A DataFrame containing association rules (for the predict_successors function).
            - predict_successors: A function that returns predicted successors for a given activity.
            - trial_name: A string used for naming the output CSV file.
    """
    # Prepare test data
    df_test = test_data[['Activity Name', 'Successors']].copy()
    df_test = df_test[df_test["Successors"] != "-"].reset_index(drop=True)
    df_test["Successors"] = df_test["Successors"].str.split(", ")
    df_test_exploded = df_test.explode("Successors").reset_index(drop=True)
```

```
# Create transactions for each activity
grouped_test = df_test_exploded.groupby("Activity Name")["Successors"].apply(list).reset_index()

# Apply the prediction function for each activity
prediction_results = []
for _, row in grouped_test.iterrows():
    activity = row["Activity Name"]
    actual_successors = row["Successors"]
    predicted_successors = predict_successors(activity, filtered_rules_df, top_n=3)
    prediction_results.append({
        "Activity Name": activity,
        "Actual Successors": actual_successors,
        "Predicted Successors": predicted_successors
    })

prediction_df = pd.DataFrame(prediction_results)

# Convert strings in the 'Predicted Successors' column to lists, if necessary
prediction_df["Predicted Successors"] = prediction_df["Predicted Successors"].apply(
    lambda x: ast.literal_eval(x) if isinstance(x, str) else x
)

# Build the work plan starting from a designated activity
start_activity = "Level -1-Grobeton İmalatı"  # This can also be user-specified
work_plan = build_work_plan(prediction_df, start_activity)
work_plan_df = pd.DataFrame(work_plan, columns=["Activity Name"])

# Filter the prediction DataFrame to include only allowed activities based on their floor levels
prediction_df = prediction_df[
    prediction_df["Activity Name"].apply(
        lambda x: is_activity_allowed(x, basements_count, normal_count)
    )
].reset_index(drop=True)

# Also filter the activities in the 'Predicted Successors' column
filtered_predictions = []
for _, row in prediction_df.iterrows():
    allowed_successors = [
        act for act in row["Predicted Successors"]
        if is_activity_allowed(act, basements_count, normal_count)
    ]
    filtered_predictions.append({
        "Activity Name": row["Activity Name"],
        "Actual Successors": [
            act for act in row["Actual Successors"]
            if is_activity_allowed(act, basements_count, normal_count)
        ],
        "Predicted Successors": allowed_successors
    })
prediction_df = pd.DataFrame(filtered_predictions)

# Filter the work plan DataFrame as well
work_plan_df = work_plan_df[
    work_plan_df["Activity Name"].apply(
        lambda x: is_activity_allowed(x, basements_count, normal_count)
    )
].reset_index(drop=True)

# Identify non-predicted activities in the work plan
```

```
    pred_activities = set(prediction_df["Activity Name"])
    work_plan_activities = set(work_plan_df["Activity Name"])
    non_predicted_in_workplan = pred_activities - work_plan_activities

    # Save the work plan to a CSV file
    work_schedule_csv_path = (
        f"/content/drive/MyDrive/Colab Notebooks/OtherS/Activity Sequencing/Results/"
        f"ML_Aktivite Sıralama_{trial_name}.csv"
    )
    work_plan_df.to_csv(work_schedule_csv_path, index=False, encoding="utf-8-sig")

    # Append non-predicted activities to the CSV file
    with open(work_schedule_csv_path, "a", encoding="utf-8-sig") as f:
        f.write("\n")
        f.write("Non-predicted Activities\n")
        for activity in non_predicted_in_workplan:
            f.write(activity + "\n")

    print("Work Schedule with non-predicted activities appended (with floor filtering):")
    print(work_plan_df)


if __name__ == "__main__":
    main()
```

6.8. Ek 8. BIM Modeli ve ML Model Çıktısının İlişkilendirilerek İş Programının Dış Ortama Aktarılması Python Düğüm Kodu

```
import sys
import datetime
import re
import pandas as pd

# Get inputs from Dynamo:
# IN[0]: CSV file content (string or list of strings)
# IN[1]: New CSV file headers
# IN[2]: List of names to compare against
# IN[3]: Default value for the "Task Type" column (e.g., "Construct")
# IN[4]: Project start date
# IN[5]: Durations for each activity (in days)
# IN[6]: File path for output CSV
# IN[7]: List of keywords to determine Task Type (e.g., ["kalıp"])
# IN[8]: Corresponding Task Type values for the keywords (e.g., ["Temporary"])
csv_data_input = IN[0]
headers = IN[1]
name_list = IN[2]
default_task_type = IN[3]
project_start_date = IN[4]
durations_list = IN[5]
file_path = IN[6]
keywords_list = IN[7]
task_types_list = IN[8]

# If project_start_date is a list, use its first element.
if isinstance(project_start_date, list) and project_start_date:
    project_start_date = project_start_date[0]

# If file_path is a list, use its first element.
if isinstance(file_path, list) and file_path:
    file_path = file_path[0]

# Validate that file_path is a non-empty string.
if not isinstance(file_path, str) or file_path.strip() == '':
    raise Exception("A valid file path was not provided.")

# Check that keywords_list and task_types_list have the same length.
if len(keywords_list) != len(task_types_list):
    error_msg = (
        f"Length mismatch between IN[7] (keywords_list) and IN[8] (task_types_list).\n"
        f"keywords_list length: {len(keywords_list)}\n"
        f"task_types_list length: {len(task_types_list)}"
    )
    raise Exception(error_msg)


def is_name_in_list(name, name_list):
    """
    Check if the given name exists in the provided list (case-insensitive).
    This function handles nested lists.

    Args:
        name (str): The name to search for.
```

```
    name_list (list): A list (or nested list) of names.

  Returns:
    bool: True if the name is found; otherwise, False.
  """
  if isinstance(name_list, list):
    for item in name_list:
      if isinstance(item, list):
        if is_name_in_list(name, item):
          return True
      else:
        if str(item).strip().lower() == name.lower():
          return True
  else:
    if str(name_list).strip().lower() == name.lower():
      return True
  return False


# Split the CSV data into rows.
if isinstance(csv_data_input, list):
  csv_rows = [row.strip() for row in csv_data_input if row.strip() != '']
else:
  csv_rows = csv_data_input.strip().split('\n')

# Assume the first row contains headers.
csv_headers = csv_rows[0].split(',')
csv_content = csv_rows[1:]

# Find the index of the 'Activity Name' column.
try:
  activity_name_index = csv_headers.index('Activity Name')
except ValueError:
  error_message = (
    "'Activity Name' column was not found in the CSV. Available headers: " +
    ', '.join(csv_headers)
  )
  raise Exception(error_message)

# Extract a list of activity names from the CSV content.
activity_names_list = []
for row in csv_content:
  columns = row.split(',')
  if len(columns) > activity_name_index:
    activity_name = columns[activity_name_index].strip()
    activity_names_list.append(activity_name)

# Parse the project start date into a datetime object.
try:
  project_start_date_str = str(project_start_date).strip()
  try:
    project_start_date_dt = datetime.datetime.strptime(project_start_date_str, "%d/%m/%Y %I:%M %p")
  except ValueError:
    try:
      project_start_date_dt = datetime.datetime.strptime(project_start_date_str, "%m/%d/%Y %I:%M %p")
    except ValueError:
      try:
        project_start_date_dt = datetime.datetime.strptime(project_start_date_str, "%Y-%m-%d
%H:%M:%S")
```

```
            except ValueError:
                raise Exception("Project start date is in an unrecognized format: " + project_start_date_str)
    except Exception as e:
        raise Exception("Project start date is not in a valid datetime or string format: " + str(project_start_date))

    # Build the new CSV output.
    output_data = []

    # Normalize and add headers.
    headers_normalized = [header.strip() for header in headers]
    output_data.append(','.join(headers_normalized))

    # Ensure name_list and durations_list have matching lengths.
    if len(name_list) != len(durations_list):
        raise Exception("The lengths of IN[2] (name_list) and IN[5] (durations_list) do not match.")

    # Create a mapping of names to durations.
    name_duration_dict = {}
    for name, duration in zip(name_list, durations_list):
        name_duration_dict[str(name).strip()] = duration

    # Initialize a Task ID counter.
    task_id_counter = 1

    # Plan the activities by calculating planned start and end dates.
    current_start_date = project_start_date_dt
    for activity_name in activity_names_list:
        # Check if the activity is in name_list.
        if is_name_in_list(activity_name, name_list):
            # Get the duration for the activity.
            duration = name_duration_dict.get(activity_name, 0)
            try:
                duration = int(duration)
            except ValueError:
                duration = 0  # Default value for invalid durations

            # Calculate the planned start and end dates.
            planned_start_date = current_start_date
            planned_end_date = planned_start_date + datetime.timedelta(days=duration - 1)

            # Format the dates.
            formatted_planned_start = planned_start_date.strftime("%d/%m/%Y")
            formatted_planned_end = planned_end_date.strftime("%d/%m/%Y")

            # Determine the Task Type based on keywords in the activity name.
            task_type = default_task_type  # Use default value
            activity_name_lower = activity_name.lower()
            for keyword, task_type_option in zip(keywords_list, task_types_list):
                if keyword.lower() in activity_name_lower:
                    task_type = task_type_option
                    break  # Stop at the first match

            # Build the CSV row.
            row = []
            for header in headers:
                header_clean = header.strip()
                if header_clean == 'Task ID':
                    row.append(str(task_id_counter))
                elif header_clean == 'Task Name':
```

```
            row.append(activity_name)
        elif header_clean == 'Task Type':
            row.append(task_type)
        elif header_clean == 'Planned Start':
            row.append(formatted_planned_start)
        elif header_clean == 'Planned End':
            row.append(formatted_planned_end)
        else:
            row.append('')  # Leave other columns empty
    output_data.append(','.join(row))

    # Increment the Task ID counter.
    task_id_counter += 1

    # Update current_start_date for the next activity.
    current_start_date = planned_end_date + datetime.timedelta(days=1)
  else:
    # Skip activities not in name_list.
    continue

# Create the output CSV text.
output_text = '\n'.join(output_data)

# Write the output text to the specified file (using utf-8-sig encoding).
with open(file_path, 'w', encoding='utf-8-sig') as f:
  f.write(output_text)

# Set the output to the file path.
OUT = file_path
```

6.9. Ek 9. Revit Eklentisi C# Kodu

```
using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using Dynamo.Applications;
using Dynamo.Utilities;
using DynamoServices;
using DynamoCore;
using System;
using System.Collections.Generic;
using System.IO;
using System.Reflection;
using System.Windows.Media.Imaging;
using TaskDialog = Autodesk.Revit.UI.TaskDialog;
using TaskDialogCommonButtons = Autodesk.Revit.UI.TaskDialogCommonButtons;
using TaskDialogIcon = Autodesk.Revit.UI.TaskDialogIcon;

namespace Dynamov4
{
   /// <summary>
   /// Revit uygulaması: Ribbon butonlarını oluşturur.
   /// </summary>
   public class App : IExternalApplication
   {
      /// <summary>
      /// Revit başlatıldığında çalışır. Ribbon paneli oluşturur ve iki adet push button ekler.
      /// </summary>
      /// <param name="application">UIControlledApplication nesnesi.</param>
      /// <returns>Result.Succeeded</returns>
      public Result OnStartup(UIControlledApplication application)
      {
         // Yeni bir Ribbon paneli oluştur.
         RibbonPanel ribbonPanel = application.CreateRibbonPanel("BIM Auto Scheduling");

         // Geçerli derlemenin konumunu al.
         string assemblyPath = Assembly.GetExecutingAssembly().Location;

         // Birinci push button: "BIMAuto Scheduling"
         PushButtonData buttonData = new PushButtonData("cmdMyTest", "BIMAuto\nScheduling",
assemblyPath, "Dynamov4.Run2Dynamo");
         PushButton pushButton = ribbonPanel.AddItem(buttonData) as PushButton;
         pushButton.ToolTip = "Revit modellerinden otomatik iş programı yapmak ve Navisworks
modellerinden 4B simülasyonlar oluşturmak için geliştirilmiştir.";

         // Push button için ikon ayarla.
         Uri iconUri = new
Uri(@"C:\Users\umitb\OneDrive\Desktop\DynamoKodları\BIMAutoSchedulingIcon.png");
         BitmapImage iconImage = new BitmapImage(iconUri);
         pushButton.LargeImage = iconImage;

         // İkinci push button: "About"
         PushButtonData buttonData2 = new PushButtonData("cmdMyTest2", "About", assemblyPath,
"Dynamov4.About");
         PushButton pushButton2 = ribbonPanel.AddItem(buttonData2) as PushButton;
         pushButton2.ToolTip = "About";

         // About butonu için ikon ayarla.
```

```
            Uri iconUri2 = new Uri(@"C:\Users\umitb\OneDrive\Desktop\DynamoKodları\AboutIcon.png");
            BitmapImage iconImage2 = new BitmapImage(iconUri2);
            pushButton2.LargeImage = iconImage2;

            return Result.Succeeded;
        }

        /// <summary>
        /// Revit kapatılırken çalışır.
        /// </summary>
        /// <param name="application">UIControlledApplication nesnesi.</param>
        /// <returns>Result.Succeeded</returns>
        public Result OnShutdown(UIControlledApplication application)
        {
            // Temizlik işlemleri yapılabilir.
            return Result.Succeeded;
        }
    }

    /// <summary>
    /// Dynamo script'lerini çalıştıran sınıf.
    /// </summary>
    [Transaction(TransactionMode.Manual)]
    [Regeneration(RegenerationOption.Manual)]
    public class Run2Dynamo : IExternalCommand
    {
        /// <summary>
        /// Dynamo script'lerini sırayla çalıştırır.
        /// </summary>
        /// <param name="commandData">ExternalCommandData nesnesi.</param>
        /// <param name="message">Hata mesajı (varsa).</param>
        /// <param name="elements">Seçili elementler.</param>
        /// <returns>Result.Succeeded veya hata kodu.</returns>
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
        {
            // Masaüstü yolunu al ve Dynamo dosyalarının yolunu oluştur.
            string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            string journalDynamoPath1 = Path.Combine(desktopPath, "DynamoKodları", "1-Proje
Parametrelerinin Oluşturulması.dyn");
            string journalDynamoPath2 = Path.Combine(desktopPath, "DynamoKodları", "2-Varsayılan
Değerler.dyn");
            string journalDynamoPath3 = Path.Combine(desktopPath, "DynamoKodları", "3-Kalıp Metrajının ve
Merdiven Beton Hacminin Hesaplanması.dyn");
            string journalDynamoPath4 = Path.Combine(desktopPath, "DynamoKodları", "4-Kullanıcı Arayüzü
ile Varsayılan Değerlerin Kontrolü.dyn");
            string journalDynamoPath5 = Path.Combine(desktopPath, "DynamoKodları", "5-Aktivite Sürelerinin
Hesaplanması.dyn");
            string journalDynamoPath6 = Path.Combine(desktopPath, "DynamoKodları", "6-Metraj Listelerinin
Oluşturulması.dyn");
            string journalDynamoPath7 = Path.Combine(desktopPath, "DynamoKodları", "7-Aktivite Adlarının
Oluşturulması.dyn");
            string journalDynamoPath8 = Path.Combine(desktopPath, "DynamoKodları", "8-Navisworks Search
Sets Dosyasının Oluşturulması.dyn");
            string journalDynamoPath9 = Path.Combine(desktopPath, "DynamoKodları", "9-ML Model Çıktısı
ile İlişkilendirme ve İş Programının Elde Edilmesi.dyn");

            DynamoRevit dynamoRevit = new DynamoRevit();

            // Ortak journal verilerini oluşturan yardımcı metot.
```

```
IDictionary<string, string> CreateJournalData()
{
   return new Dictionary<string, string>
   {
      { JournalKeys.ShowUiKey, false.ToString() },
      { JournalKeys.AutomationModeKey, true.ToString() },
      { JournalKeys.DynPathKey, string.Empty },
      { JournalKeys.DynPathExecuteKey, true.ToString() },
      { JournalKeys.ForceManualRunKey, false.ToString() },
      { JournalKeys.ModelShutDownKey, true.ToString() },
      { JournalKeys.ModelNodesInfo, false.ToString() }
   };
}

// Belirtilen Dynamo dosyasını çalıştıran yardımcı metot.
Result ExecuteDynamoCommand(string filePath)
{
   IDictionary<string, string> journalData = CreateJournalData();
   DynamoRevitCommandData dynCommandData = new DynamoRevitCommandData
   {
      Application = commandData.Application,
      JournalData = journalData
   };

   dynamoRevit.ExecuteCommand(dynCommandData);
   DynamoRevit.RevitDynamoModel.OpenFileFromPath(filePath, true);
   DynamoRevit.RevitDynamoModel.ForceRun();

   return dynamoRevit.ExecuteCommand(dynCommandData);
}

// Dynamo script'lerini sırayla çalıştır ve her adımda sonucu kontrol et.
Result result = ExecuteDynamoCommand(journalDynamoPath1);
if (result != Result.Succeeded)
   return result;

result = ExecuteDynamoCommand(journalDynamoPath2);
if (result != Result.Succeeded)
   return result;

result = ExecuteDynamoCommand(journalDynamoPath3);
if (result != Result.Succeeded)
   return result;

result = ExecuteDynamoCommand(journalDynamoPath4);
if (result != Result.Succeeded)
   return result;

result = ExecuteDynamoCommand(journalDynamoPath5);
if (result != Result.Succeeded)
   return result;

result = ExecuteDynamoCommand(journalDynamoPath6);
if (result != Result.Succeeded)
   return result;

result = ExecuteDynamoCommand(journalDynamoPath7);
if (result != Result.Succeeded)
   return result;
```

```
            result = ExecuteDynamoCommand(journalDynamoPath8);
            if (result != Result.Succeeded)
                return result;

            result = ExecuteDynamoCommand(journalDynamoPath9);
            if (result != Result.Succeeded)
                return result;

            TaskDialog.Show("BIMAutoScheduling v.1.0", "İşlem tamamlandı.");
            return Result.Succeeded;
        }
    }

    /// <summary>
    /// Global ayarları tutan sınıf.
    /// </summary>
    public static class Globals
    {
        /// <summary>
        /// Eklenti versiyonu.
        /// </summary>
        public static string Version = "1.0";
    }

    /// <summary>
    /// About penceresini gösteren sınıf.
    /// </summary>
    [Transaction(TransactionMode.Manual)]
    [Regeneration(RegenerationOption.Manual)]
    public class About : IExternalCommand
    {
        /// <summary>
        /// About penceresini çalıştırır.
        /// </summary>
        /// <param name="commandData">ExternalCommandData nesnesi.</param>
        /// <param name="message">Hata mesajı (varsa).</param>
        /// <param name="elements">Seçili elementler.</param>
        /// <returns>Result.Succeeded</returns>
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
        {
            TaskDialog aboutDialog = new TaskDialog($"BIMAutoScheduling v.{Globals.Version}")
            {
                MainIcon = TaskDialogIcon.TaskDialogIconInformation,
                MainInstruction = "BIMAutoScheduling",
                MainContent = @"   Bu eklenti, Revit modellerinden otomatik iş programı yapmak ve Navisworks
```

modellerinden 4B simülasyonlar oluşturmak için geliştirilmiştir. Dynamo ve Python ile entegre çalışarak, iş programı oluşturma sürecinde zamandan tasarruf edilmesini hedeflemektedir.

Eklentide, donatı metrajları toplam inşaat alanı dikkate alınarak yaklaşık olarak hesaplanmaktadır. Elde edilen donatı metrajı, temel elemanları için tanımlanan 'Donatı Metrajı' parametresinde görülmektedir. Donatı metrajı dikkate alınarak donatı imalat süresi hesaplanmaktadır. Toplam donatı imalat süresi, temel ve döşeme sayısına bölünerek taşıyıcı elemanların ilgili parametresine eklenmektedir. Donatı metrajının değiştirilmesi istenildiğinde, temelde yer alan parametreden veri girişi yapılması gerekmektedir. Böylece, donatı imalat süreleri de değişecektir.

Eklentide, kalıp metrajları toplamı, 'Toplam Kalıp Metrajı' parametresi ile temel elemanına eklenmektedir. Toplam kalıp imalat süresi, döşeme sayısına bölünerek taşıyıcı elemanların ilgili parametresine

eklenmektedir. Kalıp metrajının değiştirilmesi istenildiğinde, temelde yer alan parametreden veri girişi yapılması gerekmektedir. Böylece, kalıp imalat süreleri de değişecektir.

Eklenti tamamlandıktan sonra Revit modelinizin bulunduğu klasörde Navisworks Search Sets için .xml ve 4B Simülasyon için .csv dosyalarınız oluşturulacaktır.

Herhangi bir sorun ile karşılaşırsanız lütfen geliştirici ile iletişime geçiniz.

Program Gereksinimleri - Revit 2024 ve üzeri, Dynamo 2.19.3 ve üzeri, .NET Framework 4.8

Geliştirici Bilgileri
Ad-Soyad: Ümit Bahadır
E-posta: umitbahadir@ktu.edu.tr",

```
            CommonButtons = TaskDialogCommonButtons.Ok
        };

        aboutDialog.Show();
        return Result.Succeeded;
    }
  }
}
```

6.10. Ek 10. Navisworks Eklentisi C# Kodu

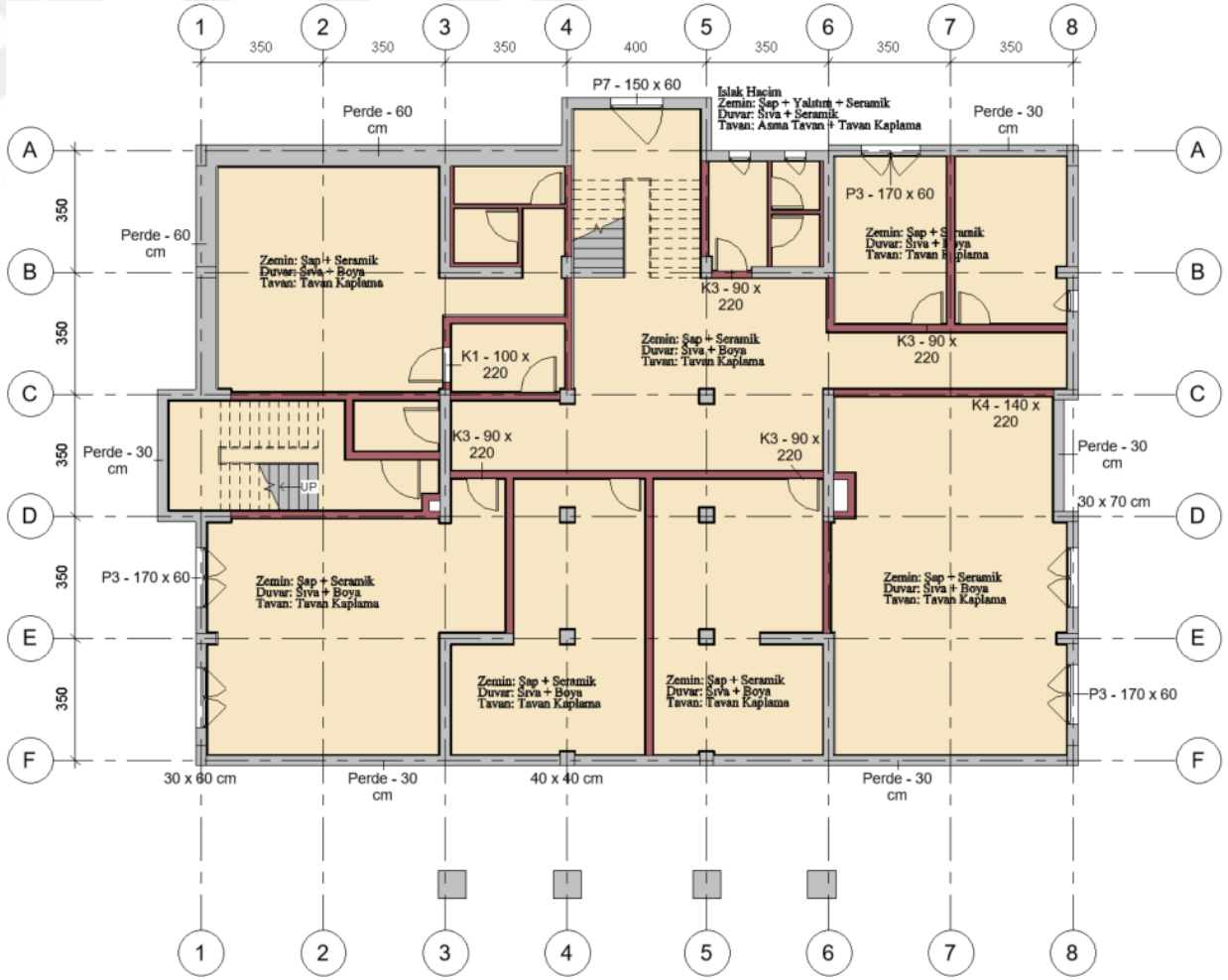
```
using System;
using System.IO;
using System.Globalization;
using System.Windows.Forms;
using Autodesk.Navisworks.Api;
using Autodesk.Navisworks.Api.Plugins;
using Autodesk.Navisworks.Api.Timeliner;

namespace AddDataSources
{
   /// <summary>
   /// BIM Auto Scheduling - Import CSV eklentisi.
   /// CSV dosyasından timeliner data source oluşturur.
   /// </summary>
   [Plugin("CreateTaskFromDataSource", "UB", DisplayName = "BIM Auto Scheduling - Import CSV")]
   [AddInPlugin(AddInLocation.None)]
   [Interface("TimelinerDataSourceProvider", "Navisworks", DisplayName = "BIM Auto Scheduling -
Import CSV")]
   public sealed class CreateTaskFromDataSourcePlugin : TimelinerDataSourceProvider
   {
      /// <summary>
      /// Varsayılan yapıcı.
      /// </summary>
      public CreateTaskFromDataSourcePlugin()
      {
      }

      /// <summary>
      /// Verilen gösterim adına göre data source oluşturur.
      /// </summary>
      /// <param name="displayName">Data source için gösterim adı.</param>
      /// <returns>Oluşturulan TimelinerDataSource nesnesi.</returns>
      public override TimelinerDataSource CreateDataSource(string displayName)
      {
         MessageBox.Show(
            ".csv uzantılı iş programı dosyanızı seçiniz. Açılan ekranda görünen ayarlar otomatik yapılacaktır.
Herhangi bir sütun seçmeniz gerekmemektedir. \n\nİş programı yüklendikten sonra Rebuild Task Hierarchy
veya Synchronize seçeneklerinden size uygun olanını seçmeyi unutmayınız.",
            "BIM Auto Scheduling - Import CSV",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);

         string filePath = string.Empty;
         using (OpenFileDialog openFileDialog = new OpenFileDialog())
         {
            openFileDialog.InitialDirectory                                                    =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            openFileDialog.Filter = "CSV files (*.csv)|*.csv|All files (*.*)|*.*";
            openFileDialog.FilterIndex = 2;
            openFileDialog.RestoreDirectory = true;

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
               filePath = openFileDialog.FileName;
            }
         }
```

```
    TimelinerDataSource dataSource = new TimelinerDataSource(displayName)
    {
        ProjectIdentifier = filePath,
        DataSourceProviderId = base.Id,
        DataSourceProviderVersion = 1.0,
        DataSourceProviderName = "CreateTaskFromDataSource"
    };

    AddAvailableFields(dataSource);
    return dataSource;
}

/// <summary>
/// Data source'dan görevleri içe aktarır.
/// </summary>
/// <param name="dataSource">İçe aktarılacak data source.</param>
/// <returns>İçe aktarma işleminin sonucunu içeren nesne.</returns>
protected override TimelinerImportTasksResult ImportTasksCore(TimelinerDataSource dataSource)
{
    TimelinerImportTasksResult importResult = new TimelinerImportTasksResult
    {
        TaskTypeWasSet = true
    };

    // Kök görev oluşturuluyor.
    TimelinerTask rootTask = new TimelinerTask
    {
        SynchronizationId = TimelinerTask.DataSourceRootTaskIdentifier
    };

    using (StreamReader reader = new StreamReader(dataSource.ProjectIdentifier))
    {
        // İlk satır (başlık) okunuyor ve atlanıyor.
        reader.ReadLine();

        while (!reader.EndOfStream)
        {
            string line = reader.ReadLine();
            string[] fields = line.Split(',');

            TimelinerTask task = new TimelinerTask
            {
                SynchronizationId = fields[0],
                DisplayName = fields[1],
                SimulationTaskTypeName = fields[2],
                PlannedStartDate = ParseDateTime(fields[3]),
                PlannedEndDate = ParseDateTime(fields[4])
            };

            SelectionSourceCollection selectionSources = GetSelectionSetByName(fields[1]);
            if (selectionSources != null && selectionSources.Count > 0)
            {
                task.Selection.CopyFrom(selectionSources);
            }

            rootTask.Children.Add(task);
        }
    }
```

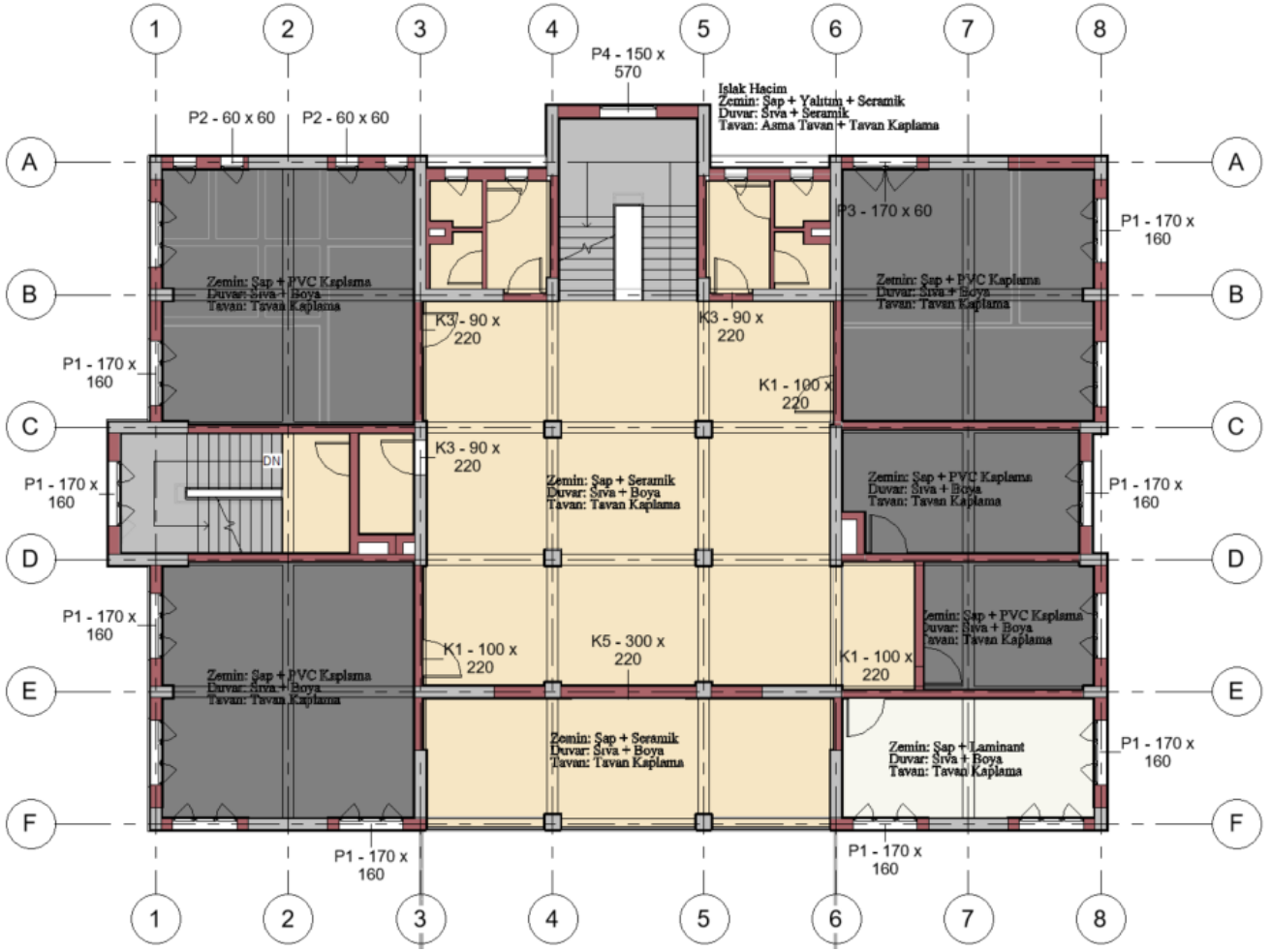
```
    importResult.RootTask = rootTask;
    return importResult;
}

/// <summary>
/// Data source güncellendiğinde çağrılır.
/// </summary>
/// <param name="dataSource">Güncellenecek data source.</param>
public override void UpdateDataSource(TimelinerDataSource dataSource)
{
    FileReferenceResolver resolver = new FileReferenceResolver();
    FileReferenceResolveResult result = resolver.Resolve(dataSource.ProjectIdentifier);
    if (result.Response == FileResolutionResponse.OK)
    {
        dataSource.ProjectIdentifier = result.FileNameToOpen;
    }

    if (dataSource.AvailableFields.Count == 0)
    {
        AddAvailableFields(dataSource);
    }
}

/// <summary>
/// Timeliner data source'a ek alanlar ekler.
/// </summary>
/// <param name="dataSource">Data source.</param>
public void AddAvailableFields(TimelinerDataSource dataSource)
{
    dataSource.AvailableFields.Add(new TimelinerDataSourceField("UniqueID", "UniqueId"));
    dataSource.AvailableFields.Add(new TimelinerDataSourceField("DisplayName", "Display Name"));
    dataSource.AvailableFields.Add(new TimelinerDataSourceField("PlannedStart", "Planned Start"));
    dataSource.AvailableFields.Add(new TimelinerDataSourceField("PlannedEnd", "Planned End"));
    dataSource.AvailableFields.Add(new TimelinerDataSourceField("TaskType", "Task Type"));
}

/// <summary>
/// Belirtilen isimdeki seçim setini döndürür.
/// </summary>
/// <param name="name">Seçim seti adı.</param>
/// <returns>Bulunan seçim seti koleksiyonu; bulunamazsa boş koleksiyon.</returns>
public SelectionSourceCollection GetSelectionSetByName(string name)
{
    Document doc = Autodesk.Navisworks.Api.Application.ActiveDocument;
    SavedItemCollection selectionSets = doc.SelectionSets.RootItem.Children;
    foreach (SavedItem set in selectionSets)
    {
        if (set.DisplayName == name)
        {
            SelectionSet selectionSet = set as SelectionSet;
            if (selectionSet != null)
            {
                SelectionSource selSource = doc.SelectionSets.CreateSelectionSource(selectionSet);
                return new SelectionSourceCollection { selSource };
            }
        }
    }
    return new SelectionSourceCollection();
```

```
        }

        /// <summary>
        /// Belirtilen tarih metnini "gün/ay/yıl" formatında DateTime nesnesine çevirir.
        /// </summary>
        /// <param name="dateString">Tarih metni.</param>
        /// <returns>Çevrilen DateTime nesnesi.</returns>
        public DateTime ParseDateTime(string dateString)
        {
            string[] parts = dateString.Split('/');
            int day = int.Parse(parts[0]);
            int month = int.Parse(parts[1]);
            int year = int.Parse(parts[2]);
            return new DateTime(year, month, day);
        }

        /// <summary>
        /// Data source ayarlarını doğrular.
        /// </summary>
        /// <param name="dataSource">Doğrulanacak data source.</param>
        /// <returns>Doğrulama sonucunu içeren nesne.</returns>
        public override TimelinerValidateSettingsResult ValidateSettings(TimelinerDataSource dataSource)
        {
            return new TimelinerValidateSettingsResult(true);
        }

        /// <summary>
        /// Yönetilen kaynakların temizlenmesi.
        /// </summary>
        protected override void DisposeManagedResources()
        {
            // Yönetilen kaynak yok.
        }

        /// <summary>
        /// Yönetilmeyen kaynakların temizlenmesi.
        /// </summary>
        protected override void DisposeUnmanagedResources()
        {
            // Yönetilmeyen kaynak yok.
        }

        /// <summary>
        /// Eklentinin kullanılabilir olup olmadığını belirtir.
        /// </summary>
        public override bool IsAvailable => true;
    }
}
```

6.11. Ek 11. Örnek Proje Kalıp Planları ve Mimari Planlar



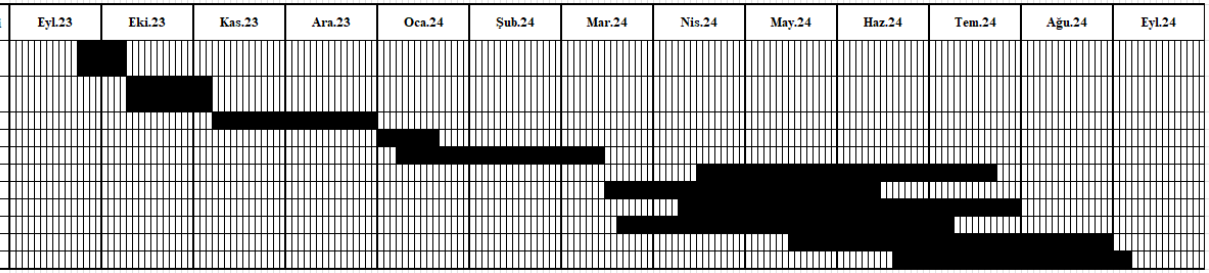
Örnek proje bodrum kat mimari plan



Örnek proje 1. kat mimari plan

6.12. Ek 12. ÇŞB Örnek İş Programı Gantt şeması

	İş Programı			Aktivite Süresi (gün)	Başlangıç Tarihi	Bitiş Tarihi
1	İnşaat	A-Su Basman Altı Grubu	Kazı Dolgu Grobeton ve Yalıtım İmalatları	16	23.09.2023	08.10.2023
2			Su Basman Altı Betonarme İmalatları	28	09.10.2023	06.11.2023
3		B-Kat Betonarme Grubu		54	07.11.2023	30.12.2023
4		C-Çatı Grubu		20	01.01.2024	20.01.2024
5		D-Duvar Grubu		68	07.01.2024	14.03.2024
6		E-Kaplamalar Grubu	Döşeme Kaplamaları	98	15.04.2024	22.07.2024
7			Duvar Kaplamaları	90	15.03.2024	14.06.2024
8			Tavan Kaplamaları	112	09.04.2024	30.07.2024
9		F-Cephe Grubu		110	19.03.2024	08.07.2024
10		G-Doğramalar		106	15.05.2024	30.08.2024
11		H-Muhtelif İmalatları		78	19.06.2024	06.09.2024



ÖZGEÇMİŞ

Ümit BAHADIR, İlk ve orta öğrenimini Fatih İlköğretim Okulu'nda, lise öğrenimini ise Trabzon Kanuni Anadolu Lisesi'nde tamamladı. 2009 yılında Karadeniz Teknik Üniversitesi, Mühendislik Fakültesi, İnşaat Mühendisliği Bölümünü kazandı. 2014 yılında İnşaat Mühendisliği Bölümündeki lisans eğitimini onur öğrencisi olarak tamamladı. Aynı yıl Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İnşaat Mühendisliği Anabilim Dalında yüksek lisans eğitimine başladı ve 2018 yılında başarı ile tamamladı. 2018 yılında aynı birimde doktora öğrenimine başladı. 2015 yılında Karadeniz Teknik Üniversitesi İnşaat Mühendisliği Bölümü Yapım Yönetimi Anabilim Dalına Araştırma Görevlisi olarak atandı. 2018-2019 Güz Dönemi, bilimsel araştırmalar yapmak üzere Orta Doğu Teknik Üniversitesi'nde bulunan BAHADIR, yapı bilgi modellemesi, sürdürülebilir yapı tasarımı, enerji verimliliği ve yapay zekâ uygulamaları ve benzeri alanlarda çalışmalarını sürdürmektedir. Evli ve bir kız çocuk babası olan BAHADIR, İngilizce bilmekte olup halen Karadeniz Teknik Üniversitesi'ndeki görevine devam etmektedir.