**T.C.**

**TOKAT GAZİOSMANPAŞA ÜNİVERSİTESİ**

**LİSANSÜSTÜ FEN ENSTİTÜSÜ**

**MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**MEKATRONİK MÜHENDİSLİĞİ YÜKSEK LİSANS**

# Uç Bilişim ile LSTM Tabanlı Arıza Tespiti

**YÜKSEK LİSANS TEZİ**

**Mert KIŞLAKÇI**

**Tez Danışmanı: Dr. Öğr. Üyesi Mahmut DURGUN**

**TOKAT- 2023**

## ETİK SÖZLEŞME

Tokat Gaziosmanpaşa Üniversitesi Lisansüstü Fen Enstitüsü tez yazım kılavuzuna göre Dr. Öğr. Üyesi Mahmut DURGUN danışmanlığında hazırlamış olduğum "Uç Bilişim ile LSTM Tabanlı Arıza Tespiti" adlı Yüksek Lisans tezinin bilimsel etik değerlere ve kurallara uygun, özgün bir çalışma olduğunu, aksinin tespit edilmesi halinde her türlü yasal yaptırımı kabul edeceğimi beyan ederim.

15 / 10 / 2023

Mert KIŞLAKÇI

# TEŞEKKÜR

Tez çalışmamın tamamlanmasında ve bu çalışmanın başarılı bir şekilde sunulmasında birçok kişiye ve kuruma minnettarım.

İlk olarak, danışmanım Dr. Öğr. Üyesi Mahmut DURGUN, akademik rehberliği ve değerli önerileri için teşekkür ederim. Bu çalışma, onun deneyim ve bilgisi olmadan mümkün olmazdı. Aileme, sürekli destekleri ve cesaretleri için teşekkür ederim. Onlar olmadan bu tez çalışması tamamlanamazdı. Ayrıca, laboratuvar imkânı sunan ve çalışma arkadaşlarıma, bu projede gösterdikleri iş birliği ve yardımları için minnettarım. Sanallaştırma Operasyonları Müdürü Bünyamin TOPAN'a çalışmanın laboratuvar deneyleri ve veri analizleri üzerindeki katkıları için özellikle teşekkür ederim. Son olarak, bu çalışma sürecinde manevi desteğini hiç esirgemeyen arkadaşım Metehan ŞENGÜL'e teşekkür ederim.

# ÖZET


UÇ BİLİŞİM İLE LSTM TABANLI ARIZA TESPİTİ

Kışlakçı, Mert

Yüksek Lisans, Mekatronik Mühendisliği Ana Bilim Dalı

Tez Danışmanı: Dr. Öğr. Üyesi Mahmut Durgun

Ekim 2023, xiii + 70 sayfa

Endüstri 4.0 çerçevesinde hayatımıza giren Nesnelerin İnterneti (IoT), birçok sistem üzerinde iletişim ve kontrol imkânı sağlayarak önemli bir rol oynamaktadır. 2020 yılında yaklaşık olarak 20,4 milyar IoT cihazının var olduğu tahmin edilmektedir. Ancak, IoT cihazlarının çeşitliliği ve yaygınlığı, bazı sorunları da beraberinde getirmektedir.

Birinci sorun, IoT cihazlarından üretilen verilerin iletilmesi ve işlenmesinin zorluğudur. Günümüzde, genellikle bulut bilişim tabanlı çözümler tercih edilmekte ve tasarlanan IoT sistem mimarilerinde bulut bilişim yaklaşımı kullanılmaktadır. Ancak, her sistem için ayrı bir bulut kaynağı tahsis etmek, gerçek zamanlı IoT sistemlerinde gecikme sorunlarına ve maliyetlere yol açabilmektedir. Bu dezavantajı ortadan kaldırmak için, uç (sınır) bilişim yaklaşımıyla tasarlanan sistemler daha verimli olabilir. Bu durumda, veriler yerel noktalarda toplanıp işlenirken buluttaki işlem yükü azaltılarak gecikme süresi düşürülür ve kaynak tasarrufu sağlanır.

Diğer bir zorluk ise IoT sistemlerindeki hataların tespitidir. Sorunun zamanında tespit edilmemesi, yedek parça stokunun olmaması ve değiştirilecek parçaların tedarik süresi gibi faktörler, sistem sürekliliğini engelleyebilmektedir. Yapılan literatür araştırmaları, yapay zeka modellerinin IoT cihazlarından elde edilen verilerle bulut katmanında sürekli olarak eğitildiğini ve arıza tespiti için kullanıldığını göstermektedir. LSTM (Uzun Kısa Süreli Bellek) gibi yapay zeka modelleri, dizi tahmin problemlerinde sıra bağımlılığını öğrenebilen bir RNN (Tekrarlayan Sinir Ağı) mimarisine sahiptir. LSTM, zaman serilerine dayalı veriler üzerinde tahminler yapmak için oldukça başarılı bir yapay zeka modelidir. IoT verileri de genellikle zaman serilerine bağlı olduğundan, LSTM bu alanda diğer modellere göre avantajlıdır. Ancak, bu tür bir yapay zeka modelinin bulut katmanında çalıştırılması, depolama kapasitesi, bilgi işlem yükü, ağ tıkanıklığı ve sunucu maliyetleri gibi sorunları beraberinde getirebilmektedir.

Bu çalışma, Nesnelerin İnterneti (IoT) cihazlarının verilerinin işlenmesi ve hata tespiti konusunda bulut bilişim ile uç bilgi işlem altyapısını birleştiren bir yaklaşım olan Uç Bilişim ile LSTM Tabanlı Arıza Tespiti (UBLTAT) sistemini tanıtmaktadır. UBLTAT, IoT cihazlarından elde edilen verileri bulut katmanında eğitim için kullanılan LSTM modeline iletmektedir. İstenilen eğitim başarısı sağlandıktan sonra, modelin ağırlıkları sınır katmanında çalışan UBLTAT'a aktarılmakta ve hata tespiti uç katmanında LSTM modeli kullanılarak gerçekleştirilmektedir. Bu şekilde, yapay zeka modellerinin UBLTAT ile uçta çalıştırılması, cihazların ürettiği verilerin yerel noktalarda işlenmesini ve hata tespitini sağlarken aynı zamanda buluta bağlı maliyeti de azaltmaktadır.

**Anahtar Kelimeler:** Bulut Bilişim, Sınır Bilişim, Nesnelerin İnterneti, IoT, Yapay Zekâ, LSTM, Arıza Tespiti, Endüstri 4.0

**ABSTRACT**

LSTM BASED FAULT DETECTİON WİTH EDGE COMPUTİNG

Kışlakçı, Mert

Master's Thesis, Department of Mechatronics Engineering

Thesis Advisor: Assit.Prof.Dr. Mahmut Durgun

October 2023, xiii + 70 pages

The Internet of Things (IoT), which has emerged under the framework of Industry 4.0, plays a significant role by providing communication and control capabilities in various systems. It is estimated that there were approximately 20.4 billion IoT devices in 2020. However, the diversity and ubiquity of IoT devices also bring forth certain challenges.

The first challenge is the difficulty in transmitting and processing data generated by IoT devices. Currently, cloud-based solutions are commonly preferred, and cloud computing approaches are employed in the design of IoT system architectures. However, allocating a separate cloud resource for each system can lead to latency issues and increased costs in real-time IoT systems. To overcome this disadvantage, designing systems based on edge computing approach can be more efficient. In this case, data is collected and processed at local points, reducing the computational load on the cloud, decreasing latency, and enabling resource savings.

Another challenge is the detection of errors in IoT systems. Failure to detect problems in a timely manner, lack of spare parts inventory, and long lead times for replacement parts can disrupt system continuity. Literature studies indicate that artificial intelligence models are continuously trained with data obtained from IoT devices in the cloud layer and used for fault detection. LSTM (Long Short-Term Memory), a type of artificial intelligence model, has a recurrent neural network (RNN) architecture that can learn sequential dependencies in time series data. LSTM is a highly successful model for making predictions based on time series data. Considering that IoT data is often time-dependent, LSTM has an advantage over other models in this field. However, running such an artificial intelligence model in the cloud layer can pose challenges such as storage capacity, computational load, network congestion, and server costs.

This study introduces the Edge Computing with LSTM-Based Fault Detection (UBLTAT) system, which combines cloud computing and edge computing infrastructure for processing IoT device data and detecting faults. UBLTAT transfers the data obtained from IoT devices to the cloud layer to train the LSTM model. Once the desired training accuracy is achieved, the weights of the model are transferred to UBLTAT, which operates at the edge layer, and fault detection is performed using the LSTM model in the edge layer. By running artificial intelligence models with UBLTAT at the edge, data generated by devices can be processed and faults can be detected locally, while also reducing the costs associated with the cloud.

**Keywords:** Cloud Computing, Edge Computing, Internet of Things, IoT, Artificial Intelligence, LSTM, Fault Detection, Industry 4.0

# İÇİNDEKİLER

# TABLOLAR LİSTESİ

## ŞEKİLLER LİSTESİ

## KISALTMALAR VE SİMGELER LİSTESİ

IOT: Nesnelerin İnterneti

LSTM: Uzun Kısa Süreli Bellek

UBLTAT: LSTM Tabanlı Arıza Tespiti

AI: Yapay Zeka

CNN: Evrişimli Sinir Ağı

RNN: Rekurrent Sinir Ağları

PLC: Programlanabilir Mantık Kontrolcüsü

CNC: Bilgisayar Sayı Kontrolü

RUL: Kalan Faydalı Ömür

YZ: Yapay Zeka

GRU: Kapılı Tekrarlama Ünitesi

SVM: Destek Vektör Makineleri

SOM: Öz Düzenleyici Haritalama

MQTT: Message Queuing Telemetry Transport

Wi-Fi: Wireless Fidelity

MLOPS: Makine Öğrenimi İşlemleri

# 1. GİRİŞ

Nesnelerin İnterneti (IoT), Derin Sinir Ağları, Yapay Zeka (AI), Bulut ve Sınır Bilişim, Endüstri 4.0'ın önemli oyuncuları arasında yer almaktadır (Verma vd., 2021). IoT, farklı cihazların internet üzerinden bağlanarak iletişim kurabildiği bir ağdır (Deorankar ve Thakare, 2020). Son yıllarda, IoT alanında hızlı bir gelişme yaşanmış ve birçok sektör, bu teknolojiyi sorunlarını çözmek için kullanmaya başlamıştır. IoT, sistemlere bağlı olarak çevresel bilgilerin izlenmesini ve uzaktan kontrol edilmesini kolaylaştırır (Ou vd., 2020). Tahminlere göre, 2023 yılına kadar 14.7 milyar IoT cihazı, kendi verilerini üretecektir. Bu veriler, sensör ağları aracılığıyla merkezi bir sistem, örneğin uç veya bulut sunucusuna iletilir (Teh vd., 2021). Bulut sunucuları, büyük veri depolama için güçlü bir altyapı sağlar. Aynı zamanda pahalı bilgisayar donanımı ve yazılımının bakım gereksinimlerini ortadan kaldırır ve büyük veri kümelerini depolar (Kars, 2021).

Geleneksel bulut tabanlı Nesnelerin İnterneti (IoT) mimarisi tasarımı, tüm verilerin buluta gönderilmesiyle beraber maliyet artışına, güvenlik açıklarına ve gerçek zamanlı uygulamalarda öngörülemeyen gecikmelere neden olur (Utomo ve Hsiung, 2019). Ancak bu yaklaşım, IoT geliştirme maliyetlerini, sistem karmaşıklığını ve yeni nesil IoT tasarımlarının sunabileceği rekabet avantajlarını azaltır (Oyekanlu, 2018). Bu nedenle, bulut bilişime bağlı dezavantajları gidermek için "uç bilişim" kavramı ortaya çıkmıştır. Uç bilişim, IoT verilerinin bulut bilişim ağının uç noktasında hesaplanmasına izin veren bir ara katman teknolojisidir. Uç bilişimde, veriler cihazın kendisi veya yerel bir bilgisayar veya sunucu tarafından işlenir ve veri merkezine gönderilmek yerine yerinde işlenir. Bu özellikleri sayesinde, Sınır (uç) bilişim, IoT sistemleri için avantajlı bir teknolojidir (Lee vd., 2019). Uç bilişim, maliyet açısından uygun performans odaklı çözümler sunarken aynı zamanda endüstriyel ölçekteki büyük verilere entegre edilerek veri analizi tabanlı gerçek zamanlı, esnek ve hızlı karar verme yetenekleri sağlar (Liu vd., 2020). Endüstriyel IoT'deki (IIoT) uç cihaz arızaları, endüstriyel ürünlerin üretimini ciddi şekilde etkileyebileceğinden doğru ve zamanında arıza tespiti giderek daha önemli hale gelmektedir (Liu vd., 2021).

Makinenin içerisindeki önemli bileşenlerde meydana gelen küçük bir arıza, tüm makinede güvenlik tehditlerine ve sonrasında bir güvenlik kazasına neden olabilir. Bu nedenle, makinelerin temel bileşenlerinin çalışma durumunun gerçek zamanlı olarak izlenmesi ve teşhis edilmesi büyük önem taşır (Lu vd., 2021). Özellikle üretimin büyüklüğü ve karmaşıklığı hızla

arttıkça, ekipman veya ürün arızası olasılığı da artmaktadır. Arızaların maliyetini en aza indirmek için üretim hattındaki mekanik cihazların arızalı davranışları mümkün olduğunca erken tespit edilmeli ve mevcut arızalar erken aşamalarda saptanmalıdır (Hsieh vd., 2019). IoT sistemlerinde arızaları tahmin etmek için güvenilir ve etkili bir modelin kullanılması bu zorlukların üstesinden gelmek için gereklidir. Arızalardan önce ortaya çıkabilecek erken anormallikler tespit edilebilir ve periyodik bakım aralıklarıyla önleyici bakım yapılarak ani arızaların önüne geçilebilir. Bu şekilde, bazı IoT cihazlarının kullanım ömrü uzatılabilir ve operasyonel maliyetlerden tasarruf sağlanabilir (Kayode & Tosun, 2019).

## 2. LİTERATÜR ÖZETLERİ/KURAMSAL TEMELLER/GENEL BİLGİLER

Lu ve arkadaşları (2021), endüstriyel bir makineden topladıkları veri setini kullanarak hibrit bir model olan Uzun Kısa Süreli Bellek-Varyasyonel Otomatik Kodlayıcıya (LSTM-VAE) dayalı yeni bir bulut-uç bileşenli akıllı arıza teşhis yöntemi önermişlerdir. Bu çalışmada, makineden gelen titreşim sinyalleri zaman serisine bağlıdır ve örnekler arasında zamansal bir korelasyon bulunmaktadır. Önerilen yöntem, ilk olarak makinelerin algılama verilerini uç katmanından depolama ve analiz için buluta aktarmaktadır. Ardından, LSTM-VAE hibrit arıza teşhis modeli bulutta eğitilmektedir. Yöntemin genel mimarisi çevrim dışı eğitim ve çevrim içi test olmak üzere iki aşamadan oluşmaktadır. Çevrim dışı eğitim aşaması, öncelikle LSTM-VAE hibrit modelini eğitmek ve uyarı eşiğini hesaplamak için bulutta gerçekleştirilmektedir. Eğitimli LSTM-VAE modeli ve uyarı eşiği daha sonra uç cihazlara aktarılmaktadır. Çevrim içi test aşaması ise uçta gerçekleştirilmekte olup temel görevi, izlenen gerçek zamanlı titreşim sinyallerini eğitimli LSTM-VAE hibrit modelini kullanarak arızalı titreşimleri hesaplamaktır. Bu çalışma sonucunda, makine arızalarının tespiti için bulut-uç birlikteliği yaklaşımına dayalı olarak LSTM-VAE hibrit modeli önerilmiş ve yöntemin tutarlılığı deneysel olarak kanıtlanmıştır. Bu yöntemin kullanılması, makinelerin kontrol edilebilir aralıkta güvenli bir şekilde üretilebilmesi için, makinenin ana bileşenlerinin çalışırken anormal durumlarını zamanında ve etkin bir şekilde tespit edebilme yeteneğini sağlamaktadır. Ayrıca, sistemin güvenli ve istikrarlı çalışmasını sağlamak ve işletmenin üretim verimliliğini artırmak açısından büyük önem taşımaktadır. Bununla birlikte, endüstriyel arıza senaryolarında hala çok değişkenli mekanik arıza senaryolarının bulunduğu unutulmamalıdır.

Liu ve arkadaşları (2021) endüstriyel üretimde arızaların ürünlerin üretimini ciddi şekilde etkilediğini ve bu arızaların doğru ve zamanında tespit edilmesinin giderek daha önemli hale

geldiğini savunmaktadır. Bu bağlamda, önerdikleri yöntem beş adımdan oluşmaktadır. İlk adımda, uç cihazlar IIoT düğümlerinden toplanan zaman serisine bağlı verileri yerel bir veri seti olarak kullanır. Uç cihazlar, yerel veri seti üzerinde AMCNN-LSTM modelinin eğitimini gerçekleştirir. Bu model, yerel cihazın güçlü bilgi işlem kaynaklarına sahip olması nedeniyle uç cihazda eğitilir. Eğitilen modelin güncellemeleri, uç cihazlar tarafından bulut sunucusuna iletilir. Bu güncellemeler, zaman serisi verilerindeki anormallikleri yakalamak için modelin performansını artırmak amacıyla kullanılır. Bulut sunucusu, uç cihazlardan gelen model güncellemelerini toplar ve yeni bir küresel model oluşturur. Bu adım, birleştirilmiş öğrenme algoritması veya diğer toplama algoritmaları kullanılarak gerçekleştirilebilir. Oluşturulan yeni küresel model, bulut sunucusu tarafından her bir uç cihaza iletilir. Bu sayede uç cihazlar, doğru ve zamanında anormallik tespiti için güncel ve kullanılabilir bir modeli çalıştırabilir.

Liu ve arkadaşlarının (2021) çalışmasının sonuçlarına göre, uç cihazlar yerel veri kümelerinde bölgesel olarak paylaşılan bir küresel modeli eğitebilmektedir. Bu çalışmada, IIoT düğümlerinden elde edilen zaman serisi verileri uç cihazlar tarafından algılanmakta ve model güncellemeleri güçlü bilgi işlem gücüne ve zengin bilgi işlem kaynaklarına sahip bir bulut sunucusuna iletilmektedir. Bu yaklaşım, bulut sistemlerine bağlı ağ trafiğini azaltarak maliyetleri düşürmekte ve modellerin uç noktalara dağıtılmasıyla buluta yük bindiren işlem yükünü hafifletmektedir. Bulut sunucusu, model güncellemelerini toplamak için birleştirilmiş öğrenme algoritması veya diğer toplama algoritmalarını kullanmaktadır. Bu sayede, uç cihazlardan gelen güncellemelerle yeni bir küresel model oluşturulmaktadır. Son adımda, uç cihazlar bulut sunucusu tarafından gönderilen yeni küresel modeli çalıştırarak doğru ve zamanında anormallik tespiti yapabilmektedir. Bu çalışmanın savunusu, bulut sistemlerine bağlı ağ trafiğini azaltmak ve verimli bir sistem elde etmek için modellerin uç cihazlara dağıtılmasının önemini vurgulamaktadır. Ayrıca, uç cihazlar tarafından toplanan verilerin güçlü bir bulut sunucusunda analiz edilmesi ve güncellemelerin bu sunucu aracılığıyla iletilmesi, anormallik tespiti için kullanılabilir bir sistem sağlamaktadır (Liu vd., 2021).

Chin ve arkadaşlarının çalışmasına göre, önerilen sistemde araçlar birbirine bağlanarak büyük veri akışları elde edilmekte ve bulutta işlenerek gerçek zamanlı arıza tespiti sağlanmaktadır. Bu sistemin temelinde, arıza tespiti için araç içinde üretilen verilere dayalı bir model kullanılmaktadır ve model eğitimi bulut katmanında gerçekleştirilmektedir. Modelin oluşturulmasında CNN'ler ve LSTM'lerin kombinasyonu kullanılmaktadır. İlk adımda, IoT uç modeli eğitilmektedir. Modelin dağıtımını otomatikleştirmek için bir veri toplayıcı, IoT uç

modülü ile elektronik kontrol ünitesi verileriyle simüle edilir. Giriş verileri puanlanır ve eğitim verileri kullanılarak tahmin modeli oluşturulur. Bu model daha sonra bir konteyner kapsayıcısına dönüştürülerek bulut katmanından uç katmana iletilir. Son adımda, model uç katmanda çalıştırılır ve ön ateşlemeler için tahmin sonuçları cihazdan toplanır. Bu tahmin sonuçları gerçek sonuçlarla karşılaştırılır ve başarısı ölçeklendirilir. Test sonuçlarına göre, modelin başarısı %80 ila %88 arasında değişmektedir. Çalışmanın uygulanabilirliği, donanım ortamı ve gerçek zamanlı gereksinimler gibi faktörlere bağlı olarak kullanılacak modelin seçilmesi gerekmektedir. Bu sistem, araçlardaki arızaların erken tespiti için etkili bir yaklaşım sunmaktadır. (Chin vd., 2019).

Huang ve arkadaşlarının çalışmasına göre, makinenin kalan faydalı ömrünü gerçek zamanlı olarak tahmin etmek için bir Uzun Kısa Süreli Bellek (LSTM) tekrarlayan sinir ağı modeli önerilmektedir. Bu model, bir uç veri işleme yöntemiyle birleştirilerek, veri temizleme ve özellik çıkarma işlemlerinin uç düğümde gerçekleştirilmesini amaçlamaktadır. Bu yaklaşımın temel amacı, iletim süresini azaltmak, iletim maliyetinden tasarruf etmek ve ömür tahmininin gerçek zamanlı performansını iyileştirmektir. Çalışmanın ilk aşamasında, veriler doğrudan veri kaynağına en yakın yerde uç bilgi işlem kullanılarak işlenir ve öznitelikler çıkarılır. Bu yöntem, aracın çalışma durumunu gerçek zamanlı olarak izlemek için iletim maliyetini ve süresini azaltmaya yardımcı olur. Veri madenciliği işlemi ise bulutta gerçekleştirilir. Zamana bağlı toplamanın istatistiksel özellikleri çıkarılır ve makine öğrenme algoritmaları kullanılarak özellikler taranır. Ardından, kalan faydalı ömrü tahmin etmek için bir LSTM modeli oluşturulur. Bu yaklaşımda, bulut hizmetleri kullanıcıların daha düşük maliyetle yüksek kaliteli hizmetler elde etmelerini ve bilgi işlem kaynaklarını esnek bir şekilde kontrol etmelerini sağlar. Ancak, bulut bilişimin sınırlamaları vardır ve büyük miktarda verinin tek bir bulut merkezine aktarılması ve orada işlenmesi durumunda darboğazlar oluşabilir. Huang ve arkadaşları tarafından önerilen yöntem, uç düğümlerde veri işleme ve özellik çıkarma yaparak bu darboğazları azaltmayı hedeflemektedir. Bu sayede, gerçek zamanlı olarak makinenin kalan faydalı ömrünü tahmin etmek mümkün hale gelir. Tamamlayıcı bir yöntem olarak, uç veri işleme mantıklı bir yaklaşım sunmaktadır. Bu yaklaşım, verilerin tamamının bulutta işlenmesi durumunda ortaya çıkabilecek veri gecikmelerini ve gerçek zamanlı performansın etkilenmesini engellemeye yardımcı olur. Özellikle Endüstriyel Nesnelerin İnterneti (IIoT) ile birlikte, araçlara bağlı sensörlerin sayısı ve toplanan veri miktarı artmaktadır. Uçta veri işleme, makinenin ürettiği sinyallerden veri özelliklerini elde etmek için kullanılır. Bu, veri iletimini

büyük ölçüde azaltabilir ve üretim kalitesini iyileştirerek tasarruf sağlayabilir. Daha sonra bulutta daha fazla veri analizi yapılabilir, önemli özellikler tespit edilebilir ve aracın çalışma durumunu gerçek zamanlı olarak izlemek için LSTM modeli kurulabilir. Uzun Kısa Süreli Bellek (LSTM), geleneksel Rekürrent Sinir Ağı (RNN) yapısından daha doğru bir zaman serisi analizi sağlayarak uzun vadeli bağımlılıkları modellemeyi hedefler. Veri ön işleme ve özellik çıkarma işlemleri doğrudan uç düğümde gerçekleştirildiğinde, aktarım maliyetinden tasarruf sağlanır ve makine takımının ömür tahmininde gerçek zamanlı performans iyileştirilir. Sensör ekipmanının arızalı olduğu durumlar veya iletim sinyallerinin kararsız olduğu durumlarda, orijinal veri setinde bireysel karakteristik anormallikler ortaya çıkabilir. Bu durumlar ele alınarak, uçta gerçekleştirilen veri işleme ve analiz yöntemleri ile bu anormallikler tespit edilebilir ve önleyici önlemler alınabilir. Huang ve diğerleri (2020) tarafından yapılan deneyler, sistemde kullanılan üç eksenli titreşim sensörü ve PLC denetleyicisi aracılığıyla CNC takım tezgahının iş mili yükü, üç eksenli mekanik koordinatlar, üç eksenli titreşim sinyali ve ilk faz akımı sinyali gibi verilerin toplandığını göstermektedir. Veri toplama süreci, yeni bir takım işleme programı çalıştırıldığında başlar ve takımın ömrü sona erdiğinde sonlanır. Bu süre boyunca veriler sensör aracılığıyla uç düğüme iletilir ve uç düğümde doğrudan işlenir. Veri ön işleme adımlarıyla başlayan işlem, verilerin temizlenmesini, sinyallerin filtrelenmesini ve anormal değerlerin algılanmasını içerir. Örneğin, titreşim sinyalinin kayan penceresindeki anormal değerler ortalama değerle doldurulur. Verilerin saklanması için etkin veri iki baytta bir kaydedilir ve toplama süresi her 5 dakikada bir dakikalık veri toplamaktadır. Toplanan veriler makinenin dört çeşit sinyalini içerir ve makinenin 1'in orijinal veri boyutu 4500 MB'dir. Ancak, çıkarılan özelliklerin toplam boyutu 60 GB'dır ve hesaplamalı bellek boyutu yaklaşık 87 MB'dir. Bu şekilde, iletim verileri %98 oranında azalmıştır. Uç veri işleme yöntemiyle veri aktarımının boyutunun büyük ölçüde azaldığı ve aktarım maliyetinin düştüğü gözlemlenmiştir. Ayrıca, araç durumundaki değişikliklere yanıt süresinin kısaltılabileceği belirtilmektedir. Bu deneylerin sonuçları, makinenin kalan faydalı ömrünü tahmin etme yeteneğinin iyileştirilebileceğini göstermektedir. Ancak, takım ömrü tahmininin doğruluğu hala zorlu bir konudur. Gelecekte, doğru bir şekilde takım durumunu temsil edebilecek frekans alanı ve zaman-frekans alanındaki sinyal özelliklerinin kazanılması gerektiği belirtilmektedir (Huang vd., 2020). Bu gelişmeler, tahmin doğruluğunu artırmak ve makine takımlarının performansını daha iyi izlemek için önemli olacaktır.

Kayode ve Tosun (2019) tarafından yapılan araştırmaya göre, IoT cihazlarının beklenmedik arızaları ve ani arızaları, özellikle güvenlik açısından kritik sistemler için büyük bir tehdit oluşturabilir. Bu nedenle, durum izleme ve sağlık durumu tahmini gibi tekniklerin kullanılması, yüksek güvenilirlik ve kullanılabilirlik sağlamak açısından son derece önemlidir. Kalan Faydalı Ömür (RUL) tahmini, derin öğrenme algoritmalarıyla desteklenen veri tabanlı yaklaşımlar aracılığıyla gerçekleştirilebilir. Ancak, bu yaklaşımlar genellikle hesaplama açısından yoğun olup, kaynakları sınırlı olan cihazlarda uygulanması zordur. Bununla birlikte, araştırmacılar, büyük miktarda veri aktarımıyla ilişkili zaman gecikmesi ve ağ maliyetini en aza indirmek için uç bilgi işlem yaklaşımının benimsenmesinin önemli olduğunu savunmaktadır. Bu yaklaşım, işlem için merkezi bir bulut yerine verilerin uç cihazda işlenmesini sağlar. Bu şekilde, veri aktarımı miktarı azalır ve işlem maliyeti düşer. Uç bilgi işlem yaklaşımı, RUL tahmini gibi önemli görevler için daha uygun bir seçenek olabilir. Bu yöntem, kaynak kısıtlı cihazlarda kullanılabilirliği artırırken, zaman gecikmesi ve ağ maliyetini en aza indirerek daha etkili bir performans sağlar. Kayode ve Tosun'un (2019) çalışması, IoT cihazlarının güvenlik ve sağlamlığını artırmak için uç bilgi işlem yaklaşımının önemini vurgulamaktadır. Bu yöntem, RUL tahmini gibi önemli görevlerin daha etkili bir şekilde gerçekleştirilmesine olanak sağlar.

### 2.1.Nesnelerin İnterneti (IoT)

IoT kısaltmasıyla bildiğimiz "Nesnelerin İnterneti" (IoT), cihazlar arasında ve cihazlar ile bulut arasında iletişimi sağlayan bir ağ ve bağlantılı cihazlar ve teknolojilerin bütününü ifade eder (IoT nedir? - Yeni Başlayanlar İçin Nesnelerin İnterneti Kılavuzu - AWS, s.y.). Bu cihazlar, basit ev eşyalarından karmaşık endüstriyel araçlara kadar çeşitlilik gösterir. Uzmanlar, bugün 7 milyardan fazla bağlantılı IoT cihazı bulunmasına rağmen, bu sayının 2020'de 10 milyara ve 2025'te 22 milyara ulaşmasını beklemektedir (What Is the Internet of Things (IoT)?, s.y.).

IoT, internetin bilgi işlem alanında uzun süredir sağladığı verimlilik avantajlarını üretim süreçlerine ve dağıtım sistemlerine taşıyabilir. Dünya genelinde milyarlarca yerleşik internete bağlı sensör, şirketlere operasyonel güvenliği artırma, varlıkları izleme ve manuel süreçleri azaltma gibi imkanlar sunan son derece zengin bir veri seti sağlar. Makinelerden elde edilen veriler, ekipman arızalarını tahmin etmek ve üreticilere uzun süreli kesinti sürelerini önlemek için erken uyarı sağlamak için kullanılabilir. Araştırmacılar, müşteri tercihleri ve davranışları

hakkında veri toplamak için IoT cihazlarını kullanabilir, ancak bunun ciddi gizlilik ve güvenlik etkileri olabilir (What is IoT? The internet of things explained | Network World, s.y.).

Şekil 2. 1. Nesnelerin İnterneti (thinger.io)

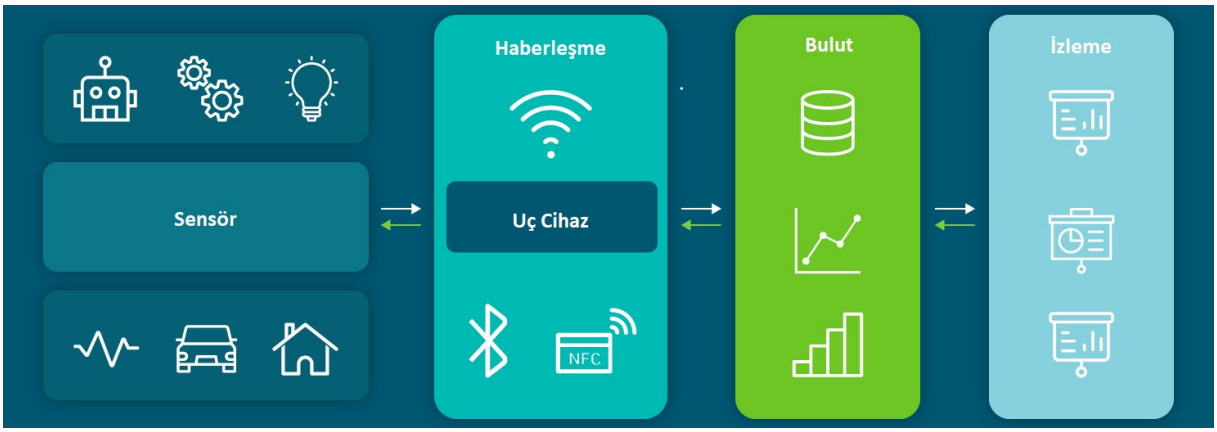
IoT, nesnelerin interneti olarak da bilinen bir teknoloji ve kavramdır. Temel olarak, fiziksel nesnelerin (cihazlar, sensörler, eşyalar vb.) internete bağlanarak veri toplaması, iletişim kurması ve etkileşimde bulunması prensibine dayanır. IoT sistemi genellikle aşağıdaki bileşenleri içerir:

a) Sensörler ve Cihazlar: IoT sisteminin temel unsurları, çevresel veri toplama ve algılama yeteneklerine sahip sensörler ve bu sensörlerle iletişim kurabilen cihazlardır. Sensörler, çeşitli parametreleri ölçebilir (sıcaklık, nem, basınç, ışık vb.) ve verileri dijital formatta toplayabilir.
b) Veri İletişimi: Sensörlerden elde edilen veriler, kablosuz veya kablolu iletişim protokolleri aracılığıyla iletişim ağına aktarılır. Bu ağlar genellikle Wi-Fi, Bluetooth, Zigbee gibi kablosuz iletişim protokolleri veya Ethernet gibi kablolu iletişim protokolleri kullanır.
c) Veri Toplama ve Aktarma: Toplanan veriler, IoT ağının bir parçası olan veri toplama noktalarına aktarılır. Bu noktalar, verileri depolama, filtreleme, birleştirme ve ön işleme yapma işlevlerine sahiptir.
d) Veri İşleme ve Analiz: Toplanan veriler, bulut bilişim platformları veya yerel sunucular gibi güçlü veri işleme kaynakları kullanılarak analiz edilebilir. Makine öğrenimi algoritmaları ve analitik teknikler, verilerden anlamlı bilgiler çıkarılmasına yardımcı olur.

e) Etkileşim ve Kontrol: IoT sistemi, kullanıcıların veya diğer sistemlerin IoT cihazlarıyla etkileşime girebilmesini sağlar. Bu etkileşim, uzaktan kontrol, komut gönderme veya veri alışverişi şeklinde olabilir.

### 2.1.1. IoT Nasıl Çalışır

Bir IoT sisteminin ilk bileşeni, veri toplama işlevine sahip cihazlardır. Bu cihazlar genellikle internete bağlıdır ve her biri bir IP adresiyle tanımlanır. Karmaşıklıkları, otonom mobil robotlar ve forkliftler gibi ürünleri fabrika katları ve depolar arasında taşıyan sistemlerden, binalardaki sıcaklık veya gaz sızıntısı gibi basit sensörlerle izlenen durumlara kadar değişir. Ayrıca, bireylerin günlük adım sayısını izleyen kişisel cihazlar gibi kullanıcı odaklı cihazlar da IoT sisteminin bir parçası olabilir. IoT sürecinin bir sonraki adımında, toplanan veriler cihazlardan bir toplama noktasına iletilir. Veri taşıma işlemi, kablosuz iletişim veya kablolu ağlar gibi çeşitli teknolojiler kullanılarak gerçekleştirilebilir. Toplanan veriler daha sonra İnternet üzerinden bir veri merkezine veya buluta iletilir. Alternatif olarak, ara cihazlar kullanılarak veriler birleştirilebilir, biçimlendirilebilir, filtrelenir ve önemli verilerin daha fazla analiz için iletilmesiyle artımlı bir aktarım sağlanabilir. Son adım olan veri işleme ve analiz aşaması, genellikle veri merkezlerinde veya bulutta gerçekleştirilir. Ancak, endüstriyel ortamlardaki kritik cihazlar gibi durumlarda, cihazdan uzak bir veri merkezine veri gönderme süresi oldukça uzun olabilir. Bu tür durumlarda, verilerin gönderilmesi, işlenmesi, analiz edilmesi ve talimatların geri dönmesi gibi süreçlerin çok uzun sürmesi, beklenmedik sorunlara neden olabilir. Örneğin, bir borunun patlamadan önce vanayı kapatmak gibi acil durumlarda hızlı kararlar almak önemlidir. Bu nedenle, IoT sisteminde akıllı uç cihazların kullanılması önemlidir. Akıllı uç cihazlar, veri toplama, analiz ve yanıt oluşturma gibi işlemleri yerel olarak gerçekleştirebilen cihazlardır. Bu sayede veri toplama ve işleme süreleri kısaltılabilir, gecikme azaltılabilir ve kritik kararlar hızlı bir şekilde alınabilir. Ayrıca, akıllı uç cihazlar, daha fazla işleme ve depolama kapasitesi için verileri yukarı yönlü bağlantılarla gönderebilir.




Şekil 2. 2. IoT Nasıl Çalışır (jelvix.com)

Bu tür durumlarda, akıllı uç cihazlar devreye girerek veri toplama, analiz ve yanıtların biçimlendirilmesi gibi işlevleri yerel olarak gerçekleştirebilir. Bu uç cihazlar, fiziksel olarak yakın bir konumda bulunarak gecikmeyi azaltır. Aynı zamanda, veri göndermek için yukarı yönlü bağlantılara sahip olarak daha fazla işleme ve depolama kapasitesine sahip merkezi sunuculara veri iletebilir. Gün geçtikçe, hızlı karar verilmesi gereken durumlar gibi uygulamalar için akıllı uç cihazların kullanımı artmaktadır. Örneğin, otonom araçlar gibi uygulamalar, verileri hemen işleyip analiz edebilen uç teknolojilerin gelişimini hızlandırmaktadır, böylece bu araçlar buluta bağımlı olmadan verileri anında kullanabilirler. Bu şekilde, akıllı uç cihazların kullanımı, IoT sistemlerinin verimliliğini artırırken gecikme sürelerini azaltır ve daha hızlı tepki süreleri sağlar. Ayrıca, yüksek hızlı veri işleme ve analiz yetenekleri, uç cihazların daha karmaşık görevleri yerine getirebilmesini ve daha fazla özerklik sağlamasını mümkün kılar. Bu da IoT teknolojilerinin daha geniş bir kullanım alanına yayılmasına ve endüstrilerin verimliliklerini artırmasına yardımcı olur.

### 2.1.2. IoT Hangi Teknolojiyi Mümkün Kıldı

IoT, bir dizi teknolojinin birleşimiyle mümkün hale gelmiştir. İşte IoT'nin mümkün kıldığı bazı temel teknolojiler:

a) Sensörler: IoT'nin temel yapı taşlarından biri olan sensörler, çevresel değişkenleri algılayabilir ve ölçebilir. Düşük maliyetli ve güçlü sensör teknolojileri, farklı parametreleri ölçmek için kullanılabilir hale gelmiştir. Sıcaklık, nem, ışık, hareket gibi birçok farklı parametreyi izlemek için sensörler kullanılır.

b) Kablosuz İletişim: IoT cihazları arasında veri alışverişi yapabilmek için kablosuz iletişim teknolojileri kullanılır. Wi-Fi, Bluetooth, Zigbee, Z-Wave gibi protokoller,

cihazlar arasındaki iletişimi sağlar. Bu teknolojiler, cihazların birbirleriyle ve internete bağlanmasını mümkün kılar.

c) Ağ Protokolleri: IoT cihazlarının internete bağlanabilmesi için çeşitli ağ protokolleri kullanılır. TCP/IP, MQTT, CoAP gibi protokoller, sensörlerin verilerini iletmek için kullanılan standart iletişim protokolleridir.
d) Bulut Bilişim: IoT cihazlarından gelen verilerin depolanması, işlenmesi ve analiz edilmesi için bulut bilişim teknolojileri kullanılır. Bulut tabanlı platformlar, büyük veri miktarlarını işlemek ve kullanıcıların verilere erişimini sağlamak için ölçeklenebilir altyapı sunar.
e) Veri Analitiği ve Makine Öğrenimi: IoT'nin ürettiği büyük miktardaki veriyi analiz etmek ve içgörüler elde etmek için veri analitiği ve makine öğrenimi teknikleri kullanılır. Bu teknikler, verilerdeki desenleri ve trendleri keşfederek, verileri değerli bilgilere dönüştürür.
f) Yapay Zeka ve Doğal Dil İşleme: Konuşma tabanlı yapay zeka ve doğal dil işleme teknolojileri, IoT cihazlarının sesli komutlarla veya metin girişiyle etkileşim kurmasını sağlar. Dijital asistanlar gibi cihazlar, doğal dil anlama ve yanıt verme yeteneklerine sahiptir.

Bu teknolojilerin birleşimi, IoT'nin pratik hale gelmesini sağlamış ve farklı sektörlerde kullanılabilen birçok uygulamanın ortaya çıkmasına imkan vermiştir. IoT'nin gelişimiyle birlikte, bu teknolojiler de sürekli olarak yenilenmekte ve IoT ekosistemine daha fazla fayda sağlamak için geliştirilmektedir.

### 2.1.3. IoT Avantajları

Verimlilik: IoT, nesnelerin ve süreçlerin birbirleriyle iletişim kurabildiği ve verileri paylaşabildiği bir ekosistem sağlar. Bu sayede, iş süreçlerinde otomasyon ve verimlilik artışı sağlanır. Sensörler sayesinde gerçek zamanlı veri toplanır, analiz edilir ve kararlar hızla alınır.

İyileştirilmiş Karar Verme: IoT, geniş veri toplama ve analiz yetenekleri sayesinde daha iyi kararlar almayı mümkün kılar. Toplanan veriler, analitik yöntemler ve makine öğrenimi algoritmalarıyla işlenerek içgörüler elde edilir. Bu sayede daha iyi bilgilendirilmiş kararlar alınabilir ve daha etkili stratejiler geliştirilebilir.

Operasyonel Verimlilik: IoT, işletmelerin operasyonel süreçlerini optimize etmelerine yardımcı olur. Veri toplama ve analiz sayesinde enerji tüketimi, üretim verimliliği, envanter yönetimi

gibi alanlarda iyileştirmeler yapılabilir. Otomasyon, kaynakların daha etkin kullanılmasını sağlar ve işletmelerin maliyetleri düşürmesine yardımcı olur.

Müşteri Deneyimi: IoT, müşterilere daha iyi bir deneyim sunmayı sağlar. Örneğin, akıllı ev sistemleri müşterilere evlerini uzaktan kontrol etme ve enerji tasarrufu yapma imkanı sağlar. Perakende sektöründe IoT, müşterilere kişiselleştirilmiş teklifler sunabilir ve alışveriş deneyimini geliştirebilir.

Güvenlik ve Güvenilirlik: IoT, güvenlik önlemleriyle birlikte kullanıldığında, güvenlik ve güvenilirlik avantajları sağlar. Sensörler ve akıllı cihazlar, anormallikleri tespit edebilir ve güvenlik ihlallerine karşı uyarılar verebilir. Ayrıca, çevrimiçi olarak izleme ve takip edilen nesneler, kaybolma veya hırsızlık durumlarında yerlerini tespit etmeyi sağlar.

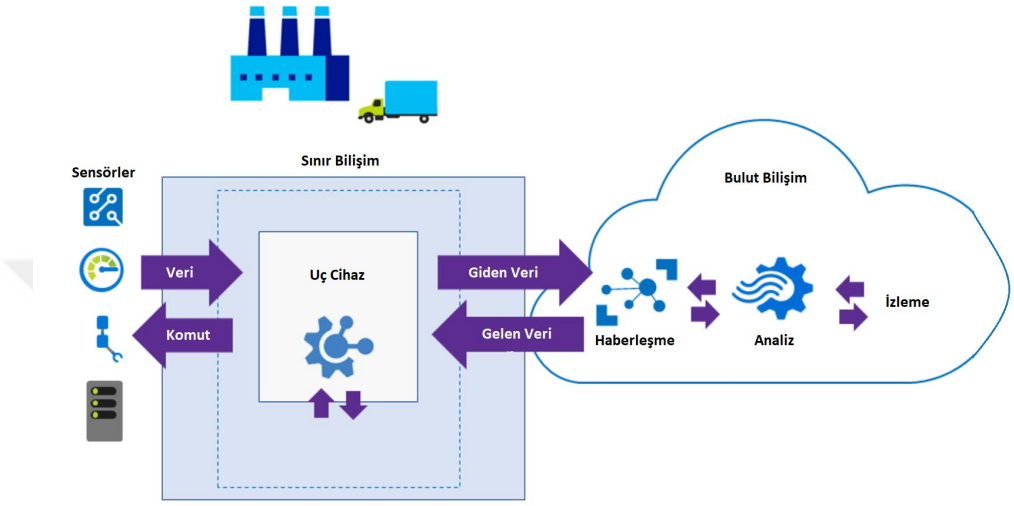
Sürdürülebilirlik: IoT, kaynakların daha sürdürülebilir bir şekilde kullanılmasına yardımcı olabilir. Örneğin, akıllı şebekeler enerji tüketimini optimize ederken, akıllı tarım sistemleri su ve gübre kullanımını azaltabilir. IoT aynı zamanda atık yönetimi ve çevresel izleme gibi alanlarda da sürdürülebilirlik sağlayabilir.

Sonuç olarak, IoT ağ geçitleri, IoT cihazlarının ve sensörlerinin merkezi olarak yönetilmesini sağlayan önemli bileşenlerdir. Veri yönetimi, güvenlik, protokol dönüştürme, yerel işleme ve bağlantı yönetimi gibi işlevleriyle IoT ağının etkin ve güvenli çalışmasını sağlarlar.

### 2.2.Uç Cihazlar

IoT'de "uç cihaz" terimi, IoT ağına bağlı olan ve veri toplama, iletişim ve yerel işleme yetenekleri olan cihazları ifade eder. Uç cihazlar, genellikle sensörler, kontrol cihazları, cihaz ağ geçitleri, akıllı ev cihazları, endüstriyel ekipmanlar gibi çeşitli şekillerde karşımıza çıkabilir. Uç cihazlar, IoT sisteminin temel bileşenleridir ve çevrelerindeki fiziksel dünyadan veri toplarlar. Örneğin, bir akıllı termostat evin sıcaklık verilerini algılar ve bu verileri IoT ağına iletebilir. Bir endüstriyel sensör, bir makineden alınan verileri toplayabilir ve bunları bir IoT ağı üzerinden yönlendirebilir. Uç cihazlar, genellikle düşük güç tüketimi, küçük boyutlar ve bağlantı yeteneklerine sahip olacak şekilde tasarlanırlar. Uç cihazlar, IoT'nin yaygınlaşmasına katkıda bulunan birkaç önemli avantaja sahiptir. Uç cihazlar, çevredeki verileri doğrudan toplar ve bu verileri hızla işleyebilir. Bu, gerçek zamanlı veri analizi ve hızlı geri bildirim sağlama imkânı verir. Örneğin, bir akıllı enerji izleme cihazı, anlık enerji tüketimi bilgilerini toplayarak kullanıcılara anında geri bildirimde bulunabilir. Uç cihazlar, hassas verilerin doğrudan cihazda

işlendiği ve saklandığı için veri güvenliği ve gizliliği sağlar. Verilerin yerel olarak işlenmesi, veri güvenliği risklerini azaltır ve güvenlik açıklarının hedeflenmesini zorlaştırır. Bulut tabanlı bir altyapıya sürekli olarak bağlı kalmak zorunda olmadıkları için bağımsız çalışabilirler. Bu, ağ bağlantılarının olmadığı durumlarda bile veri toplama ve yerel kararlar alma yeteneklerini korumalarını sağlar. Yerel olarak veri işleyerek sadece önemli verileri iletebilir. Bu, ağ trafiğini azaltır ve veri aktarımında bant genişliği tasarrufu sağlar. Sadece önemli verilerin iletilmesi,




Şekil 2. 3. Uç Cihaz Mimarisi

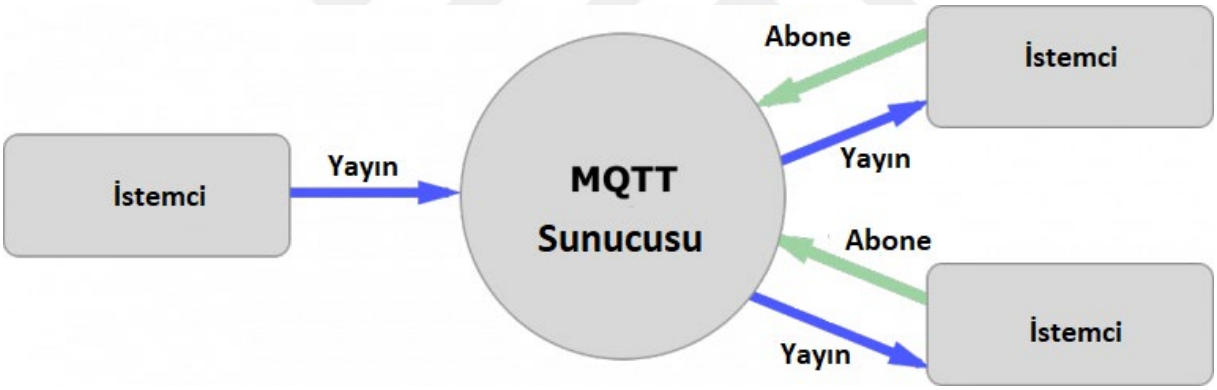
bulut tabanlı analitik ve depolama kaynaklarının daha etkin kullanılmasını sağlar. Kuruluşlar genellikle bir dizi farklı IoT cihazı dağıtır ve bu da bu cihazları izlemeyi ve yönetmeyi zorlaştırabilir. IoT ağ geçidi, IoT cihazlarını ve sensörlerini bulut tabanlı bilgi işlem ve veri işlemeye bağlayan merkezi bir merkezdir.

Uç cihazlar, IoT'nin temel bir parçasıdır çünkü IoT'nin temel amacı, birçok cihazın birbirleriyle etkileşimde bulunmasına ve iş birliği yapmasına olanak tanıyan bir ağ oluşturmaktır. Uç cihazlar, bu ağın temelini oluşturur ve verileri toplayarak ve aktararak, IoT'nin çeşitli avantajlarından yararlanılmasını sağlar. Modern IoT ağ geçitleri, genellikle bulut tabanlı hizmetlerle ve IoT cihazlarıyla iki yönlü veri akışına izin vererek önemli bir rol oynamaktadır. Bu çift yönlü veri akışı, IoT sistemlerinin etkin bir şekilde çalışmasını ve verimli bir şekilde yönetilmesini sağlamaktadır. İlk olarak, IoT sensörlerinden ve cihazlardan toplanan veriler, ağ geçidi aracılığıyla bulut tabanlı hizmetlere iletilir. Bu veriler, bulut tabanlı uygulamalar ve analiz platformları tarafından işlenir ve değerli içgörüler elde etmek için kullanılır. Örneğin, sensör verileri temelinde gerçek zamanlı analiz yapılabilir veya veriler gelecekteki tahminler veya optimizasyon stratejileri için kullanılabilir. Diğer taraftan, IoT cihazlarına komutlar ve

yönergeler de bulut tabanlı hizmetler aracılığıyla ağ geçidi üzerinden iletilir. Bu, IoT cihazlarının uzaktan kontrol edilebilmesini sağlar. Örneğin, bir uygulama veya kontrol paneli aracılığıyla belirli bir cihaza bir komut gönderilebilir veya cihazın çalışma parametreleri ayarlanabilir. Bu çift yönlü veri akışı, IoT sistemlerinin merkezi yönetimini ve kontrolünü kolaylaştırır. Ayrıca, verilerin bulut tabanlı analitik ve işlem platformlarına iletilerek daha geniş ölçekte analiz edilmesini sağlar. Bu da işletmelere veya kuruluşlara daha iyi kararlar alabilme ve operasyonlarını daha verimli hale getirme imkanı sunar. (*(1) Check Point*, y.y.).

### 2.3.MQTT

MQTT (Message Queuing Telemetry Transport), IoT cihazları arasında hafif, yaygın olarak kullanılan bir mesajlaşma protokolüdür. Bu protokol, düşük bant genişliği, yüksek düşük gecikme süreleri ve düşük güç tüketimi gibi özellikleriyle IoT uygulamaları için optimize edilmiştir. MQTT, yayıncı-abone modelini kullanır, bu da veri iletimini yayıncılar ve aboneler arasında gerçekleştirir. Yayıncılar, belirli bir konuya (topic) ilişkin verileri yayınlar, aboneler ise ilgilenilen konulara abone olarak bu verileri alır (OASIS Standard, 2014 ).

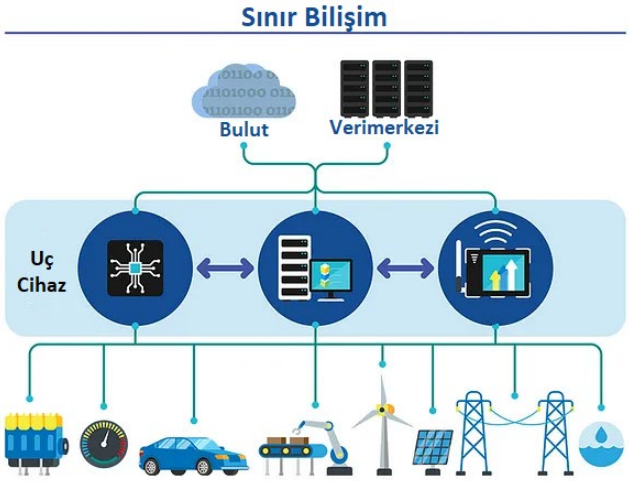



Şekil 2. 4. MQTT

Bu şekilde, IoT cihazları arasında veri paylaşımı ve iletişim kolaylaşır. Protokol, esneklik sağlamak için üç seviyeli bir kalite hizmeti (Quality of Service - QoS) sunar. Bu QoS seviyeleri, verinin iletiminde güvenilirlik, tekrar deneme ve teslimat garantisi gibi faktörleri kontrol etmeye yardımcı olur. MQTT, internete bağlı cihazlarla etkileşimde bulunmak için kullanılır ve yaygın olarak IoT uygulamalarında kullanılmaktadır. Örneğin, akıllı evler, enerji yönetimi, endüstriyel otomasyon ve sensör ağları gibi alanlarda sıkça kullanılan bir protokoldür (HiveMQ, 2019).

## 2.4.Sınır Bilişim

Sınır bilişim, IoT (Nesnelerin İnterneti) ve diğer benzeri uygulamalarda ortaya çıkan büyük veri ve hesaplama ihtiyaçlarına cevap verebilmek için kullanılan bir bilişim modelidir. Geleneksel bilişim altyapılarına kıyasla, sınır bilişim verilerin üretildiği veya tüketildiği yerlere, yani




Şekil 2. 5. Sınır Bilişim (belden.com)

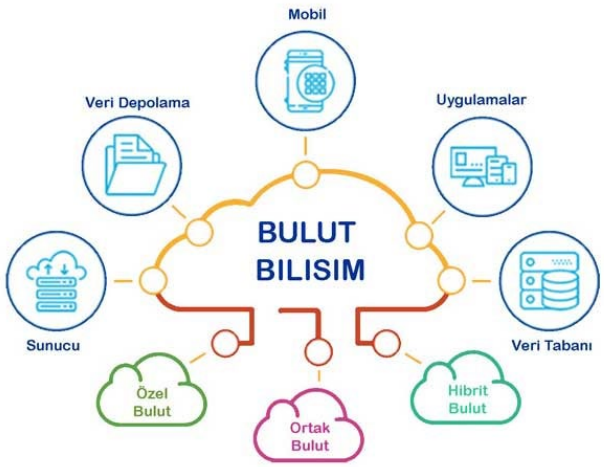
"sınır" olarak adlandırılan IoT cihazlarına daha yakın bir hesaplama yeteneği sunar. Sınır bilişimde, veri işleme ve depolama işlemleri IoT cihazlarına veya yerel ağa yerleştirilmiş sınır cihazlarına (edge device) taşınır (Davies, N, 2009).

Bu sayede, verilerin bulut tabanlı veri merkezlerine gönderilmesi ve işlenmesi için ağ trafiği azaltılır. Sınır cihazları, genellikle daha hızlı yanıt süreleri, düşük gecikme ve daha iyi bant genişliği kullanımı gibi avantajlar sağlar. Sınır bilişim, verilerin yerel olarak işlenmesi ve analiz edilmesi yoluyla gerçek zamanlı kararlar alınmasını ve tepki sürelerinin azaltılmasını sağlar. Ayrıca, sınırlı veya kesintili ağ bağlantılarına sahip ortamlarda da güvenilir bir işleme yeteneği sunar. Bu bilişim modeli, endüstriyel otomasyon, akıllı şehirler, akıllı evler, sağlık hizmetleri, taşımacılık ve diğer birçok IoT uygulamasında kullanılmaktadır (Xu L, 2016).

## 2.5.Bulut Bilişim

Bulut bilişim, bilgi işlem kaynaklarının (sunucular, depolama alanı, veritabanları, ağlar vb.) internet üzerinden paylaşılan ve isteğe bağlı olarak kullanılabilen bir hizmet olarak sunulduğu bir bilişim modelidir. Bu hizmetler, kullanıcılara ihtiyaç duydukları kaynaklara erişim sağlama, veri depolama, uygulama çalıştırma, veritabanı yönetimi ve daha fazlasını içerebilir. Bulut bilişim, kullanıcılara kendi donanım ve altyapılarını satın alma, kurulum ve yönetme

zorunluluğu olmadan, ihtiyaçlarına uygun şekilde bilişim kaynaklarına erişim imkanı sunar. Kullanıcılar, genellikle internet üzerinden erişebildikleri bir bulut hizmet sağlayıcısı aracılığıyla bu kaynaklara erişirler. Bulut bilişim, ölçeklenebilirlik, esneklik ve maliyet tasarrufu gibi bir dizi avantaj sunar. Kaynaklar isteğe bağlı olarak kullanılabilir ve talebe göre ölçeklenebilir, bu da kullanıcıların ihtiyaçlarına göre bilişim kaynaklarını kolayca ayarlayabilmesini sağlar (Mell, P, 2011) . Ayrıca, donanım altyapısını satın alma ve yönetme




Şekil 2. 6. Bulut Bilişim (lojistikbilimi.com)

maliyetlerinden kaçınarak maliyetleri düşürebilir. Bulut bilişim hizmet modelleri genellikle IaaS (Altyapı olarak Hizmet), PaaS (Platform olarak Hizmet) ve SaaS (Yazılım olarak Hizmet) olarak sınıflandırılır. IaaS, kullanıcılara sanal sunucu, depolama ve ağ gibi temel bilişim kaynaklarını sağlar. PaaS, uygulama geliştirme ve dağıtımı için geliştirme platformunu sağlar. SaaS, kullanıcılara uygulama yazılımlarını internet üzerinden kullanma imkanı sunar. Bulut bilişim, işletmelerin esneklik, ölçeklenebilirlik ve verimlilik avantajlarından yararlanmasına olanak tanır. Ayrıca, veri yedekleme ve güvenlik gibi konulara da odaklanır (Zaharia, 2010).

### 2.6.Bulut Bilişim ve Sınır Bilişimin Farkları

Bulut Bilişim (Cloud Computing) ve Sınır Bilişimi (Edge Computing), günümüz bilişim dünyasında önemli kavramlar haline gelmiştir. Her ikisi de farklı avantajlar ve kullanım senaryoları sunan bilişim modelleridir. Bulut bilişim, bilgisayar kaynaklarına ağ üzerinden erişim sağlayan bir bilişim modelidir. Bu modelde, uygulama, veri ve işlem gücü, uzak bir sunucu tarafından sağlanır ve kullanıcının cihazında yerel olarak saklanmaz. Bu, birçok avantaj sağlar, örneğin, kullanıcının cihazında yüksek miktarda işlem gücü gerektiren uygulamaları çalıştırmasına gerek kalmaz. Bunun yerine, işlem gücü ve depolama, uzak bir sunucuda

barındırılır ve ağ üzerinden kullanıcıya sağlanır (Satyanarayanan, 2017). Sınır bilişim, bilgisayar kaynaklarının yerel bir cihazda (örneğin, bir sensör) saklandığı ve işlendiği bir bilişim modelidir. Bu model, cihazların ve uygulamaların doğrudan ağa bağlanmasına olanak tanır ve böylece daha hızlı tepki süreleri ve daha az ağ trafiği sağlar. Sınır bilişim modelinde, cihazlar genellikle sensörler gibi düşük güç tüketen cihazlardır ve sınırlı işlem gücüne sahip olabilirler.

Bulut bilişim ve sınır bilişim arasındaki farklar;

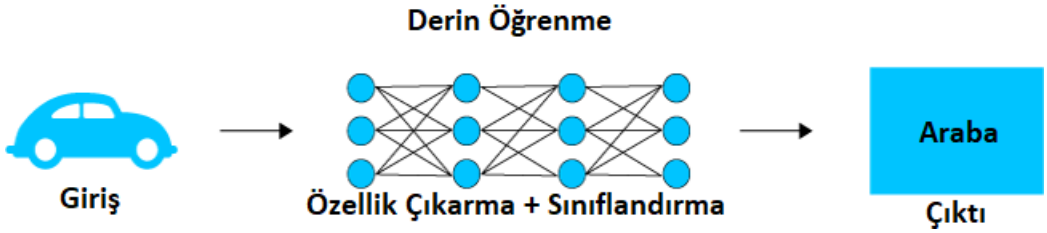
a. İşlem gücü: Bulut bilişim modelinde, işlem gücü ve depolama uzak bir sunucuda bulunurken, sınır bilişim modelinde işlem gücü yerel bir cihazda bulunur.
b. Veri saklama yeri: Bulut bilişim modelinde, veriler uzak sunucularda saklanırken, sınır bilişim modelinde veriler yerel cihazlarda saklanır.
c. Ağ trafiği: Sınır bilişim modeli, cihazların doğrudan ağa bağlanmasına olanak tanıdığından, daha az ağ trafiği gerektirirken, bulut bilişim modelinde ağ üzerinden daha fazla veri trafiği olabilir.
d. Tepki süresi: Sınır bilişim modelinde, cihazların doğrudan ağa bağlanması daha hızlı tepki süreleri sağlar. Bulut bilişim modelinde ise, uzak sunuculara erişimde gecikme olabilir. Bulut bilişim ve sınır bilişimi arasındaki farklar öncelikle veri işleme yerinin konumu, gecikme süresi, ağ bağımlılığı ve kaynak kullanımı açısından ortaya çıkar. Bulut bilişimi, geniş bir kullanıcı kitlesine genel olarak erişim sağlarken, sınır bilişimi daha özelleştirilmiş ve yerel olarak odaklanmış bir yaklaşıma sahiptir.

Bulut bilişim ve sınır bilişim, farklı işlem gücü ve veri saklama yöntemlerine dayalı iki farklı bilişim modelidir. Bulut bilişim, merkezi sunucular üzerinde yoğunlaşırken, sınır bilişim ise, yerel cihazlarda işlem gücü sağlayarak daha hızlı tepki süreleri ve düşük ağ trafiği sağlar. Özellikle gerçek zamanlı uygulamalarda kullanılır ve düşük gecikme süresi kritik öneme sahiptir. Ayrıca, veri güvenliği ve gizliliği açısından da avantaj sağlar, çünkü veriler yerel cihazlarda saklanır ve doğrudan ağa gönderilmez. Bulut bilişim ve sınır bilişim arasında seçim yaparken, uygulama gereksinimleri, veri miktarı, tepki süresi beklentileri ve güvenlik ihtiyaçları gibi faktörler dikkate alınmalıdır (Zaharia, M, 2010). Bazı durumlarda, bulut ve sınır bilişim modellerini bir arada kullanmak da mümkündür, böylece hem bulutun avantajlarından yararlanılırken hem de yerel cihazlarda hızlı işlemler gerçekleştirilebilir. Sonuç olarak, bulut bilişim ve sınır bilişim, farklı kullanım senaryolarına ve gereksinimlere göre tercih edilen

bilişim modelleridir. Her iki modelin de avantajları ve dezavantajları vardır ve doğru modelin seçilmesi, uygulamanın ihtiyaçlarına uygun olarak belirlenmelidir.

### 2.7.Derin Öğrenme

Derin öğrenme, yapay sinir ağları adı verilen matematiksel modeller kullanarak veri analizini gerçekleştiren bir yapay zeka yöntemidir. Bu yöntem, beynin sinir hücrelerinden ilham alarak,

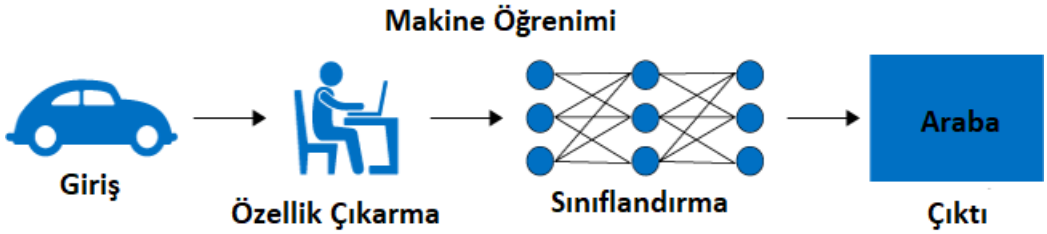



Şekil 2. 7. Derin Öğrenme (inonu.edu.tr)

çok katmanlı ve karmaşık yapılardan oluşan sinir ağları oluşturur. Bu ağlar, büyük miktardaki verileri analiz ederek desenleri tanımlayabilir, tahminlerde bulunabilir ve kararlar alabilir. Derin öğrenme, genellikle üç temel bileşen üzerine kuruludur: giriş katmanı, gizli katmanlar ve çıkış katmanı. Giriş katmanı, veriyi alır ve gizli katmanlara iletilir. Gizli katmanlar, gelişmiş algoritmalar ve matematiksel işlemlerle veriyi analiz eder ve çeşitli özelliklerini ortaya çıkarır (Hilton 2012). Son olarak, çıkış katmanı, elde edilen sonuçları sunar ve istenen hedefi gerçekleştirmek için kararlar alır. Derin öğrenme, örnek verilerle beslenerek kendini eğitir ve veriler arasındaki karmaşık ilişkileri öğrenir. Bu sayede, daha sonra benzer veri örneklerini doğru bir şekilde sınıflandırabilir veya tahmin edebilir.

### 2.8.Makine Öğrenmesi

Makine öğrenimi, bilgisayar sistemlerine verilen verileri analiz ederek desenleri ve ilişkileri tespit etmelerini, bu bilgileri kullanarak tahminler yapmalarını ve sonuçları öğrenme deneyimiyle birleştirerek kendilerini geliştirmelerini sağlayan bir disiplindir (Mitchell, T,




Şekil 2. 8. Makine Öğrenmesi (inonu.edu.tr)

1997). Bu süreçte, makine öğrenimi algoritmaları, istatistiksel yöntemler ve matematiksel modeller kullanarak verileri analiz eder, desenleri belirler ve sonuçları tahmin eder.

Öğrenme deneyimi sayesinde, sistemler daha doğru tahminler yapabilir ve performanslarını sürekli olarak iyileştirebilirler. Makine öğrenimi, birçok farklı uygulama alanında başarılı bir şekilde kullanılmaktadır. Örneğin, görüntü ve ses tanıma, doğal dil işleme, otomatik sürüş, tıbbi teşhis, pazarlama tahminleri, finansal analiz, güvenlik ve daha birçok alanda makine öğrenimi teknikleri kullanılmaktadır (Alpaydin, 2020). Ara tahminler yapar veya kararlar verir. Örneğin, görüntü ve ses tanıma uygulamalarında, makine öğrenimi algoritmaları, binlerce görüntü veya ses kaydını analiz ederek nesneleri veya konuşmayı tanıyabilir. Doğal dil işleme uygulamalarında, metin verilerini analiz ederek metinleri anlayabilir ve dil tabanlı görevleri gerçekleştirebilir. Otomatik sürüş alanında, makine öğrenimi teknikleri, sürüş davranışlarını analiz ederek araçların otomatik olarak sürülmesini sağlar (Hastie, 2009). Tıbbi teşhis uygulamalarında, makine öğrenimi algoritmaları, hastalık teşhisinde kullanılan verileri analiz ederek doğru teşhisler yapabilir. Pazarlama tahminleri ve finansal analizde makine öğrenimi, müşteri davranışlarını ve pazar trendlerini analiz ederek gelecekteki talepleri veya finansal performansı tahmin edebilir (Goodfellow, 2016). Makine öğrenimi aynı zamanda güvenlik alanında da kullanılır. Örneğin, güvenlik kameralarından alınan verileri analiz ederek şüpheli aktiviteleri tespit edebilir veya kimlik doğrulama sistemlerini güçlendirebilir. Bu gibi uygulamalar, insanların analiz etmesi zor veya zaman alıcı olan büyük veri setlerini etkin bir şekilde işleyebilir ve değerli bilgileri ortaya çıkarabilir (Bishop, 2016). Sonuç olarak, makine öğrenimi, verilerin analiz edilmesi ve desenlerin tespit edilmesi için güçlü bir araçtır. Bu teknik, birçok farklı uygulama alanında kullanılarak tahminlerin yapılmasını, kararların verilmesini ve sistemlerin kendini geliştirmesini sağlar. Makine öğrenimi, teknolojik gelişmelerle birlikte daha da önem kazanmaktadır ve gelecekte daha fazla yenilik ve uygulama beklenmektedir (Shalev-Shwartz, 2014).

### 2.9.Yapay Zekâ

Yapay zeka (YZ), bilim ve teknolojide son yıllarda hızla gelişen ve pek çok alanda etkileri hissedilen bir kavramdır. YZ, bilgisayar sistemlerinin insan benzeri zekaya sahip olmasını sağlayarak, karmaşık sorunları analiz etme, öğrenme ve karar verme yeteneklerini kazanmasını amaçlar. Yapay zeka, bilgisayar sistemlerinin insanların yaptığı gibi düşünme ve problem çözme yeteneklerini kazanmasını hedefleyen bir alanı ifade eder. YZ, karmaşık veri analizi,

desen tanıma, makine öğrenimi ve derin öğrenme gibi yöntemlerle bilgisayarların öğrenme kapasitelerini artırır. Bu sayede, YZ sistemleri, verileri analiz ederek bilgi üretebilir, öngörülerde bulunabilir ve kararlar alabilir (Murphy, 2012).

### 2.9.1. Yapay Zeka, Derin Öğrenme ve Makine Öğrenimi Farkları

Yapay Zeka, Derin Öğrenme ve Makine Öğrenimi, birbirleriyle yakından ilişkili olan ancak farklı kavramlar ve tekniklerdir.

### 2.9.2. Yapay Zeka (YZ)

Yapay Zeka, bilgisayar sistemlerinin insan benzeri zekaya sahip olmasını hedefleyen geniş bir disiplindir. Amacı, karmaşık görevleri gerçekleştirebilen ve çevrelerini anlayabilen bilgisayar sistemleri oluşturmaktır. Yapay Zeka, insan benzeri düşünme, öğrenme, problem çözme ve karar verme yeteneklerini içerir (Kelleher, 2015).

### 2.9.3. Makine Öğrenimi (MO)

Makine öğrenimi, bilgisayar sistemlerinin verileri analiz ederek kendi kendine öğrenme yeteneği kazanmasını sağlayan bir yapay zeka yöntemidir. Makine öğrenimi, algoritmaların veri setlerindeki desenleri tanımlamasına ve gelecekteki verilere dayalı tahminler yapmasına olanak tanır. Makine öğrenimi, önceden programlanmış kurallar yerine veriye dayalı örüntüleri tanıyarak öğrenir (Marsland, 2014).

### 2.9.4. Derin Öğrenme

Veri tabanlı öğrenmeyi gerçekleştiren bir alt alanıdır ve daha karmaşık modellerin oluşturulmasına olanak tanır. Derin öğrenme algoritmaları, büyük veri setlerini analiz ederek derin katmanlarda karmaşık desenleri tanımlayabilir ve öğrenebilir (Rakhlin, 2014).


Şekil 2. 9. Yapay Zekaya Genel Bakış (beyaz.net)

Yapay zeka genel bir kavramdır ve Makine Öğrenimi bu alanda bir alt disiplindir. Derin Öğrenme ise Makine Öğrenimi içinde yer alan bir yaklaşımdır ve çok katmanlı yapay sinir ağlarıyla karmaşık desenleri tanımak için kullanılır. Yani, Derin Öğrenme, Makine Öğreniminin bir alt dalıdır ve daha karmaşık problemleri çözmek için kullanılan bir tekniktir.

## 2.10. Makine Öğreniminde Model Yaklaşımları

Makine öğrenimi, bir bilgisayar sisteminin belirli bir görevi veya problemi çözmek için verilerden öğrenme yeteneği kazandığı bir alanı ifade eder. Makine öğrenimi modelleri, bu öğrenme sürecini gerçekleştirmek için kullanılan algoritmalar ve yaklaşımlardır.

### 2.10.1. Doğrusal Regresyon

Doğrusal regresyon, bir bağımlı değişkenin bir veya daha fazla bağımsız değişkenle ilişkisini modellemek için kullanılan bir modeldir. Temel olarak, veri noktalarını en iyi şekilde temsil eden bir doğru veya düzlem bulmaya çalışır (Bengio, 2016).

$$y = W_0 + W_1X_1 + W_2X_2 + \cdots + W_nX_n$$

- $y$ bağımlı değişkeni (tahmin edilmek istenen değer)

- $X_1, X_2, \ldots X_n$ bağımsız değişkenler (tahmin edici değişkenler)
- $W_0, W_1, \ldots W_n$ katsayılar veya ağırlıklar (modelin öğrenmesi gereken parametreler)

Bu formülde $W_0$ kesme noktası veya doğrunun y-eksenini kestiği nokta (y-intercept) olarak adlandırılır ve $W_1, W_2, \ldots W_n$ bağımsız değişkenlerin katsayıları veya etkilerini temsil eder. Model, girdi verileri olan $X_1, X_2, \ldots X_n$ değerlerine dayanarak bağımlı değişken $y$ için tahminler yapar. Bu tahminler, öğrenilen katsayılar kullanılarak hesaplanır.

### 2.10.2. Lojistik Regresyon

Lojistik regresyon, sınıflandırma problemlerinde kullanılan bir modeldir. İki veya daha fazla sınıf arasında bir örneğin sınıflandırılmasıyla ilgilenir. Lojistik regresyon, giriş verilerini kullanarak bir olasılık dağılımı oluşturur ve örneğin hangi sınıfa ait olduğunu tahmin etmek için bu olasılıkları kullanır. Lojistik regresyon, bir bağımlı değişkenin (genellikle ikili bir sonuç) bir veya daha fazla bağımsız değişkenle ilişkisini modellemek için kullanılan istatistiksel bir yöntemdir. Bu yöntem, olasılık teorisi temelinde çalışır ve sonuç, 0 ile 1 arasında bir olasılık değeri olarak ifade edilir (Murphy, 2012).

$$P(Y = 1|X) = 1 / (1 + e\text{^} - (\beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta n * Xn))$$

$P(Y = 1|X)$ bağımlı değişkenin 1 olma olasılığını ifade eder.

$e$, tabanı Euler sayısı (yaklaşık 2.71828) olarak ifade eder.

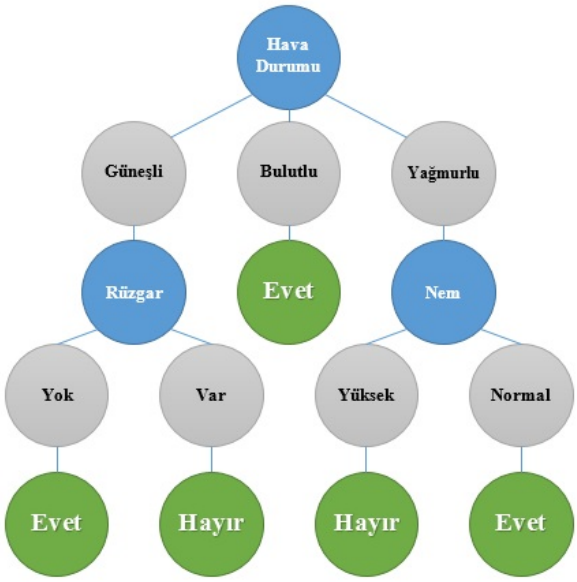
$\beta 0$, $\beta 1$, $\beta 2$, ..., $\beta$n, modelin tahmin ettiği bağımsız değişkenlerin katsayılarıdır.

$X1$, $X2$, ..., $Xn$, bağımsız değişkenlerin değerleridir.

Bağımsız değişkenlerin değerleri ve katsayıları kullanılarak, lojistik regresyon modeli bağımlı değişkenin 1 olma olasılığını hesaplar. Eğer hesaplanan olasılık belirli bir eşik değerinin üzerindeyse, model genellikle bağımlı değişkenin 1 olacağını tahmin eder; aksi durumda 0 olarak tahmin eder. Katsayılar ($\beta 0$, $\beta 1$, $\beta 2$, ..., $\beta$n), modelin eğitilmesi sırasında veriye uygun bir şekilde tahmin edilir. Bu tahmin işlemi, genellikle en küçük kareler yöntemi veya benzeri optimizasyon teknikleriyle gerçekleştirilir.

### 2.10.3. Karar Ağaçları

Karar ağaçları, bir dizi karar kuralını kullanarak verileri sınıflandıran veya tahmin eden bir modeldir. Her bir düğümdeki karar kuralı, veri özelliklerine dayalı olarak bir dal seçer ve sonuç olarak sınıf veya değer tahmini yapar.




Şekil 2. 10. Karar Ağacı (medium.com)

### 2.10.4. Destek Vektör Makineleri (SVM)

Destek vektör makineleri, sınıflandırma ve regresyon problemlerinde kullanılan bir modeldir. Veri noktalarını sınıflara ayırmak için bir hiper düzlem oluşturmaya çalışır. SVM, sınıflar arasındaki en iyi ayrımı sağlamak için destek vektörleri kullanır.

### 2.10.5. K-En Yakın Komşu (KNN)

K-En Yakın Komşu, basit ve popüler bir sınıflandırma ve regresyon modelidir. Veri noktaları arasındaki benzerlik ölçütlerine dayalı olarak tahminler yapar. Yeni bir veri noktası tahmin edildiğinde, en yakın k komşularının sınıfları veya değerleri dikkate alınır. KNN algoritması sınıflandırma ve regresyon için ayrı ayrı ele alınabilir (Sridharan, 2014).

$$D\_i \; = \; Uzaklık(x\_i, x\_q) \; for \; i \; = \; 1, 2, \ldots, n$$

Veri kümesi: $(x\_1, y\_1), (x\_2, y\_2), \ldots, (x\_n, y\_n)$

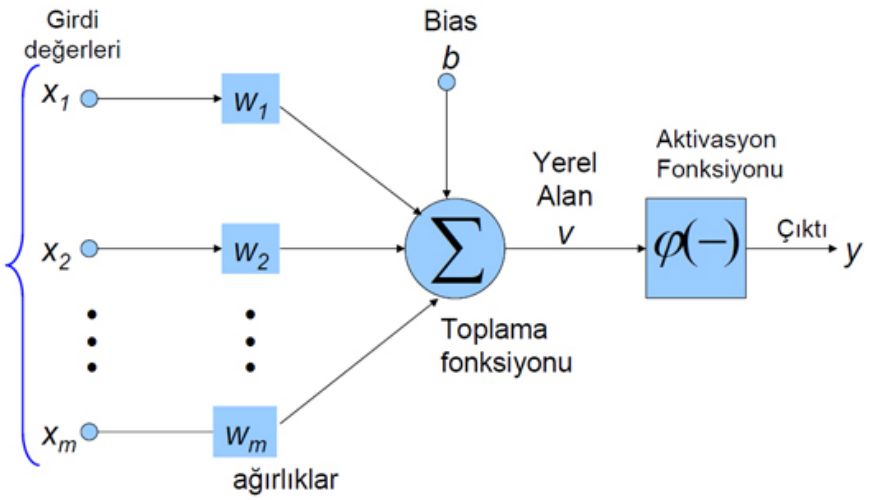
Sorgu noktası: $x\_q$

Komşu sayısı: $K$

Sınıflandırma için, uzaklıkları küçükten büyüğe sıralanır ve ilk K komşu seçilir. K, seçilen komşunun etiketleri incelenir ve en fazla sayıda olan sınıfı tahmin edilir. Regresyon için, uzaklıkları küçükten büyüğe sıralanır. K seçilen komşunun değerlerini kullanarak tahmin yapılır. Tahmin değeri, seçilen K komşunun değerlerinin ortalama veya ağırlıklı ortalaması olabilir. KNN algoritması, belirli bir sorgu noktasının çevresindeki en yakın veri noktalarını inceleyerek tahmin yapar veya sınıflandırma yapar. K değeri ve uzaklık fonksiyonu gibi parametreler, algoritmanın performansını etkiler.

### 2.10.6. Yapay Sinir Ağları (YSA)

Yapay sinir ağları, biyolojik sinir sisteminden esinlenen matematiksel modellemelerdir. Yapay sinir ağları, çok katmanlı yapay sinir ağlarından oluşur ve veri içindeki karmaşık desenleri tanımak ve tahminler yapmak için kullanılır. Derin öğrenme, yapay sinir ağlarının çok katmanlı yapılarına dayanan bir alt dalıdır. Temel olarak, bir yapay sinir ağı, girdi verilerini işler, ağırlıkları ayarlar ve çıktı sonuçlarını üretir (Joseph, 2010). Ağırlıklar, öğrenme sırasında veriye




Şekil 2. 11. Yapay Sınır Ağları (dergipark.org.tr)

uyarlanır.

Bir yapay sinir ağı, girdi katmanı, gizli katmanlar ve çıktı katmanı gibi farklı katmanlardan oluşur. Her katman, bir veya daha fazla yapay sinir hücresi (nöron) içerir. Ayrıca, katmanlar arasındaki bağlantılar ağırlıklarla temsil edilir. Bir girdi verisi (x) ağı girer ve işlemler şu şekilde gerçekleşir:

a. Her nöronun girdi katmanından çıkan bir ağırlığı (w) vardır.

b. Bu ağırlıklı girdiler, nöronun aktivasyon fonksiyonuna (genellikle sigmoid, ReLU gibi) uygulanır.
c. Aktivasyon fonksiyonunun çıktısı, gizli katmanlarda veya çıktı katmanında kullanılabilir.
d. Ağırlıklar, veriye göre öğrenme algoritmaları (genellikle geri yayılım) ile güncellenir.

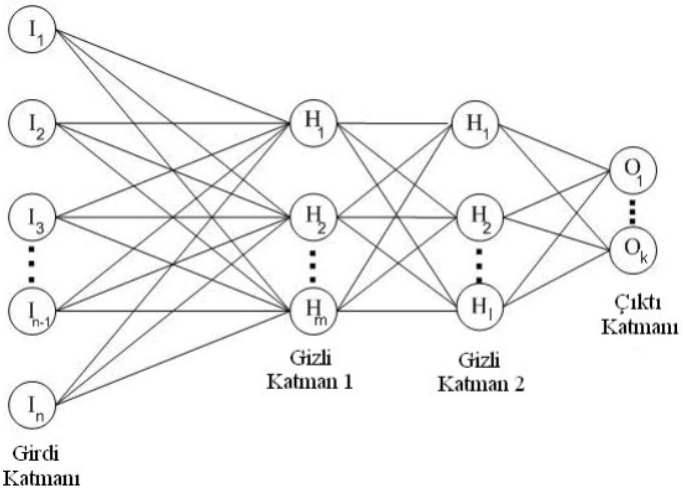
Ağın çıktı katmanındaki nöronlar, genellikle sınıflandırma veya tahmin sonuçlarını temsil eder. Bu sonuçlar, işlenen girdi verisine bağlı olarak belirli bir sınıfa veya değere karşılık gelir.

### 2.10.7. Yapay Zeka Sinir Ağları Modelleri

Yapay sinir ağları (artificial neural networks) adı verilen matematiksel modellemeleri ifade eder. Yapay sinir ağları, biyolojik sinir sistemlerinden esinlenerek tasarlanmış ve karmaşık problemleri çözmek için kullanılan yapay zeka yöntemleridir.

### 2.10.8. İleri Beslemeli Sinir Ağları (Feedforward Neural Networks)

İleri beslemeli sinir ağları, en temel yapay sinir ağı modelidir. Bu modelde, veri girişi ağı boyunca tek yönlü olarak ilerler. Her katmandaki nöronlar, önceki katmandaki nöronların çıkışlarından etkilenir ve son katmanda bir çıktı üretir. Yapay sinir ağlarının çoğu bu yapıya dayanır.




Şekil 2. 12. İleri Beslemeli Sinir Ağları (dergipark.org.tr)

İleri Beslemeli Sinir Ağları (Feedforward Neural Networks veya FNNs), en temel yapay sinir ağı türlerinden biridir ve veri akışının yalnızca ileri doğru (girdiden çıktıya) olduğu bir yapay sinir ağı türüdür (Courville, 2016).

$$a\hat{}(i+1)\_j \; = \; Aktivasyon(w\hat{}(i+1)\_j \; * \; a\hat{}(i) \; + \; b\hat{}(i+1)\_j)$$

Veri kümesi, $(x\_1, y\_1), (x\_2, y\_2), \ldots, (x\_n, y\_n)$

Girdi katmanı boyutu, $n\_i$

Gizli katmanları, :$n\_h1, n\_h2, \ldots, n\_hk$

Çıktı katmanı boyutu, $n\_o$

Her bir sinir hücresi, girdileri $(x)$ ve ağırlıkları $(w)$ kullanarak çıktı üretir. Bu çıktılar, genellikle bir aktivasyon fonksiyonundan geçirilir.

$$a \; = \; Aktivasyon(w \; * \; x \; + \; b)$$

$w$, ağırlık vektörü,

$x$, girdi vektörü,

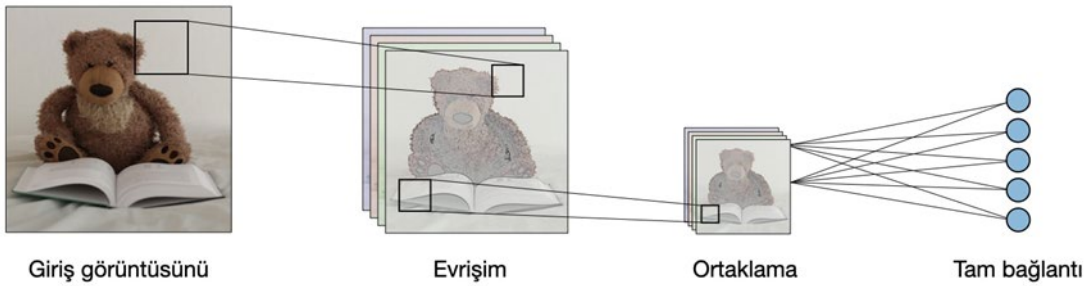
$b$, bias (ofset) terimi,

$a$, sinir hücresinin çıktısıdır.

Veri, girdi katmanına verilir. İlk gizli katmandan başlayarak, her bir katmandaki sinir hücreleri, girdileri ağırlıklarla çarparak ve aktivasyon fonksiyonuna uygulayarak yeni çıktılar üretir. Bu işlem, çıktı katmanına ulaşana kadar devam eder. Son gizli katmanın çıktıları, çıktı katmanında kullanılır ve aynı şekilde ağırlıklarla çarpılıp aktivasyon fonksiyonuna uygulanarak son tahmin veya sınıflandırma çıktıları elde edilir.

### 2.10.9. Evrişimli Sinir Ağları (Convolutional Neural Networks - CNN)

Evrişimli sinir ağları, özellikle görüntü işleme ve görsel tanıma problemlerinde etkili olan bir sinir ağı modelidir. Bu modelde, veri üzerinde evrişim işlemleri yapılır ve daha sonra öğrenilen özelliklerin birleştirilmesi ve sınıflandırma için kullanılması sağlanır. Evrişimli sinir ağları, veri içindeki mekansal ilişkileri ve desenleri etkili bir şekilde yakalayabilir.




Şekil 2. 13. Evrişimli Sinir Ağları (stanford.edu)

$$Conv(feature\_map, filter) = (feature\_map * filter) + bias$$

feature_map, giriş görüntüsünün belirli bir bölgesini,

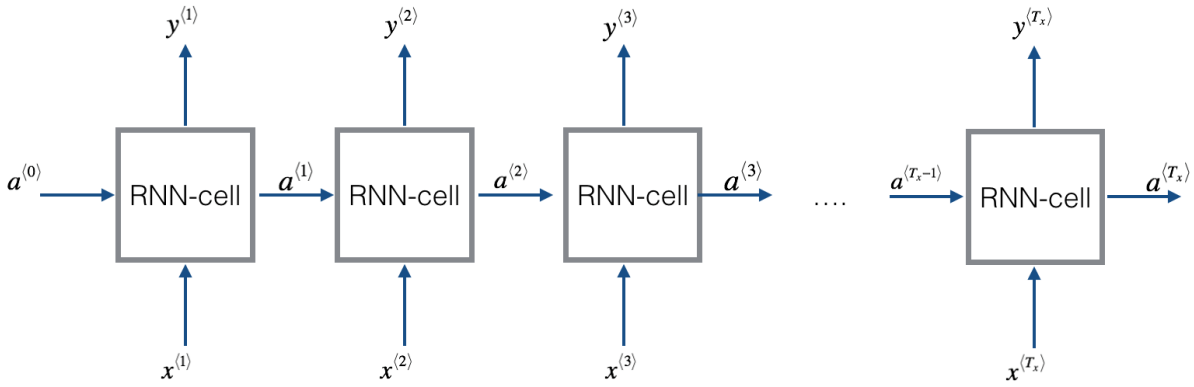
filter, kullanılan filtreyi,

bias, ofset terimini temsil eder.

Bir evrişim katmanı, görüntüdeki lokal özellikleri çıkarmak için kullanılır. Bu katmanda genellikle bir veya daha fazla filtre kullanılır. Bir filtre, görüntünün belirli bir bölgesindeki özellikleri vurgulamak için kullanılır. Evrişim işlemi, filtrelerin görüntü üzerinde kaydırılması ve özellik haritasının oluşturulması şeklinde gerçekleşir.

### 2.10.10. Rekürrent Sinir Ağları (Recurrent Neural Networks - RNN)

Rekürrent sinir ağları, sıralı verileri ve zaman serilerini işlemek için kullanılan bir sinir ağı modelidir. Bu modelde, geri dönüş bağlantıları sayesinde önceki durumlar hafızada tutulur ve geçmiş bilgiler gelecekteki tahminlerde kullanılır. RNN'ler, doğal dil işleme, metin üretimi ve zaman serisi tahmini gibi problemlerde etkilidir.




Şekil 2. 14. Rekürrent Sinir Ağları (github.com)

$$h\_t = Aktivasyon(W\_xh * x\_t + W\_hh * h\_(t-1) + b\_h)$$

$h\_t$, t zaman adımındaki gizli durum (hücre) çıktısı

$W\_xh$, girdi-gizli durum ağırlık matrisi

$x\_t, t$ zaman adımındaki girdi verisi

W_hh, gizli durum-gizli durum ağırlık matrisi

$h\_(t-1), (t-1)$ zaman adımındaki gizli durum (hücre) çıktısı

$b\_h$, gizli durum (hücre) bias (ofset) terimi

Gizli durum (hücre) çıktıları, çıktı katmanında kullanılır ve çıktılar ağırlıklarla çarpılıp aktivasyon fonksiyonuna uygulanarak elde edilir.

$$y\_t = Aktivasyon(W\_hy * h\_t + b\_y)$$

$y\_t, t$ zaman adımındaki çıktı

$W\_hy$, gizli durum-çıktı ağırlık matrisi

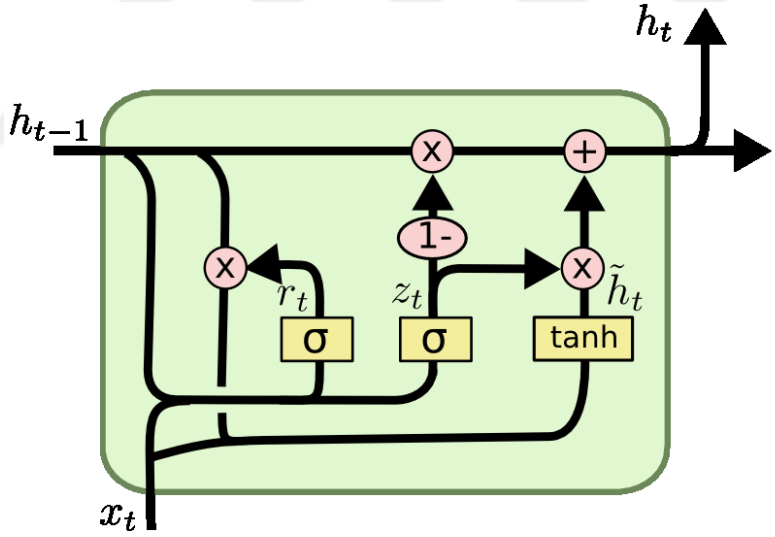
$h\_t, t$ zaman adımındaki gizli durum (hücre) çıktısı

$b\_y$, çıktı bias (ofset) terimi

RNN'ler, gizli durum (hücre) güncellemesi ve çıktı üretimi adımlarını içerir. Bu formülasyon, RNN'lerin temel işleyişini açıklar. Ancak, RNN'lerin bazı zorlukları da vardır, örneğin uzun zaman bağımlılıklarını yakalama konusunda sıkıntı yaşayabilirler. Bu nedenle, daha gelişmiş RNN türleri (örneğin, LSTM veya GRU gibi) bu zorlukları ele almak üzere tasarlanmıştır. RNN'ler, doğal dil işleme, metin üretimi, dil çevirisi, zaman serisi tahmini, müzik besteleme ve daha birçok alanda kullanılmaktadır. Yine de, daha uzun zaman bağımlılıklarını daha iyi ele almak için daha yeni ve gelişmiş model türleri (örneğin, dönüşümsel ve dönüşümsel-değer model türleri) de kullanılmaktadır.

### 2.10.11. Geçitli Tekrarlayan Birim (Gated Recurrent Unit - GRU)

Kapılı Tekrarlama Ünitesi (GRU), rekürrent sinir ağları (RNN'ler) ailesine ait bir türdür ve özellikle zaman serisi analizi, doğal dil işleme ve anormallik tespiti gibi alanlarda kullanılır. GRU, uzun dönem bağımlılıkları daha etkili bir şekilde ele almayı amaçlayan ve LSTM (Long Short-Term Memory) gibi bir başka RNN türüne alternatif olarak tasarlanmış bir modeldir.




Şekil 2. 15. Geçitli Tekrarlayan Birim (researchgate.net)

Geçitli Tekrarlayan Birim (GRU), gizli durum güncellemesi ve çıktı üretimi işlemlerini içeren bir rekürrent sinir ağı (RNN) modelidir. GRU, gizli durumun güncellenmesi ve geçmiş bilginin ne kadarını koruyacağının kontrol edilmesi için güncelleme ve sıfırlama kapıları kullanır.

Güncelleme kapısı, z_t = $sigmoid(W\_z * [x\_t, h\_(t-1)])$

Sıfırlama kapısı, r_t = $sigmoid(W\_r * [x\_t, h\_(t-1)])$

Güncelleme kapağı ile hafıza güncellemesi, $h\sim t\ =\ tanh(W\_h\ *\ [x\_t, r\_t\ *\ h(t-1)])$

Gizli durum, $h\_t\ =\ (1\ -\ z\_t)\ *\ h\_(t-1)\ +\ z\_t\ *\ h\sim\_t$

Çıktı üretimi, $y\_t\ =\ softmax(W\_y\ *\ h\_t)$

$x\_t, t$ zaman adımındaki girdi verisi

$h\_(t-1), (t-1)$ zaman adımındaki gizli durum (hücre) çıktısı

$z\_t$, t zaman adımındaki güncelleme kapısı

$r\_t, t$ zaman adımındaki sıfırlama kapısı

$h\sim\_t, t$ zaman adımındaki hafıza güncellemesi

$h\_t, t$ zaman adımındaki güncellenmiş gizli durum (hücre) çıktısı

$y\_t, t$ zaman adımındaki çıktı

$sigmoid, tanh\ ve\ softmax$ aktivasyon fonksiyonları

$W\_z, W\_r\ ve\ W\_h$, ağırlık matrisleri

GRU'nun formülasyonu, güncelleme ve sıfırlama kapıları aracılığıyla geçmiş bilginin nasıl korunacağını veya unutulacağını ve yeni bilginin nasıl entegre edileceğini açıklar. Bu mekanizmalar sayesinde GRU, uzun dönem bağımlılıkları daha etkili bir şekilde ele alabilir ve hızlı eğitim yeteneği ile öne çıkar.

### 2.10.12. Derin Sinir Ağları (Deep Neural Networks - DNN)

Derin sinir ağları, genellikle büyük veri setleri üzerinde eğitilir ve yüksek hesaplama gücü gerektirebilir. Çok sayıda parametrenin olduğu bu modeller, derin öğrenme yöntemlerini uygulayarak karmaşık desenleri ve ilişkileri tanımak için kullanılır. Derin sinir ağları, geniş bir uygulama yelpazesine sahiptir ve görüntü ve ses tanıma, doğal dil işleme, otomatik sürüş, oyun stratejileri ve sağlık gibi alanlarda büyük başarılar elde edebilir.

$$a\hat{}(i+1)\_j\ =\ Aktivasyon(w\hat{}(i+1)\_j\ *\ a\hat{}(i)\ +\ b\hat{}(i+1)\_j)$$

Her bir sinir hücresi (nöron), girdi verilerini (x) ve ağırlıkları (w) kullanarak çıktı üretir. Bu çıktılar, genellikle bir aktivasyon fonksiyonundan geçirilir.

$$a\ =\ Aktivasyon(w\ *\ x\ +\ b)$$

$w$, ağırlık vektörü

$x$, girdi vektörü

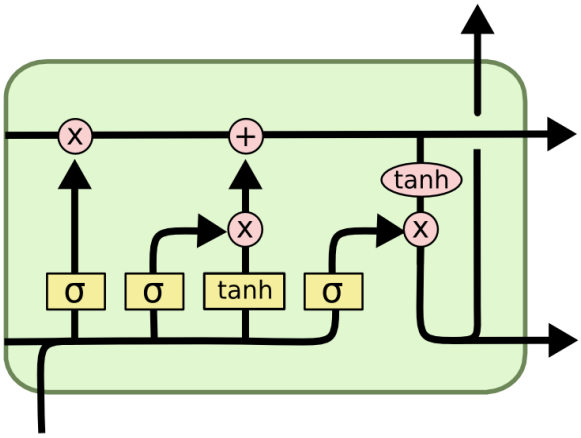
$b, bias$ (ofset) terimi

$a$, sinir hücresinin çıktısı.

Veri, girdi katmanına verilir ve ardışık olarak her bir gizli katmandan geçer. Her gizli katman, girdiyi ağırlıklarla çarparak ve aktivasyon fonksiyonuna uygulayarak yeni özellikleri çıkarır. Son gizli katmanın çıktıları, çıktı katmanında kullanılır ve aynı şekilde ağırlıklarla çarpılıp aktivasyon fonksiyonuna uygulanarak son tahmin veya sınıflandırma çıktıları elde edilir.

### 2.10.13. Uzun Kısa Süreli Hafıza (Long Short-Term Memory - LSTM)

Uzun Kısa Süreli Hafıza, RNN'lerin bellek sorununu çözmek için geliştirilmiş bir versiyonudur. LSTM, uzun vadeli bağımlılıkları daha etkin bir şekilde koruyabilen hafıza hücrelerine sahiptir. Bu sayede, LSTM modelleri karmaşık zaman serileri ve sıralı verilerin analizinde başarılı olabilir. GRU, özellikle daha kısa zaman serileri veya daha hızlı eğitim gerektiren durumlar için tercih edilebilir. LSTM'ye kıyasla daha az parametreye sahip olması nedeniyle, genellikle daha hızlı eğitilebilir ve daha az overfitting eğilimindedir. Ancak, veri setinizin karmaşıklığına ve uzun dönem bağımlılıkların önemine göre tercih yapılması gerekebilir. LSTM (Uzun Kısa Süreli Hafıza), derin öğrenme alanında yaygın olarak kullanılan bir yapay sinir ağı modelidir (Greff, 2017). Bu makalede, LSTM'nin ne olduğunu, nasıl çalıştığını ve kullanım alanlarını detaylı bir şekilde ele alacağız. LSTM, sıralı verilerin işlenmesi için tasarlanmış bir rekürrent sinir ağı (RNN) türüdür. Standart RNN'lere kıyasla, LSTM yapıları daha uzun zaman aralıklarındaki bağımlılıkları daha iyi koruyabilir. Bu, uzun vadeli bağımlılıkların ve sıralı verilerdeki desenlerin daha iyi analiz edilmesini sağlar. LSTM'nin temel bileşenleri "hücre" olarak adlandırılır. Her hücre, üç farklı kapıya sahiptir: unutma kapısı (forget gate), giriş kapısı (input gate) ve çıkış kapısı (output gate). Bu kapılar, hücrenin geçmiş bilgileri nasıl unutacağını, yeni bilgileri nasıl alacağını ve hücre çıkışının nasıl hesaplanacağını kontrol eder. Unutma kapısı, hücrenin geçmiş bilgilerini ne kadar koruyacağını belirler. Giriş kapısı, yeni bilgilerin hücreye nasıl ekleneceğini belirler. Çıkış kapısı ise hücrenin hangi bilgileri çıkış olarak vereceğini belirler. Bu kapılar, zaman içindeki farklı durumları dikkate alarak bilgilerin akışını düzenler ve böylece LSTM, uzun süreli bağımlılıkları daha etkin bir şekilde koruyabilir (Gers, 2000).




Şekil 2. 16. Uzun Kısa Süreli Hafıza (pytorch.org)

Unutma kapısı, $f\_t = sigmoid(W\_f * [x\_t, h\_(t-1)])$

Güncelleme kapısı, $i\_t = sigmoid(W\_i * [x\_t, h\_(t-1)])$

Hafıza hücresi güncellemesi, $g\_t = tanh(W\_g * [x\_t, h\_(t-1)])$

Hafıza hücresi, $C\_t = f\_t * C\_(t-1) + i\_t * g\_t$

Çıktı Kapısı, $o\_t = sigmoid(W\_o * [x\_t, h\_(t-1)])$

Güncellenmiş gizli durum, $h\_t = o\_t * tanh(C\_t)$

$x\_t, t$ zaman adımındaki girdi verisi

$h\_(t-1), (t-1)$ zaman adımındaki gizli durum (hücre) çıktısı

$f\_t, t$ zaman adımındaki unutma kapısı

$i\_t, t$ zaman adımındaki güncelleme kapısı

$g\_t, t$ zaman adımındaki hafıza hücresi güncellemesi

$C\_(t-1), (t-1)$ zaman adımındaki hafıza hücresi

$C\_t, t$ zaman adımındaki hafıza hücresi

$o\_t, t$ zaman adımındaki çıktı kapısı

$h\_t, t$ zaman adımındaki güncellenmiş gizli durum (hücre) çıktısı

$sigmoid$ ve $tanh$ aktivasyon fonksiyonları

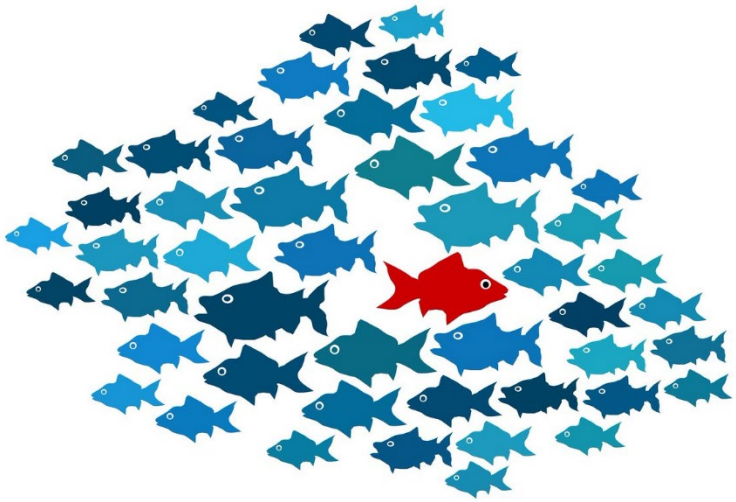
$W\_f, W\_i, W\_g\ ve\ W\_o,$ ağırlık matrisleri

LSTM'nin başarısı, büyük ölçüde geriye doğru yayılım (backpropagation) ve gradyan silinmesi (gradient vanishing) sorunlarının üstesinden gelmesine dayanır. Geriye doğru yayılım, ağın eğitimi sırasında hata gradyanını geriye doğru ileterek ağı günceller. LSTM, geçmiş zaman adımlarındaki hataları geriye doğru iletebilir ve böylece uzun vadeli bağımlılıkları daha iyi öğrenebilir. LSTM, çeşitli alanlarda başarılı bir şekilde kullanılmaktadır. Özellikle doğal dil işleme (natural language processing), metin sınıflandırma, duygu analizi, zaman serisi tahmini ve konuşma tanıma gibi alanlarda önemli bir rol oynamaktadır. Ayrıca, LSTM'nin diğer sinir ağı modelleriyle birleştirildiği ve daha karmaşık yapılar oluşturulduğu çalışmalar da mevcuttur (Graves, 2012).

Sonuç olarak, LSTM, uzun vadeli bağımlılıkları koruyabilen ve sıralı verilerdeki desenleri analiz etmek için etkili bir yapay sinir ağı modelidir. Hücrelerin unutma, giriş ve çıkış kapıları sayesinde geçmiş bilgilerin korunması, yeni bilgilerin eklenmesi ve çıkışın hesaplanması sağlanır. LSTM, geriye doğru yayılım ve gradyan silinmesi sorunlarının üstesinden gelerek uzun vadeli bağımlılıkları daha iyi öğrenebilir. LSTM'nin kullanım alanları oldukça geniştir. Doğal dil işleme, metin sınıflandırma, duygu analizi, zaman serisi tahmini, konuşma tanıma gibi alanlarda başarılı bir şekilde kullanılır. Ayrıca LSTM, diğer sinir ağı modelleriyle birleştirilerek daha karmaşık yapılar oluşturulabilir. LSTM'nin etkili bir şekilde kullanılabilmesi için uygun veri ön işleme, uygun hiperparametre ayarlaması ve yeterli eğitim verisi gibi faktörler önemlidir. Ayrıca, overfitting gibi sorunları önlemek için düzenleme yöntemleri de kullanılabilir. LSTM'nin avantajlarından biri, uzun vadeli bağımlılıkları koruyabilmesi ve sıralı verilerdeki desenleri daha iyi analiz edebilmesidir. Ancak, daha karmaşık yapıya sahip olması nedeniyle eğitim ve hesaplama süreleri daha uzun olabilir. LSTM, yapay zeka alanında önemli bir rol oynamaktadır ve birçok gerçek dünya uygulamasında başarılı sonuçlar vermektedir. Geliştiriciler ve araştırmacılar tarafından yaygın olarak kullanılan bir yapay sinir ağı modeli olarak LSTM, gelecekte de daha fazla geliştirme ve iyileştirmeye tabi tutulacaktır.

## 2.11. Anomali Tespiti

Yapay zeka, karmaşık veri setlerinde desenleri tanımak ve tahminler yapmak için kullanılan bir dizi algoritma ve teknikten oluşur. Ancak, gerçek dünyada, bazı durumlarda normal davranıştan sapmalar veya anormallikler meydana gelebilir.



Şekil 2. 17 Anomali Tespiti (analyticsvidhya.com)

Anomali, genel olarak normal veri dağılımından belirgin şekilde farklı olan veya beklenen davranışın dışında olan olaylar veya durumlar olarak tanımlanabilir. Anomali tespiti için kullanılabilecek bazı modeller ve yöntemler var.

### 2.11.1. İstatistiksel Yöntemler

Temel istatistiksel yöntemler, veri setindeki anormallikleri tespit etmek için kullanılabilir. Örneğin, veri setinin ortalaması, standart sapması veya tuhaflıkları belirleyen diğer istatistiksel özellikleri kullanarak, verilerdeki istatistiksel sapmaları tespit edilebilir (Chandola, 2009).

### 2.11.2. Eşik Değer Yöntemi

Eşik değer yöntemi, bir veri noktasının belirlenen bir eşik değerinden ne kadar uzak olduğunu değerlendirir. Eşik değeri, anormallik olarak kabul edilebilecek bir değeri temsil eder. Bu yöntemde, veri noktaları eşik değeri üzerinde veya altında ise anormal olarak etiketlenebilir.

### 2.11.3. Veriye Dayalı Anomali Tespiti Yöntemleri

#### 2.11.3.1. İstatistiksel Yöntemler

Bu yöntemler, veri setinin istatistiksel özelliklerini analiz ederek anormallikleri tespit etmeye çalışır. Örneğin, ortalamalar, standart sapmalar ve dağılım analizi gibi istatistiksel metrikler kullanılabilir (Hawkins, 2002).

#### 2.11.3.2. Aykırı Değer Tespiti

Bu yöntem, veri setindeki aykırı değerleri belirlemek için istatistiksel yöntemler veya veri dağılımları kullanır. Aykırı değerler, diğer veri noktalarından belirgin şekilde farklı olan ve anormal davranış sergileyen verilerdir (Aggarwal, 2013).

#### 2.11.3.3. Kümelenme Tabanlı Yöntemler

Bu yöntemler, veri setindeki verileri gruplara ayırarak benzerlik veya uzaklık ölçütlerine dayalı olarak anormallikleri tespit etmeye çalışır. Örneğin, k-ortalama kümeleme veya hiyerarşik kümeleme gibi algoritmalar kullanılabilir (Chawla, 2019).

### 2.11.4. Model Tabanlı Anomali Tespiti Yöntemleri

#### 2.11.4.1. Makine Öğrenimi Tabanlı Yöntemler

Bu yöntem, eğitim verileri kullanılarak bir model oluşturur ve bu model üzerinden anormallikleri tespit eder. Örneğin, Lojistik Regresyon, Destek Vektör Makineleri (SVM), Karar Ağaçları algoritmalar kullanılır (Haider, 2015).

#### 2.11.4.2. Derin Öğrenme Tabanlı Yöntemler

Bu yöntemler, derin sinir ağları ve özellikle Rekurrent Sinir Ağları (RNN) ve Uzun Kısa Süreli Hafıza (LSTM) gibi modeller kullanarak anormallikleri tespit etmeye odaklanır. Derin öğrenme, karmaşık yapıları ve gelişmiş öğrenme yetenekleri nedeniyle anormallik tespitinde etkili olur.

#### 2.11.4.3. Yapay Sinir Ağı Tabanlı Yöntemler

Bu yöntemler, genellikle bir veya daha fazla gizli katmana sahip yapay sinir ağı modellerini kullanır. Örneğin, Öz Düzenleyici Haritalama (SOM) veya Yapay Sinir Ağı Tabanlı Anomali Dedektörleri (ANNAD) gibi yöntemler kullanılır.

Yukarıda örneği verilen yöntemler, anormali tespiti için farklı yaklaşımlar sunar ve uygulanacak yöntem, veri seti özelliklerine, anormalliğin doğasına ve kullanım senaryosuna bağlı olarak seçilmelidir. Ayrıca, birden fazla yöntemin bir arada kullanıldığı veya birbirini tamamlayan bir yaklaşımın da tercih edilebileceği unutulmamalıdır.

## 2.12. FNN, CNN, RNN, GRU, DNN, LSTM Modellerin Zaman Serisine Bağlı Anormallik Tespitindeki Durumu

Zaman serisine bağlı anormallik tespiti için en iyi modelin seçimi, veri setinizin özelliklerine, anormalliklerin desenlerine ve performans hedeflerinize bağlıdır. Her bir modelin farklı avantajları ve dezavantajları vardır ve en iyi sonucu elde etmek için deneme yanılma yaklaşımı kullanılması gerekebilir.

### 2.12.1. FNN (Feedforward Neural Network)

Temel düzeyde anormallik tespiti yapabilir, ancak ardışık desenleri yakalama konusunda diğer modellere kıyasla daha zayıf olabilir. Daha düşük karmaşıklıkta veri setleri veya daha basit desenler üzerinde başarılı olabilir.

### 2.12.2. CNN (Convolutional Neural Network)

Görüntü işleme ve benzeri yapısal veriler için uygundur. Eğer zaman serileri özel bir şekilde temsil ediliyorsa (örneğin, spektrogramlar), anormallik tespiti için kullanılabilecek bir seçenek olabilir.

### 2.12.3. RNN (Recurrent Neural Network)

Ardışık desenleri yakalama ve uzun dönem bağımlılıkları ele alma yeteneğine sahiptir. Temel RNN'ler gradient kaybı sorunlarına yol açabilir, bu nedenle daha gelişmiş RNN türleri (GRU, LSTM) tercih edilir.

### 2.12.4. GRU (Gated Recurrent Unit)

Uzun dönem bağımlılıkları yakalama ve ardışık desenleri öğrenme konusunda özellikle etkilidir. Zaman serisine bağlı anormallik tespitinde başarılı olabilir.

### 2.12.5. DNN (Deep Neural Network)

Genel veri analizi için kullanılabilir, ancak saf zaman serisi verileri üzerinde anormallik tespiti için spesifik avantaj sağlamayabilir. Eğer özellik mühendisliği veya farklı özellik türleri kullanılıyorsa, anormallik tespitinde kullanışlı olabilir.

### 2.12.6. LSTM (Long Short-Term Memory)

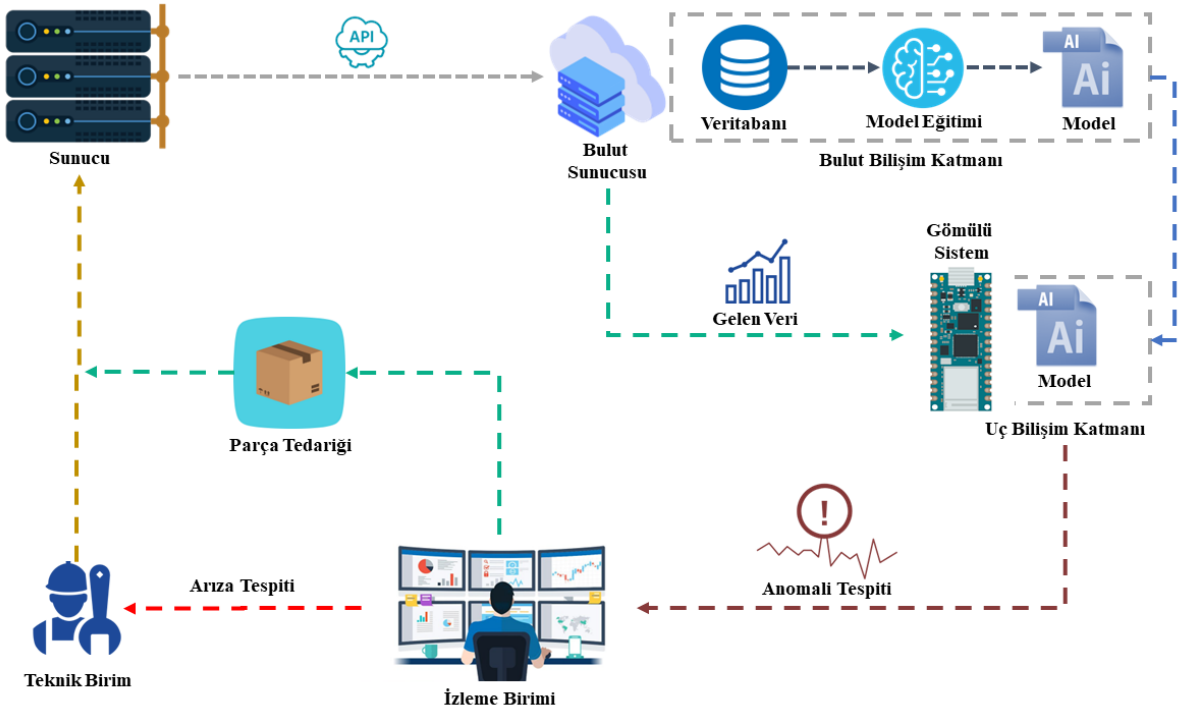
Uzun dönem bağımlılıkları ele alma ve zaman içindeki desenleri öğrenme yeteneği nedeniyle zaman serisine bağlı anormallik tespitinde etkili bir seçenektir. Yukarıdaki model yaklaşımlarında problemin sahip olduğu parametreler model seçiminde bize yardımcı olacaktır. Bu tez de anomali tespiti için zaman serilerine bağlı bir veri seti kullanılacaktır. Bu da yukarıdaki modellerle karşılaştırıldığında GRU ve LSTM kullanmanın daha doğru bir yaklaşım olduğu kabul edilebilir. GRU (Gated Recurrent Unit) ve LSTM (Long Short-Term Memory), her ikisi de rekürrent sinir ağı (RNN) türleri olup, zaman serisine bağlı anormallik tespiti gibi görevlerde kullanılabilecek modellerdir. Her iki model de uzun dönem bağımlılıkları ele alabilme yetenekleriyle öne çıkar, ancak bazı farklılıkları vardır. Hangi modelin daha iyi performans göstereceği, veri setinin özelliklerine ve problem bağlamına bağlıdır. LSTM, daha geleneksel RNN'lere kıyasla daha gelişmiş bir hücre yapısına sahiptir. Uzun dönem bağımlılıkları daha etkili bir şekilde yakalayabilir ve zaman içindeki desenleri öğrenme konusunda güçlüdür. Fakat daha fazla parametreye sahip olduğu için daha fazla hesaplama gücü ve veri gerektirebilir. GRU, LSTM'e kıyasla daha az parametreye sahiptir ve bu nedenle daha hızlı eğitilebilir. Özellikle daha kısa zaman serileri veya hızlı eğitim gerektiren durumlar için tercih edilebilir. Uzun dönem bağımlılıkları da iyi ele alabilir, ancak LSTM kadar karmaşık yapıları öğrenme yeteneğine sahip olmayabilir. Hangi modelin daha iyi performans göstereceğini belirlemek için aşağıdaki faktörleri göz önünde bulundurmanız önemlidir.

a. Veri Seti Boyutu: Büyük bir veri setiniz varsa, LSTM gibi daha karmaşık modeller daha iyi sonuçlar verir.
b. Veri Serisinin Uzunluğu: Daha uzun zaman serileri, LSTM veya GRU gibi modellerin uzun dönem bağımlılıkları ele almasına yardımcı olur.
c. Hızlı Eğitim Gereksinimi: Eğer hızlı eğitim gerekiyorsa, GRU daha hızlı bir seçenek olacaktır.
d. Performans ve Doğruluk: Deneme ve hata yaparak, her iki modelin performansını karşılaştırılır.

# 3. MATERYAL ve YÖNTEM

## 3.1. Sistem Tasarımı

Bu bölüm Uç Bilişim ile LSTM Tabanlı Arıza Tespiti mimarisini ve uygulama adımlarına genel bir bakış sunar.




Şekil 3. 1. Uç Bilişim ile LSTM Tabanlı Arıza Tespiti Mimarisi

Bu çalışma, donanım arızalarının önceden tespit edilmesi ve müdahale edilmesi için bütünleşik bir sistem sunmaktadır. Bu sistem, sensör verileri analizi, yapay zekâ modeli eğitimi, uç katmanda çalışma ve etkili bir uyarı sistemi ile donatılmıştır. Bu sayede, sunucu arızalarının neden olduğu kesintilerin önüne geçilmesi ve sistem sürekliliğinin sağlanması hedeflenmektedir. Sunucu üzerinde yer alan sensörlerden elde edilen verilerin kullanılmasıyla donanım arızalarını önceden tahmin etmeyi hedeflemektedir. Bu amaçla, sunucudan elde edilen veriler API aracılığıyla bulut sunucusuna aktarılmaktadır. Aktarılan veriler, sıcaklık, voltaj gibi ölçümleri içermektedir ve bu veriler bulut sunucusunda bulunan veri tabanına kaydedilmektedir. Yeterli miktarda veri toplandıktan sonra, bulut sunucusundaki yapay zeka modeli, toplanan bu verilere dayanarak kendisini eğitmeye başlamaktadır. Başarı eşiği belirlendikten sonra, modelin çıktısı uç cihazda çalıştırılmak üzere gömülü bir sistem kartına Wi-Fi üzerinden iletilir. Bu aşamada, bulut sunucusu pasif hale getirilir ve model sadece uç

katmanda faaliyet gösterir. Bu şekilde, maliyet tasarrufu sağlanmış olur. Uç katmanda çalışan yapay zekâ modeli, sunucudan anlık veriler alarak anomali tespiti yapmaya başlar. Algılanan anormal veriler, izleme birimine uyarı olarak iletilir. Uyarı ekibi, bu alarmları dikkate alarak muhtemel arızalı parçanın teminini önceden gerçekleştirir ve teknik ekip, planlanan bir zaman diliminde parçayı yeni bir parça ile değiştirerek sistemin kesintisiz çalışmasını sağlar (Bkz. Şekil 3.1).

## 3.2.Veri Kaynağı ve Toplama

Sunucularında bulunan sensörler, sunucuların fiziksel ve sanal bileşenlerinin durumunu izlemek ve yönetmek amacıyla kullanılır. Bu sensörler, sunucuların performansını izlemek, sorunları tespit etmek ve gerekli önlemleri almak için önemlidir.

### 3.2.1. Sıcaklık Sensörleri

Sunucunun farklı bölgelerindeki sıcaklıkları izlemek için kullanılır. Aşırı ısınma durumlarını tespit ederek sunucu performansını ve verimliliğini korumak önemlidir.

a. İşlemci Sıcaklığı: İşlemci (CPU) sıcaklığı, işlemcinin ısınma durumunu gösterir. Yüksek işlemci sıcaklıkları, işlemcinin aşırı ısındığını ve performans düşüklüğü veya arıza riski olabileceğini gösterebilir.
b. Bellek Sıcaklığı: Bellek modüllerinin sıcaklık durumunu izlemek önemlidir. Aşırı ısınma, bellek performansını etkileyebilir.
c. Grafik İşlemci (GPU) Sıcaklığı: Sanal grafik işlemcilerinin sıcaklık durumu, sanal grafik performansını etkileyebilir.
d. Sunucu Kasa Sıcaklığı: Sunucunun kasası veya kabinin iç sıcaklığını izlemek, sunucunun genel ısınma durumunu gösterir.
e. Fan Hızı ve Sıcaklığı: Fanların sıcaklık durumu ve hızı, soğutma sistemlerinin etkinliğini ve fan arızalarını gösterir.
f. Voltaj Regülatörleri (VRM) Sıcaklığı: Sunucunun güç beslemesi için kullanılan voltaj regülatörlerinin sıcaklık durumunu izlemek önemlidir.
g. Depolama Birimi Sıcaklığı: Depolama birimlerinin sıcaklığı, veri depolama bileşenlerinin sağlığını ve performansını etkileyebilir.

### 3.2.2. Fan Hızı Sensörleri

Sunucunun fanlarının hızını izleyerek aşırı ısınma durumlarını önlemek için kullanılır. Fan hızlarını kontrol ederek donanımın aşırı ısınmasını engellemek amaçlanır.

a. CPU Fan Hızı: İşlemci soğutma fanının hızı, işlemcinin ısınma durumuna bağlı olarak ayarlanır. İşlemci fan hızı, işlemcinin yüküne göre otomatik olarak ayarlanarak aşırı ısınmayı önlemek amaçlanır.
b. Kasa Fan Hızları: Sunucunun kasası veya kabininin içinde bulunan fanların hızları, genel soğutma performansını gösterir. Kasa fanları, sunucunun iç sıcaklığını düşürmek ve bileşenlerin aşırı ısınmasını önlemek için kullanılır.
c. Grafik İşlemci (GPU) Fan Hızı: Sanal grafik işlemcilerinin fan hızları, sanal grafik performansını ve ısınma durumunu gösterir.
d. Güç Kaynağı Fan Hızı: Sunucunun güç kaynağının fanının hızı, güç kaynağının performansını ve soğutmasını izlemek amacıyla kullanılır.
e. Depolama Fanları: Sunucunun depolama birimlerini soğutmak için kullanılan fanların hızları, veri depolama bileşenlerinin sağlığını etkileyebilir.

### 3.2.3. Voltaj Sensörleri

Sunucunun güç beslemesi ve voltaj düzeylerini izlemek için kullanılır. Anormal voltaj düzeyleri, donanım sorunlarını veya güç kaynaklı sıkıntıları gösterebilir.

a. İşlemci (CPU) Voltajı: İşlemcinin çalışması için gereken voltaj düzeyini izlemek amacıyla kullanılır. Anormal voltaj düzeyleri, işlemci performansını veya dayanıklılığını etkileyebilir.
b. Bellek Voltajı: Bellek modüllerinin güç beslemesi için kullanılan voltaj düzeyini izlemek önemlidir. Yanlış voltaj, bellek sorunlarına yol açabilir.
c. Grafik İşlemci (GPU) Voltajı: Sanal grafik işlemcilerinin güç beslemesi için kullanılan voltaj düzeyi, sanal grafik performansını etkileyebilir.
d. Ana Kart Voltajı: Ana kartın genel güç beslemesi ve voltaj düzeylerini izlemek, sunucunun güvenli ve sağlıklı çalışmasını sağlamak için önemlidir.
e. Depolama Birimi Voltajı: Depolama birimlerinin güç beslemesi için kullanılan voltaj düzeylerini izlemek, veri depolama bileşenlerinin güvenliğini ve performansını etkileyebilir.

### 3.2.4. Güç Tüketimi Sensörleri

Sunuculardaki güç tüketimi sensörleri, sunucunun enerji tüketimini izlemek ve güç verimliliğini optimize etmek amacıyla kullanılır. Bu sensörler, sunucunun enerji kullanımını gözlemleyerek gereksiz enerji tüketimini azaltmak, enerji maliyetlerini düşürmek ve çevresel etkiyi en aza indirmek amacıyla kritik öneme sahiptir. Sunucunun enerji tüketimini izlemek ve gerektiğinde enerji tasarrufu sağlamak için kullanılır.

a. Toplam Sistem Güç Tüketimi: Sunucunun toplam enerji tüketimini izlemek amacıyla kullanılır. Bu veri, sunucunun enerji kullanımını genel olarak değerlendirmek için önemlidir.
b. İşlemci (CPU) Güç Tüketimi: İşlemcinin enerji tüketimini izlemek, işlemci performansını ve güç verimliliğini değerlendirmek için kullanılır.
c. Bellek Güç Tüketimi: Bellek modüllerinin enerji tüketimini izlemek, bellek kullanımının enerji etkilerini değerlendirmek için kullanılır.
d. Depolama Birimi Güç Tüketimi: Depolama birimlerinin enerji tüketimini izlemek, veri depolama bileşenlerinin enerji verimliliğini değerlendirmek için kullanılır.
e. Grafik İşlemci (GPU) Güç Tüketimi: Sanal grafik işlemcilerinin enerji tüketimini izlemek, sanal grafik performansını ve güç verimliliğini değerlendirmek için kullanılır.

### 3.2.5. Depolama Sensörleri

Depolama birimlerinin sağlık durumunu ve kapasitesini izlemek için kullanılır. Depolama birimlerinde oluşabilecek arızaların önceden tespit edilmesi amaçlanır.

a. Depolama Kapasitesi: Sunucunun depolama birimlerinin kullanılan ve boş kapasitelerini izlemek amacıyla kullanılır. Depolama birimlerinin doluluk seviyeleri, veri depolama alanının yönetiminde yardımcı olur.
b. Depolama Performansı: Depolama birimlerinin okuma ve yazma hızlarını izlemek, depolama performansının etkinliğini değerlendirmek için kullanılır.
c. Depolama Sıcaklığı: Depolama birimlerinin sıcaklık durumunu izlemek, aşırı ısınma durumlarını önlemek ve veri depolama bileşenlerinin sağlığını korumak amacıyla kullanılır.
d. Depolama Durumu ve Durumu: Depolama birimlerinin çalışma durumu, arıza durumu veya bakım gereksinimleri gibi durumları izlemek, kesintilere karşı erken uyarı sağlamak amacıyla kullanılır.

e. Depolama Arayüz Hızı: Depolama birimlerinin veri aktarım hızını izlemek ve depolama arayüzünün etkinliğini değerlendirmek amacıyla kullanılır.

### 3.2.6. Bellek Sensörleri

Sunucunun bellek modüllerinin durumunu ve performansını izlemek amacıyla kullanılır. Bu sensörler, bellek kullanımını izlemek, bellek hatalarını tespit etmek ve sunucunun genel performansını optimize etmek amacıyla önemlidir Sunucunun bellek kullanımını izlemek ve bellek sorunlarını tespit etmek için kullanılır.

a. Toplam Bellek Kapasitesi: Sunucunun toplam bellek kapasitesini izlemek, bellek kullanımını değerlendirmek için kullanılır.
b. Kullanılan Bellek: Sunucunun şu anda kullanılan bellek miktarını izlemek, bellek kullanımını kontrol etmek için önemlidir.
c. Boş Bellek: Sunucunun boş bellek miktarını izlemek, bellek kullanımını optimize etmek için kullanılır.
d. Bellek Hızı: Bellek modüllerinin hızını izlemek, bellek performansını değerlendirmek için kullanılır.
e. Bellek Sıcaklığı: Bellek modüllerinin sıcaklık durumunu izlemek, aşırı ısınma durumlarını önlemek için kullanılır.
f. Bellek Durumu: Bellek modüllerinin çalışma durumunu izlemek, bellek hatalarını veya arızalarını tespit etmek için önemlidir.
g. Bellek Hataları: Bellek hatalarını izlemek, bellek hatalarının tespit edilmesi ve giderilmesi için kullanılır.

### 3.2.7. Ağ Bağlantı Sensörleri

Sunuculardaki ağ bağlantı sensörleri, sunucunun ağ bağlantılarının durumunu izlemek amacıyla kullanılır. Bu sensörler, ağ trafiğini gözlemlemek, ağ bağlantılarının hızını ve sağlığını değerlendirmek ve ağ sorunlarını tespit etmek için önemlidir. Sunucularındaki ağ bağlantı sensör verileri, sunucunun ağ performansını optimize etmek, ağ sorunlarını önceden tespit etmek ve bağlantı kesintilerini minimize etmek amacıyla kullanılır.

a. Ağ Bağlantı Durumu: Sunucunun ağ bağlantısının durumunu izlemek, ağ bağlantısının kesilmesi veya sorun yaşanması durumunda erken uyarı sağlamak için kullanılır.

b. Ağ Bağlantı Hızı: Sunucunun ağ bağlantısının hızını izlemek, ağ trafiğini ve hızını değerlendirmek için önemlidir.
c. Ağ Trafik Durumu: Sunucunun ağ trafiğini izlemek, ağ kullanımını değerlendirmek ve ağ sorunlarını tespit etmek için kullanılır.
d. Ağ Bağlantısı Kalitesi: Sunucunun ağ bağlantısının kalitesini izlemek, ağ performansını ve bağlantı sorunlarını değerlendirmek için kullanılır.
e. Ağ Gecikmesi (Ping) Süreleri: Sunucunun ağ bağlantısının gecikme sürelerini izlemek, ağ erişim hızını ve sorunlarını değerlendirmek için kullanılır.

### 3.2.8. İşlemci Sensörleri

Sunuculardaki işlemci sensörleri, sunucunun işlemci (CPU) bileşenlerinin durumunu ve performansını izlemek amacıyla kullanılır. Bu sensörler, işlemci sıcaklığını, işlemci yükünü ve diğer işlemci özelliklerini gözlemleyerek sunucunun işlemci performansını optimize etmek ve arızaları önceden tespit etmek amacıyla kritik öneme sahiptir.

a. İşlemci Sıcaklığı: İşlemci bileşeninin sıcaklık durumunu izlemek, işlemcinin aşırı ısınma durumlarını tespit etmek ve işlemci performansını korumak için önemlidir.
b. İşlemci Hızı: İşlemci çalışma hızını izlemek, işlemci performansını değerlendirmek için kullanılır.
c. İşlemci Yükü: İşlemcinin ne kadar yüklü olduğunu izlemek, işlemci kullanımını değerlendirmek ve performans sorunlarını tespit etmek için önemlidir.
d. İşlemci Çekirdek Durumu: İşlemci çekirdeklerinin durumunu izlemek, her bir çekirdeğin performansını değerlendirmek için kullanılır.
e. İşlemci Gerilimi: İşlemci bileşeninin gerilim durumunu izlemek, işlemci sağlığını değerlendirmek için kullanılır.

### 3.2.9. PCIe Slot Sensörleri

PCIe yuvalarının durumunu izlemek ve bağlı bileşenlerin çalışma durumunu gözlemlemek için kullanılır.

a. PCIe Yuva Durumu: PCIe yuvalarının çalışma durumunu izlemek, genişleme yuvalarının sağlığını ve kullanılabilirliğini değerlendirmek için kullanılır.
b. PCIe Hızı: PCIe yuvalarının veri aktarım hızını izlemek, genişleme yuvasının performansını değerlendirmek için önemlidir.

c. PCIe Kullanımı: PCIe yuvalarının kullanım durumunu izlemek, hangi genişleme kartlarının hangi yuvalarda kullanıldığını ve genişleme kapasitesini değerlendirmek için kullanılır.
d. Genişleme Kartı Durumu: Kurulu genişleme kartlarının durumunu izlemek, genişleme kartlarının çalışmasını ve performansını gözlemlemek için kullanılır.

### 3.2.10. Soğutma Sensörleri

Sunuculardaki soğutma sensörleri, sunucunun iç sıcaklık durumunu, fan hızlarını, soğutma sistemlerinin performansını ve diğer ilgili parametreleri izlemek amacıyla kullanılır. Bu sensörler, sunucunun aşırı ısınma durumunu tespit etmek, soğutma sistemi performansını optimize etmek, enerji verimliliğini artırmak ve donanım arızalarını önceden tahmin etmek amacıyla kritik öneme sahiptir.

a. Ortam Sıcaklığı: Sunucu odasının veya kabinin iç sıcaklığını izlemek, genel soğutma performansını değerlendirmek için kullanılır.
b. Kasa İçi Sıcaklık: Sunucunun kasası içindeki sıcaklığı izlemek, sunucunun genel ısınma durumunu ve bileşenlerin sağlığını gözlemlemek için kullanılır.
c. Fan Hızları: Fanların hızını izlemek, soğutma sistemi performansını ve fanlar arasındaki dengeyi değerlendirmek için önemlidir.
d. Soğutma Performansı: Soğutma sisteminin genel performansını izlemek, aşırı ısınma riskini azaltmak ve sunucunun stabil çalışmasını sağlamak için kullanılır.
e. Fan Durumu: Fanların çalışma durumunu izlemek, fan arızalarını tespit etmek ve gerektiğinde müdahalede bulunmak için kullanılır.

### 3.2.11. Gürültü Sensörleri

Sunucunun gürültü seviyelerini izlemek ve olası fan veya soğutma sorunlarını tespit etmek amacıyla kullanılır.

a. Gürültü Seviyesi İzleme: Sensörler, sunucu odasındaki gürültü seviyelerini izler ve bu verileri yöneticilere raporlar. Aşırı gürültü durumlarında uyarılar veya bildirimler gönderebilirler.
b. Fan Kontrolü: Sunuculardaki fan hızlarını izler ve yönetir. Fan hızlarındaki anormallikleri tespit edebilir ve gerektiğinde fanları ayarlayarak gürültü seviyelerini düşürebilirler.

c. Sıcaklık İzleme: Sıcaklık sensörleri kullanarak sunucunun iç ısısını izlerler. Aşırı ısınma durumlarında gerekli önlemleri alabilirler.
d. Uyarılar ve Bildirimler: Sensörler, anormal gürültü seviyeleri veya sıcaklık durumları tespit edildiğinde yöneticilere uyarılar gönderebilirler.
e. Performans İyileştirmesi: Sunucu odasındaki gürültü düzeyini optimize ederek hem çalışanların hem de donanımın verimliliğini artırabilirler.

Sunucularda anomali tespiti, sunucuların, sanal makinelerin ve diğer bileşenlerin performansını izlemek ve olası sorunları önceden tespit etmek için kullanılan bir tekniktir. Anomali tespiti için kullanılan veriler, genellikle CPU kullanımı, bellek kullanımı, disk kullanımı, ağ trafiği ve diğer bileşenlerin davranışlarını ve performansını izleyen verilerdir. Bu veriler, olası sorunları önceden tespit etmek, performans düşüşlerini önlemek ve hızlı müdahalede bulunmak amacıyla analiz edilir. Anomali tespiti, belirli bir model veya davranışın normale göre sapmalarını belirlemeyi içerir. Yukarıdaki sensör metriklerine göre anomali tespiti aşağıdaki alanlardan biri veya birden fazlası referans alınarak yapmak mümkündür. Bu tür sunucu tabanlı anomali tespiti, sunucu ortamlarında genellikle aşağıdaki alanlara dayanmaktadır:

a. Performans Verileri: Sanal makinelerin, sunucuların ve ağ trafiğinin performans verileri izlenir. CPU kullanımı, bellek kullanımı, disk ve ağ trafiği gibi parametreler analiz edilir.
b. Sıcaklık ve Soğutma Verileri: Sunucuların ve donanım bileşenlerinin sıcaklık değerleri izlenir. Aşırı ısınma veya soğutma sorunları tespit edilebilir.
c. Ağ Trafiği ve Bağlantı Verileri: Sanal makineler arası ağ trafiği, bağlantı hızları ve ağ bağlantı durumları izlenir. Anormal trafik veya bağlantı sorunları tespit edilebilir.
d. Depolama Verileri: Disk kullanımı, IOPS (giriş/çıkış işlemi), depolama hızları ve veri mağazalarının durumu izlenir.
e. Olay Günlükleri: Sunucu ve sanal makine olay günlükleri incelenir. Anormal hatalar, uyarılar veya olaylar tespit edilebilir.

Bu çalışmada, makine öğrenimi veya yapay zeka yöntemleri kullanılarak sunucu üzerindeki sensörler aracılığıyla elde edilen verilerin analizi ve işlenmesi amaçlanmaktadır. Bu veriler, sunucunun performansını ve çevresel koşullarını izlemek için kullanılan temel ölçümleri içerir. Bu analiz, sunucuların davranışını anlamak, normal işleyişin dışındaki durumları tespit etmek ve potansiyel sorunları önceden belirlemek amacıyla gerçekleştirilir. Verilerin analizi ve

işlenmesi için LSTM modelinin eğitimi için kullanılacaktır. Bu model, sunucu üzerinde bulunan güç sensörlerinden gelen verileri temel alarak sunucunun normal davranışını öğrenecektir. Model, bu verilere dayanarak belirli bir davranış modelini oluşturacak ve eğitim süreci sırasında bu modeli oluşturan parametreleri ayarlayacaktır. Eğitilmiş model, sunucunun gerçek zamanlı verilerini analiz ederek anormal durumları tespit etmeye çalışacaktır. Örneğin, sunucunun kullandığı toplam gücün beklenmedik anormal şekilde yükselme gibi durumları model tarafından belirlenebilir. Bu anormaliteler, potansiyel bir donanım sorununun sorunlarına nedendir. Anomali tespiti, veri merkezi yönetimini daha etkili hale getirmek ve sunucu performansını artırmak için önemli bir araçtır. Bu yöntem sayesinde, potansiyel sorunlar önceden tespit edilerek hızlı müdahale sağlanabilir, kesinti süreleri azaltılabilir ve veri merkezi verimliliği optimize edilebilir.

### 3.3.Veri Toplama

Küme yapıda bir sunucuya ait üzerinde bulunan güç sensöründen elde edilen bilgiler kullanılarak, 15 dakika aralıklarla toplanan veri setiyle 1 aylık bir zaman dilimi kapsanmıştır (Tablo 3.2.1). Bu çalışmanın amacı, elde edilen bu zaman serisi verilerini kullanarak LSTM (Uzun Kısa Süreli Bellek) modeli yardımıyla gelecekteki güç tüketimini tahmin etmek ve aynı zamanda anormal güç artışlarında oluşabilecek donanım arızalarına dikkat çekmektir. Sunucu üzerinde kullanılan güç sensörü bir sunucunun veya bilgisayar sisteminin sistem kartının güç tüketimini ölçen sensördür. Sistem kartı, bir sunucunun ana bileşenlerini barındıran ve birçok önemli işlevi yerine getiren ana devre kartıdır. Bu kart, işlemci, bellek, depolama birimleri, ağ bileşenleri ve diğer temel bileşenlerin bağlandığı ve etkileşimde bulunduğu bir platformdur. Veri setinde yer alan enerji ifadesi, belirli bir sistem kartının güç tüketimini belirtir. Bu tür güç tüketimi ölçümleri, sunucunun veya bilgisayar sisteminin enerji verimliliğini değerlendirmek, potansiyel enerji tasarrufu fırsatlarını tanımlamak ve gerektiğinde donanım arızalarını tespit etmek için kullanılabilir.

| Zaman | Enerji (W) |
|---|---|
| **1.05.2023 09:00** | **25740 Watts** |
| **1.05.2023 09:15** | **24570 Watts** |
| **1.05.2023 09:30** | **35100 Watts** |
| **1.05.2023 09:45** | **32760 Watts** |
| **1.05.2023 10:00** | **26910 Watts** |
| **1.05.2023 10:15** | **24570 Watts** |
| **1.05.2023 10:30** | **25740 Watts** |
| **1.05.2023 10:45** | **38610 Watts** |

| | |
|---|---|
| **1.05.2023 11:00** | **26910 Watts** |
| **1.05.2023 11:15** | **25740 Watts** |
| **1.05.2023 11:30** | **24570 Watts** |
| **1.05.2023 11:45** | **33930 Watts** |
| **1.05.2023 12:00** | **25740 Watts** |
| **1.05.2023 12:15** | **26910 Watts** |
| **1.05.2023 12:30** | **32760 Watts** |
| **1.05.2023 12:45** | **24570 Watts** |
| **1.05.2023 13:00** | **29250 Watts** |

**Tablo 3.2 1 Örnek Veriseti**

### 3.4.LSTM Modeli

Bu çalışmada LSTM modeli tensorflow kütüphanesi kullanarak hazırlanmıştır. TensorFlow, Google tarafından geliştirilen ve yayınlanan açık kaynaklı bir makine öğrenimi ve derin öğrenme çerçevesidir. TensorFlow, yapay sinir ağları ve derin öğrenme modellerinin oluşturulması, eğitilmesi ve dağıtılması için güçlü bir kütüphane sunar. Özellikle LSTM (Uzun Kısa Süreli Bellek) modeli gibi karmaşık derin öğrenme modellerinin oluşturulması için TensorFlow sıklıkla kullanılır. LSTM modelinin birçok parametresi vardır ve bu parametreler modelin davranışını ve performansını önemli ölçüde etkiler.

#### 3.4.1. Hücre Sayısı

LSTM katmanında bulunan hücre sayısı, modelin karmaşıklığını belirler. Daha fazla hücre, modelin daha fazla öğrenme kapasitesine sahip olmasını sağlar, ancak aynı zamanda daha fazla hesaplama gücü gerektirir.

#### 3.4.2. Giriş Sekansı Uzunluğu

LSTM'nin girdi olarak alacağı veri sekansının uzunluğunu belirtir. Özellikle zaman serileri analizinde veya doğal dil işlemede bu parametre önemlidir.

#### 3.4.3. Dönem Sayısı

LSTM'nin bir zaman serisi üzerinde çalışırken geriye dönük kaç adımı dikkate alacağını belirler. Örneğin, bir dönem sayısı 3 ise, model üç zaman adımı boyunca geriye dönük bilgileri kullanabilir.

### 3.4.4. Giriş Boyutu

LSTM'nin girdi verisinin boyutunu belirtir. Örneğin, metin işleme uygulamalarında kelime gömme (word embedding) kullanılıyorsa, bu boyut kelime gömme boyutuna eşit olacaktır.

### 3.4.5. Çıkış Boyutu

LSTM'nin çıkış verisinin boyutunu belirtir. Sınıflandırma için kullanılıyorsa, sınıf sayısına eşit olabilir.

### 3.4.6. Aktivasyon Fonksiyonları

LSTM'deki giriş kapısı, unutma kapısı ve çıkış kapısı için kullanılan aktivasyon fonksiyonlarını belirtir. Genellikle sigmoid veya tanh fonksiyonları kullanılır.

### 3.4.7. Droplar ve Dönüşüm Oranları

Aşırı uyuma karşı koruma sağlamak için kullanılan droplar ve dönüşüm oranlarını belirtir.

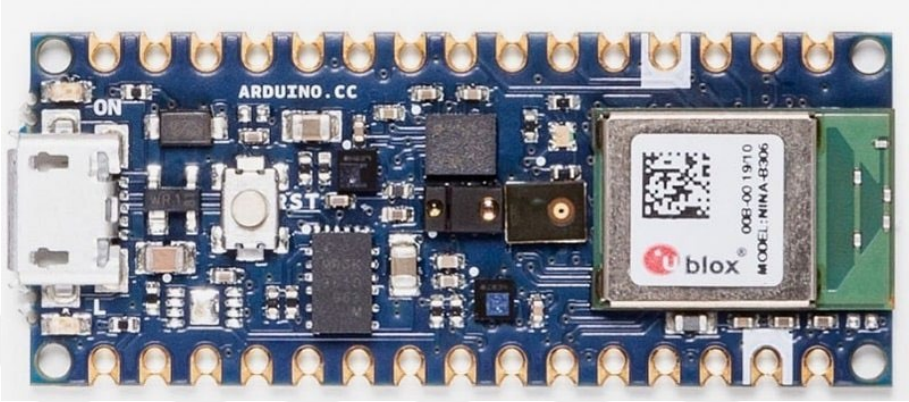
### 3.4.8. Aktivasyon Fonksiyonları

LSTM'nin hücreleri ve kapıları için kullanılan aktivasyon fonksiyonlarını belirtir. Genellikle tanh veya sigmoid fonksiyonları kullanılır.

Yukarıda bahsi geçem LSTM'nin temel parametrelerinden bazılarıdır. LSTM modeli, bu parametrelerin uygun şekilde ayarlanmasıyla daha iyi performans elde edebilir ve belirli bir uygulama veya veri setine en iyi şekilde uyarlanabilir.

### 3.5.Geliştirme Kartı (TinyML)

TinyML (Tiny Machine Learning), mikrodenetleyiciler gibi düşük güçlü ve kaynak kısıtlı cihazlarda yapay zeka (AI) ve makine öğrenimi (ML) modelinin uygulanması için kullanılan bir kavramdır. Bu cihazlar, genellikle akıllı telefonlar, IoT (Nesnelerin İnterneti) cihazları, sensörler, gömülü sistemler ve diğer benzeri donanım parçalarını içerir.




Şekil 3. 2 TinyML Geliştirme Kartı

TinyML, bu tür cihazlarda yapay zeka uygulamalarını çalıştırmak için kaynakların sınırlı olduğu durumları hedefler. TinyML, cihazların sınırlı pil kapasitesiyle çalışmasını sağlamak için enerji verimli algoritmalar ve modeller kullanır. Ayrıca bu tür cihazlar genellikle sınırlı bellek ve işlemci gücüne sahiptir, bu nedenle TinyML, küçük boyutlu ve hafif modelleri tercih eder.

| **Özellikler** | **Değer** |
|---|---|
| Besleme Gerilimi | 3,3 V |
| Çalışma Akımı | 15 - 200 mA |
| Mikrodenetleyici | nRF 52840 |
| Dijital Pin | 14 |
| PWM Pin | 6 |
| Analog Pin | 8 |
| Arayüz | GPIO, SPI, I2C, USART, PWM |
| Wi-Fi | IEEE 802.11 b/g/n |

Tablo 3.5 1. TinyML Özellikleri

### 3.6.Geliştirme Ortamı

LSTM modelinin güçlü bir bilgisayar üzerinde hızlı geliştilmesi ve geliştirilen modelin daha sonraki süreçte dağıtılabilmesi adına bulut ortamına ihtiyaç duyulmuştur. Bu kapsamda 12 saat ücretsiz bulut sunucu tahsisi sağlayan Google Colab ortamı kullanılmıştır.

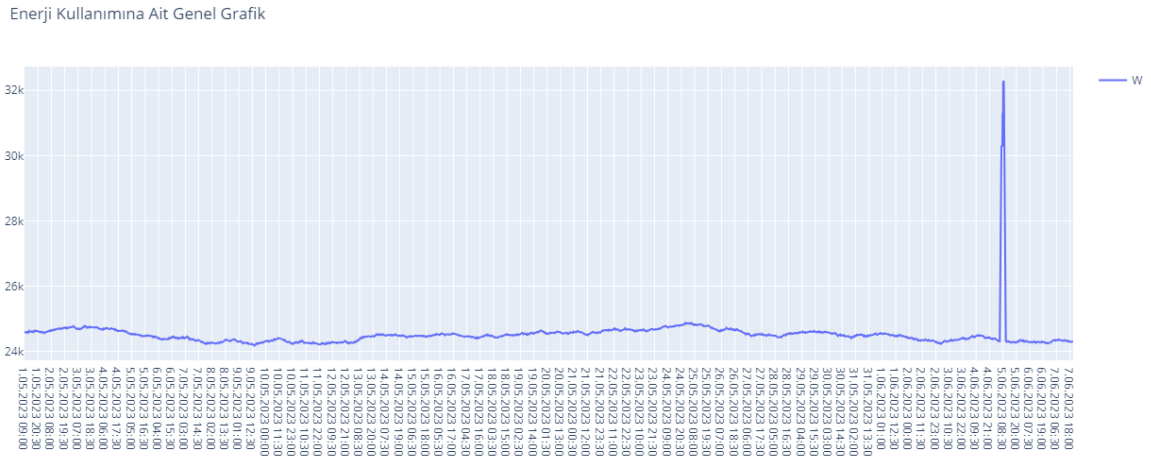
| Bileşen | Değer |
|---|---|
| CPU | Intel(R) Xeon(R) CPU @ 2.30GHz |
| RAM | 13 GB |
| Disk | 100 GB |
| Ekran Kartı | Tesla K80 |

Tablo 3.5 2 Geliştirme Ortam Özellikleri

## 4. BULGULAR ve TARTIŞMA

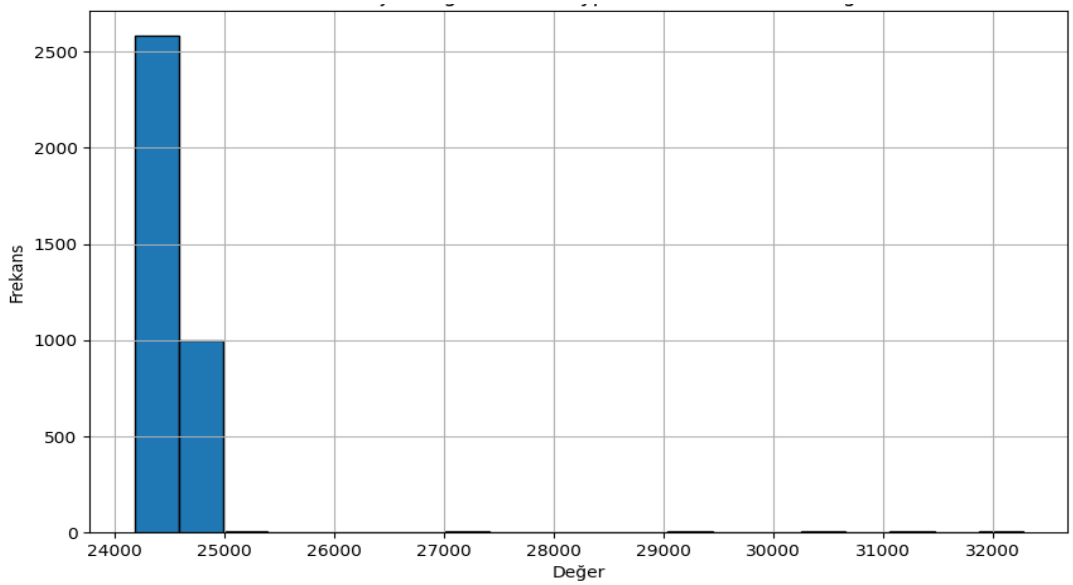
LSTM Tabanlı Arıza Tespiti (UBLTAT) ile sunucuya özgü yapılan anomali tespitine ait sonuçlar değerlendirilmiştir. UBLTAT sisteminde anormallik durumlarınınanalizleri yapılmış ve alınan sonuçlar değerlendirilmiştir. UBLTAT sisteminde arıza tespiti için geliştirilen makine öğrenmesi tabanlı LSTM modeli verileri analiz edilmiş ve sonuçlar grafiklerle birlikte yorumlanmıştır.

Çalışma, Cisco UCS B200 M4 Blade tipi sunucusu üzerinde bulunan güç sensöründen alınan veriler kullanışmıştır. Bu veriler, 15 dakika aralıklarla bir aylık bir süre boyunca toplanmıştır (Şekil 4.1).




Şekil 4. 1. Sunucuya Ait Güç Değerleri

Elde edilen verilerin histogram grafiği incelendiğinde, sunucuya ait verilerin ağırlıklı olarak 24.000 ile 25.000 arasında olduğu gözlemlenmektedir. Bu aralık, sunucunun genellikle bu güç seviyelerinde çalıştığını ve bu veri aralığının belirli bir süre boyunca sürekli olarak tekrarlandığını göstermektedir. Bu istikrarlı güç seviyeleri, sunucunun enerji tüketimi veya iş yükü ile ilgili önemli bilgilere işaret edebilir ve bu verilerin performans izleme veya optimizasyon amaçlarıyla kullanılması potansiyeline işaret edebilir.




Şekil 4. 2. Enerji Histogram Grafiği

### 4.1. Model Mimarisi

Sunucu üzerinde geliştirilen model, bir zaman serileri veri seti üzerinde çalışmak üzere tasarlanmış bir LSTM ağıdır. Model, toplamda 6 katmandan oluşmuştur (Şekil 4.2).

```
model = Sequential()
model.add(LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(rate=0.1))
model.add(RepeatVector(X_train.shape[1]))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(rate=0.1))
model.add(TimeDistributed(Dense(X_train.shape[2])))
model.compile(optimizer='adam', loss='mae')
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               66560

 dropout (Dropout)           (None, 128)               0

 repeat_vector (RepeatVecto  (None, 12, 128)           0
 r)

 lstm_1 (LSTM)               (None, 12, 128)           131584

 dropout_1 (Dropout)         (None, 12, 128)           0

 time_distributed (TimeDist  (None, 12, 1)             129
 ributed)

=================================================================
Total params: 198273 (774.50 KB)
Trainable params: 198273 (774.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Şekil 4. 3 LSTM Katmanı

### 4.1.1. Katman 1: LSTM Katmanı

İlk katman bir LSTM katmanıdır ve 128 nöron içerir. LSTM, uzun ve kısa vadeli bağımlılıkları modelleme yeteneği ile zaman serileri verilerinin işlenmesi için mükemmel bir seçenektir. Bu katman, girdi verisini alır ve içindeki gizli desenleri çıkarmak için kullanılır.

### 4.1.2. Katman 2: Dropout Katmanı

İlk LSTM katmanının çıkışından sonra gelen ikinci katman bir dropout katmanıdır ve oranı %0.1 olarak belirtilmiştir. Dropout, aşırı öğrenmeyi önlemek için kullanılır. Her adımda rastgele seçilen nöronların devre dışı bırakılması, modelin daha genelleme yapmasını sağlar.

### 4.1.3. Katman 3: RepeatVector Katmanı

Üçüncü katman, RepeatVector katmanıdır. Bu katman, ilk LSTM katmanının çıkışını alır ve giriş zaman adımlarının sayısına kopyalar. Böylece, model, ardışık tahminler üretebilir. Bu, özellikle ardışık veri analitiği problemleri için önemlidir.

### 4.1.4. Katman 4: LSTM Katmanı (128 Nöron, return_sequences=True)

Dördüncü katman, bir başka LSTM katmanıdır ve 128 nöron içerir. Bu katman, bir önceki katmandan gelen çıkışı alır ve return_sequences parametresi sayesinde her zaman adımında bir çıkış üretir. Bu sayede, ardışık tahminlerin hesaplanması için gerekli olan sekans oluşturulur.

### 4.1.5. Katman 5: Dropout Katmanı (0.1 Oranında)

Beşinci katman, ikinci LSTM katmanının çıkışından sonra gelen bir dropout katmanıdır ve aşırı öğrenmeyi önlemek için kullanılır. Yine %0.1 oranında rastgele nöronlar devre dışı bırakılır.

### 4.1.6. Katman 6: TimeDistributed ve Dense Katmanları

Altıncı katman, bir TimeDistributed katmanı ve ardından bir Dense (Tam Bağlantılı) katmanı içerir. Bu katmanlar, her bir zaman adımındaki girdilere tam bağlantılı işlemler uygular. Bu, tahminlerin her zaman adımı için ayrı ayrı hesaplanmasını sağlar ve ardışık veri analitiği problemleri için çok önemlidir. Sonuç olarak, bu 6 katmandan oluşan model, zaman serileri verilerini işlemek ve ardışık tahminler üretmek için tasarlanmış bir LSTM tabanlı bir yapıdır. Modelin katman sayısı, kullanılan veriye ve probleme bağlı olarak değiştirilebilir ve model optimizasyonu sürecinde ayarlanabilir.

### 4.2. Model Eğitimi

Bu çalışmada, model eğitimi için kullanılan veri seti, genellikle yaygın bir uygulama olan %80 eğitim verisi ve %20 test verisi olarak ayrılmıştır. Veri setinin bu şekilde ayrılması, modelin eğitilmesi ve performansının değerlendirilmesi için standart bir yaklaşımdır. Eğitim sürecine odaklandığımızda, kullanılan 6 katmanlı LSTM modeli, X_train ve y_train değişkenlerine atanan eğitim verilerini modele girdi olarak alır. Bu veriler, modelin temelini oluşturan öğrenme sürecinin ana bileşenleridir. Eğitim süresi, bu modelde 500 adım olarak belirlenmiştir. Bu, modelin eğitim verileri üzerinde kaç tur yapacağını belirler. 500 adım boyunca eğitim yapılması, modelin veriye uyum sağlamasını ve öğrenme sürecini tamamlamasını sağlar. Ancak, bu sayı veri seti ve probleme bağlı olarak değiştirilebilir. Ayrıca, eğitim sırasında bir bölümün doğrulama (validation) için ayrılması önemlidir. Bu, modelin her adım sonunda doğrulama verileri üzerinde performansını değerlendirmesine olanak tanır. Bu örnekte, eğitim verilerinin %10'u doğrulama için ayrılmıştır. Bu, modelin eğitim verilerinin dışındaki verilere nasıl genelleştiğini izlemek için kullanılır (Şekil 4.4).

```
history = model.fit(X_train, y_train, epochs=500, batch_size=128, validation_split=0.1,
                    callbacks=[keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, mode='min')], shuffle=False)
```

```
Epoch 1/500
20/20 [==============================] - 19s 416ms/step - loss: 24505.8438 - val_loss: 24526.3867
Epoch 2/500
20/20 [==============================] - 6s 307ms/step - loss: 24497.1211 - val_loss: 24520.9648
Epoch 3/500
20/20 [==============================] - 7s 371ms/step - loss: 24493.1953 - val_loss: 24517.8340
Epoch 4/500
20/20 [==============================] - 6s 297ms/step - loss: 24490.1875 - val_loss: 24514.8887
Epoch 5/500
20/20 [==============================] - 7s 373ms/step - loss: 24487.3398 - val_loss: 24512.0996
```

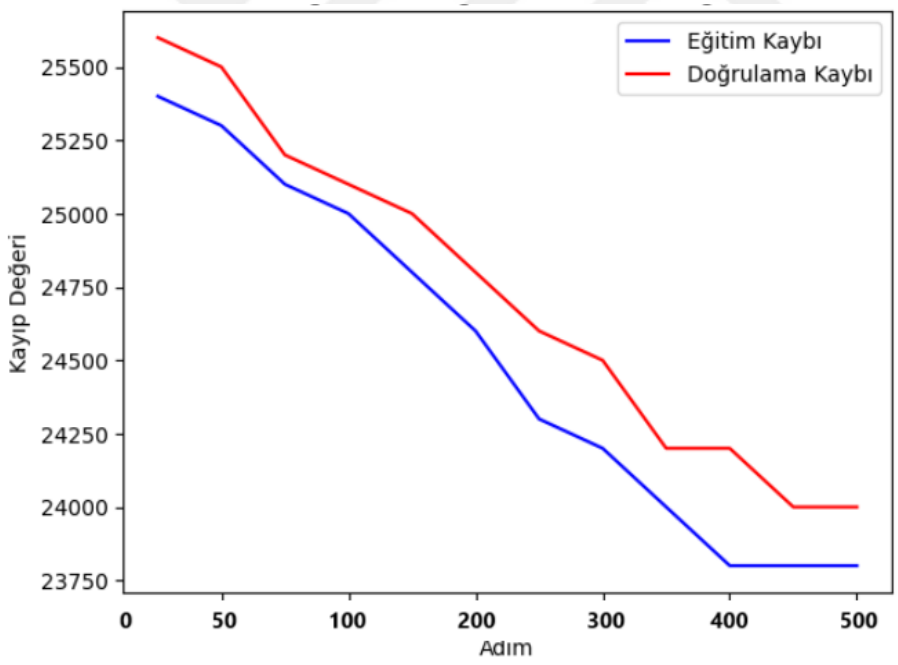
Şekil 4. 4 Model Eğitimi

Son olarak, eğitim sırasında kullanılan bir erken durma mekanizması vardır. Doğrulama kaybının belirli bir süre boyunca düşmediğini izler ve belirli bir sabır süresini aşarsa eğitimi durdurur. Bu, modelin aşırı öğrenmeyi önlemesine yardımcı olur ve en iyi performansı elde etmek için önemlidir. Eğitim kaybı ve Doğrulama kaybı grafikleri, bir makine öğrenme veya derin öğrenme modelinin eğitimi sırasında kaybın nasıl değiştiğini görselleştiren önemli grafiklerdir. Bu grafikler, modelin öğrenme sürecini ve performansını izlemeye yardımcı olur. Eğitim kaybı grafiği, modelin eğitim verileri üzerindeki performansının bir ölçüsüdür. Bu kayıp, modelin eğitim verilerini ne kadar iyi uyum sağladığını gösterir. Eğitim süreci ilerledikçe (her adım sonunda), model eğitim verilerine daha fazla uyum sağlar ve bu nedenle eğitim kaybı genellikle azalır. Ancak, eğitim kaybının hızla düşmeye devam etmesi (örneğin, sert bir şekilde

inmesi), modelin aşırı uyum (overfitting) yapabileceğini gösterebilir. Bu nedenle, training loss grafiği aşırı öğrenme konusunda bir gösterge olarak kullanılabilir. Doğrulama kaybı, modelin doğrulama verileri üzerindeki performansının bir ölçüsüdür. Bu kayıp, modelin genelleme yeteneğini gösterir. Her adım sonunda, model, doğrulama verileri üzerinde tahminlerde bulunur ve bu tahminlerin kaybını hesaplar. Bu kayıp, modelin yeni ve görünmeyen verilere ne kadar iyi uyum sağladığını gösterir. İdeal durumda, doğrulama kaybı, eğitim kaybı gibi azalmalıdır. Ancak, doğrulama kaybı, eğitim kaybından önemlidir çünkü modelin aşırı öğrenme yapmadığından emin olmak için kullanılır. Eğer doğrulama kaybı eğitim sırasında artmaya başlarsa veya sabit kalıyorsa, bu modelin aşırı uyum yaptığı veya optimize edilmesi gerektiği anlamına gelir.

### 4.3.Model Sonuçları

Kullanılan veri setinde enerji verileri isteğe bağlı olarak ölçeklendirilmemiştir. Bu nedenle model eğitimi sonrasında modelin durumu hakkında yorum yapmayı sağlayan model kayıp grafiğinde bu değerler büyük gözükmektedir (Şekil 4.5).



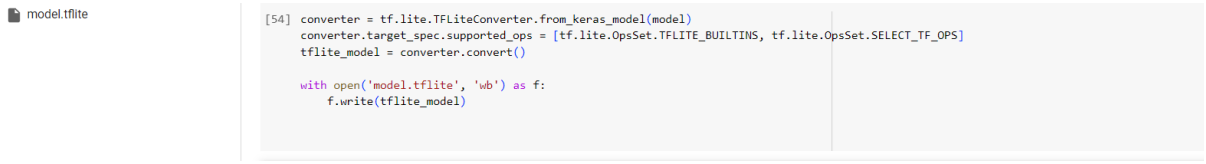

Şekil 4. 5 Model Kayıp Grafiği

Öncelikle, kullanılan LSTM modeli 6 katmandan üretilmiştir ve eğitim süresi 500 adımdan oluşmaktadır. Bu, modelin oldukça karmaşık bir yapıya sahip olduğunu ve uzun bir süre boyunca eğitildiğini gösterir. Eğitim sonuçlarını değerlendirmek için oluşturulan kayıp grafiği, bu sürecin nasıl ilerlediğini görsel olarak yansıtmaktadır. Grafiği incelediğimizde şunları

gözlemlenmektedir. Modelin öğrenme başarısı, eğitim başladığı andan itibaren artış göstermektedir. Eğitim kaybı (mavi çizgi), modelin eğitim verilerine daha iyi uyum sağladığını gösterir. Bu, modelin veriye adapte olma yeteneğinin iyi olduğunu göstermektedir. Ancak, yaklaşık olarak 400 adımdan sonra, eğitim kaybı grafiği bir doygunluk noktası oluşturmaktadır. Bu, modelin eğitim verilerine uyum sağlama hızının yavaşladığını ve daha fazla öğrenmeye direnç gösterdiğini göstermektedir. Modelin artık veriyi daha iyi anlamadığı bir aşamadır. Aynı şekilde, doğrulama kaybı grafiği (kırmızı çizgi), modelin genelleme yeteneğini değerlendirmek için önemlidir. Bu grafiği incelediğimizde, yaklaşık olarak 450 adımdan sonra bir başka doygunluk noktası gözlemlenmektedir. Bu, modelin yeni verilere uyum sağlama yeteneğinin sınırlı olduğunu ve hatta kötüleşebileceğini göstermektedir. Sonuç olarak, modelin eğitim süreci 500 adımda tamamlanmış gibi görünmektedir. Ancak, adım sayısını arttırmanın modelin daha fazla öğrenmesine katkı sağlamayabileceği açıktır. Modelin doygunluğa ulaştığı ve daha fazla adımın performansı iyileştirmediği görülmektedir. Bu gözlemler, modelin davranışını daha iyi anlamamıza ve modelin optimize edilmesi gerekip gerekmediğini değerlendirmemize yardımcı olmaktadır.

### 4.4.Model Dönüşümü

TensorFlow Lite, TensorFlow modelini hafifletilmiş bir formatta ve optimize edilmiş bir şekilde taşımak ve çalıştırmak için kullanılan bir araçtır. TensorFlow Lite'a dönüştürmek, özellikle mobil cihazlar, gömülü sistemler ve diğer kaynak sınırlı platformlarda derin öğrenme modelinin uygulanabilirliğini artırmak için önemlidir. TensorFlow Lite, modelin boyutunu ve bellek kullanımını önemli ölçüde azaltır. Bu, modelin daha hızlı yüklenmesini ve çalıştırılmasını sağlar. Uç nokta cihazlar için kaynak sınırlı platformlarda önemlidir. Modelin çalışma hızını artırır. Daha küçük boyutlu ve optimize edilmiş model, daha hızlı tahminler yapabilir. Bu nedenlerle, TensorFlow modelini TensorFlow Lite'a dönüştürmek, modelinizi daha taşınabilir, hızlı ve verimli hale getirir. Bu, özellikle mobil uygulamalar ve gömülü sistemler için önemlidir. Bu bağlamda, geliştirilen karmaşık 6 katmanlı LSTM modelinin uç noktadaki cihazlarda etkili bir şekilde çalıştırılabilmesi için TensorFlow Lite formatına dönüştürülmesi gerekmektedir (Şekil 4.6). Bu dönüşüm, bulut tabanlı makine öğrenme modellerinin ağırlık ve boyutları düşürülerek, sınır bilişim teknolojisinin gücünden yararlanmak amacıyla önemlidir. TensorFlow Lite'a dönüşüm, modelin daha hafif ve taşınabilir hale getirilmesini sağlar. Bu, mobil cihazlar, gömülü sistemler ve diğer uç noktalar gibi kaynak




Şekil 4. 6 Model Dönüşümü

sınırlı platformlarda modelin sorunsuz bir şekilde çalıştırılmasını kolaylaştırır. Ayrıca, bulut üzerindeki işlem gücünü azaltarak uç noktalarda daha hızlı ve daha verimli sonuçlar elde etmeyi mümkün kılar. Bu dönüşüm, veri gizliliği açısından da önemlidir. Çünkü model, kullanıcının cihazında çalıştığı için kullanıcının verileri cihazda kalır ve güvenli bir şekilde işlenir. Bu da kullanıcıların verilerinin daha iyi korunmasını sağlar. Sonuç olarak, TensorFlow Lite'a dönüşüm, karmaşık makine öğrenme modellerini uç noktalarda kullanılabilir ve etkili hale getirebilir. Bu, sınır bilişim teknolojisinin önemli bir bileşeni olarak modelin uç noktalara taşınmasını sağlar ve bulut-tabanlı işlem gücünün azaltılmasına katkıda bulunur.

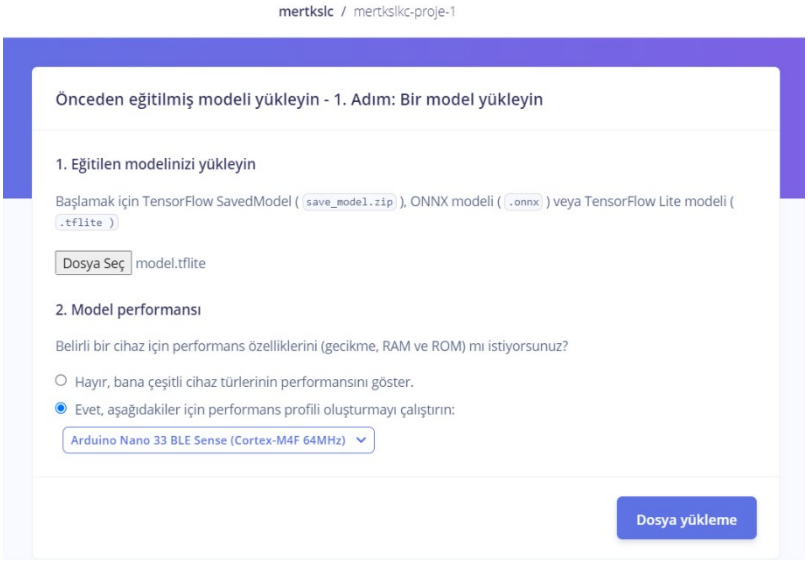
### 4.5.Model Dağıtımı

OTA, kablosuz iletişim teknolojileri aracılığıyla cihazlara güncellemeleri, yazılım veya firmware'i dağıtmak için kullanılan bir yöntemdir. OTA güncellemeleri, cihazların işletim sistemlerini, uygulamalarını veya firmware'lerini güncellemek için geleneksel kablo bağlantılarına veya fiziksel erişime ihtiyaç duymadan yapılabilir.OTA özellikle büyük IoT (Nesnelerin İnterneti) cihaz ağları veya dağıtılmış gömülü sistemler gibi durumlarda, cihazların yönetimini ve güncellemesini büyük ölçüde kolaylaştırır. Bu, cihazların daha güvenli, güncel ve etkin bir şekilde çalışmasını sağlar. Bu kapsamda TinyML kartına yapay zeka modelinin uzaktan aktarılabilmesi ve daha sonra geliştirilen yeni modellerin dağıtılabilmesi amacıyla OTA kullanılmıştır. TinyML kartında OTA'yı kullanmak ile için Arduino IDE üzerinden OTA kodları yüklenmiştir (Şekil 4.7).

```
sketch_sep11a §
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

const char* ssid = "SSID"; // WiFi SSID
const char* password = "Şifre"; // WiFi şifre

void setup() {
  Serial.begin(115200);
  Serial.println("Booting");

  WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }

  // OTA güncellemesi için hazırlık
  ArduinoOTA.begin();
  ArduinoOTA.setHostname("ESP32OTA"); // ESP32'nin adını belirleyebilirsiniz
  ArduinoOTA.onStart([]() {
    Serial.println("OTA güncelleme başladı");
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("\nOTA güncelleme tamamlandı");
  });
```

Şekil 4. 8 OTA Yükleme

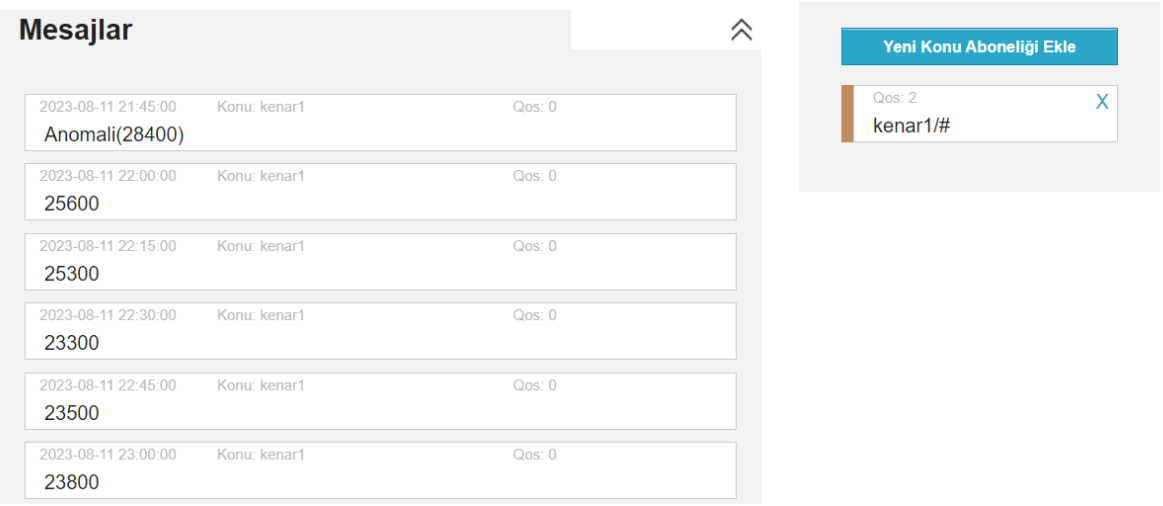
OTA kodları yüklenen TinyML kartına dönüşümü yapılan modelin uzaktan dağıtılması amacıyla Edge Impulse sitesi kullanılmıştır (Şekil 4.8). Edge Impulse, nesnelerin interneti (IoT) ve gömülü sistemler için özel olarak tasarlanmış bir platformdur. Bu platform, düşük güçlü cihazlarda (örneğin, mikrodenetleyicilerde) çalışabilen yapay zeka (AI) ve makine öğrenme (ML) modellerinin geliştirilmesini, eğitilmesini ve dağıtılmasını kolaylaştıran bir dizi araç ve hizmet sunmaktadır.




Şekil 4. 7 Model Dağıtımı

### 4.6. Uç Nokta LSTM Modeli

Uç noktada çalışan TinyML kart, sunucudan gelen enerji değerlerini API kullanarak düzenli aralıklarla 15 dakika gibi belirli zaman dilimlerinde ölçmektedir. Bu ölçümler, enerji tüketimi veya üretimi gibi kritik verilerin gerçek zamanlı olarak izlenmesini sağlar. TinyML kartı, bu ölçümler üzerinden yapay zeka (AI) ve makine öğrenme (ML) modeli kullanarak zamanla tahminlerde bulunur. Bu tahminler, sunucudaki gerçek enerji değerlerini modelin öngörülerinden ayırarak analiz edilir. Eğer modelin tahminleri ile gerçek değerler arasında belirgin bir fark tespit edilirse, sistem olası bir anomaliyi veya olağandışı bir durumu işaret ederek bir uyarı mesajı üretir (Şekil 4.9). Sistem, bu tür anormallikleri tespit ettiğinde üretilen




Şekil 4. 9 MQTT ile Alınan Değerler

uyarı mesajı MQTT sunucusu aracılığıyla bulut sunucusunda bulunan bir veritabanına iletir. Bu veritabanı, enerji yönetimi, güvenlik ve verimlilik gibi alanlarda verilerin izlenmesi ve analiz edilmesi için kullanılır.Bu süreç, enerji yönetimi, güvenlik veya verimlilik gibi alanlarda önemli bir rol oynar. Sunucudan gelen enerji verilerinin yanı sıra TinyML kartının tahmin yetenekleri, enerji tüketimini optimize etmek, kaynakları etkin bir şekilde kullanmak ve enerji kesintilerini önceden tahmin etmek gibi işlevleri yerine getirebilir.

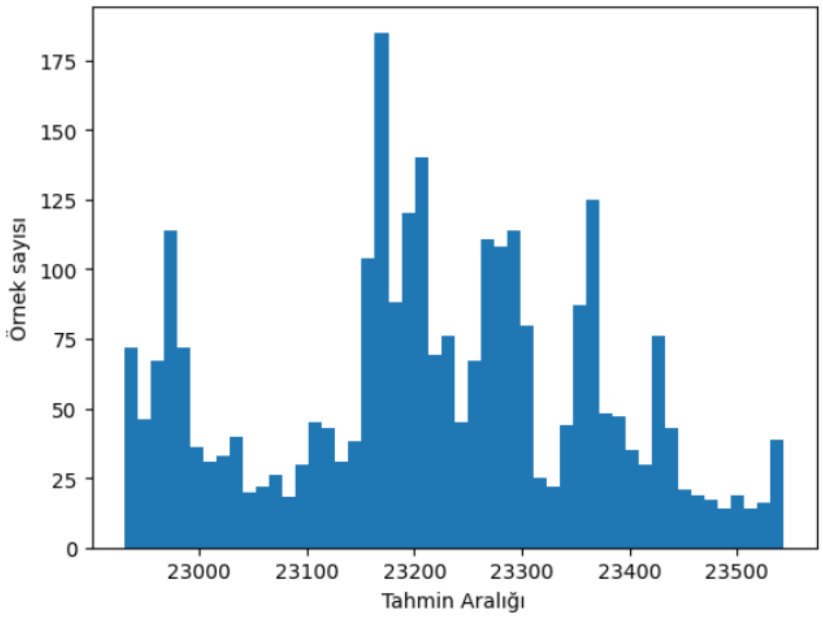
### 4.7.Anomali Tespiti

Güven aralığı değeri, bir sistemin veya sürecin belirli bir durumu veya olayı tespit etmek veya belirlemek için kullanılan bir referans değeridir. Güven aralığı, genellikle bir karar verme noktası olarak işlev görür ve bu değer üzerinde veya altında gerçekleşen olaylar veya veriler, belirli bir eylemi tetikleyebilir. Anomali tespitinde, Güven aralığı normalden sapmış veya olağandışı verileri belirlemek için kullanılır. Eğitilmiş bir modeli kullanarak eğitim verileri olan

X_train üzerinde tahminlerde bulunulur. Bu tahminler, modelin girdiyi nasıl yeniden oluşturduğunu temsil eder. Tahmin edilen veriler ile gerçek eğitim verileri arasındaki farkın mutlak değerini hesaplanır. Bu, her bir öğe (veri noktası) için bir hata değeri üretilir. Her öğenin hata değerlerini kullanarak ortalamayı hesaplar ve bu veri noktası için ortalama güven aralığı değerini elde etmeyi sağlar (Şekil 4.10). Sonuç olarak, train_mae_loss adlı değişken, her eğitim

```
X_train_pred = model.predict(X_train, verbose=0)
train_mae_loss = np.mean(np.abs(X_train_pred - X_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel('Tahmin Aralığı')
plt.ylabel('Örnek sayısı');

threshold = np.max(train_mae_loss)
print(f'Güven Aralığı: {threshold}')
```

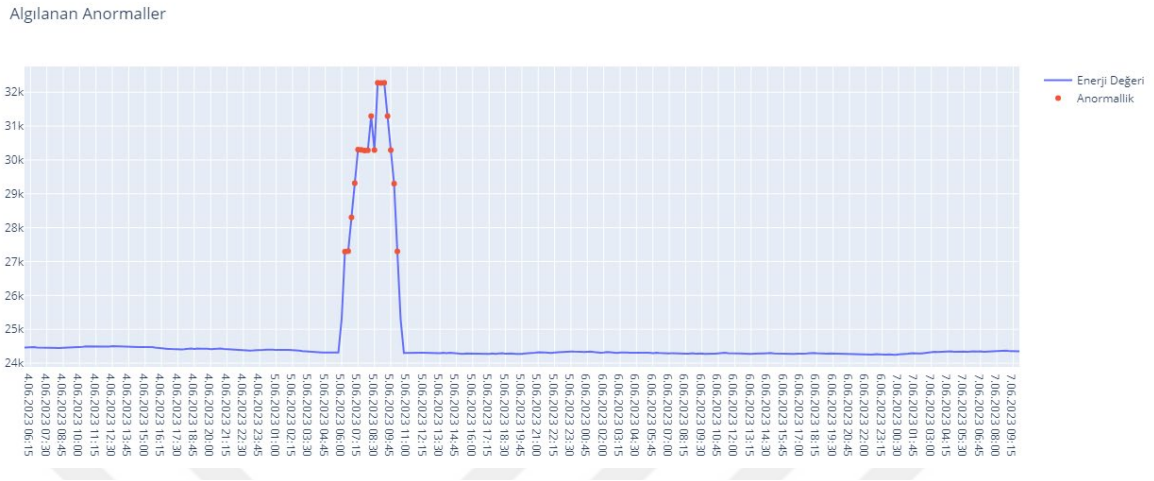
```
Güven Aralığı: 25543.174153645832
```




Şekil 4. 10 Anomali Tespiti

verisi noktası için ortalama mutlak hata değerlerini içeren bir dizi olur. Bu değerler, modelin eğitim verilerini ne kadar iyi yeniden oluşturduğunu ölçer. Daha düşük bir MAE, modelin daha iyi performans gösterdiğini gösterir. Bu ölçüm, modelin eğitimi sırasında performansı izlemek ve gerekirse modelin güven aralığını ayarlamak için kullanılabilir. Şekil 4.10 incelendiğinde, güven aralığı 25,543 olarak belirlenmiştir. Bu değerin üzerinde bir enerji artışı, anormallik olarak kabul edilecektir. İncelemeler sonucunda 5.06.2023 tarihinde saat 06:00 ile 5.06.2023 tarihinde saat 11:00 arasında sunucudaki enerji seviyesinin arttığı ve 32,000 seviyelerine kadar

yükseldiği görülmüştür. Bu dönemde gelen veriler, model tarafından anormal olarak algılanmış ve etiketlenmiştir (Şekil 4.11).




Şekil 4. 11 Anomali Grafiği

Bu tespit, modelin sistemin çalışma düzeni üzerindeki değişiklikleri ve belirgin enerji artışlarını başarıyla tespit ettiğini göstermektedir. Anormal durumlar, enerji yönetimi veya güvenlik gibi önemli uygulama alanlarında erken uyarılar sağlayabilir ve gerektiğinde müdahale edilmesine yardımcı olabilir. Bu tür tespitler, endüstriyel süreçlerin izlenmesi, sorunların hızlıca çözülmesi ve kaynakların verimli bir şekilde kullanılması için büyük bir öneme sahiptir.

# 5. SONUÇ ve ÖNERİLER

Bu çalışma, LSTM tabanlı arıza tespiti yöntemlerinin uygulanması ve sonuçlarının değerlendirilmesini amaçlamaktadır. Geliştirilen LSTM modeli, mevcut veri seti üzerinden sunucuda yaşanan enerji dalgalanmalarını başarılı bir şekilde öğrenmiş ve arıza tespiti konusunda son derece etkili sonuçlar elde etmiştir. Model, zaman serilerini analiz etme yeteneği sayesinde, dönem dönem değişen iş yüklerine göre verilerin içerdiği karmaşıklığı başarılı bir şekilde ele almıştır. Bu sayede, modelin arıza tespiti doğruluk oranları tatmin edici düzeydedir. Yapılan çalışma sonucunda, anormallikler algılandığı süre zarfında sunucuda RAM arızası meydana gelmiş ve bu nedenle enerji tüketimi pik yapmıştır. Elde edilen sonuçlar, enerji dalgalanmalarının sunucu performansına olan etkisini açıkça göstermektedir ve bu tür arızaların önceden tespitinin kritik öneme sahip olduğunu vurgulamaktadır. Ayrıca, LSTM modelinin arızaları erken uyarabilme yeteneği, işletme sürekliliğini artırmada önemli bir rol oynar. Bu durum, bakım süreçlerini optimize etmeye ve işletme maliyetlerini düşürmeye yardımcı olabilir. Modelin güncel kalmasını sağlamak ve yeni veriye uyum sağlamak için belirli periyotlarla eğitilmesi gereklidir. Ayrıca, modelin güncellenmiş sürümlerinin MLOPS (Makine Öğrenimi İşlemleri) metodolojisi kullanılarak uç noktalara başarıyla dağıtılması kolaylaştırılmıştır. Bu, modelin sürekli iyileştirilmesi ve operasyonel kullanımda etkin bir şekilde sürdürülmesi için kritik bir adımdır.

Sonuç olarak, LSTM tabanlı arıza tespiti yöntemi, endüstriyel tesislerdeki arızaların önceden tahmin edilmesini sağlayarak işletme sürekliliğini artırma konusunda etkili bir araçtır. Bu teknoloji, veri analizi ve makine öğrenme alanındaki ilerlemelerle birlikte endüstriyel işletmelerin daha verimli ve güvenilir bir şekilde çalışmasına katkı sağlamaya devam edecektir. Ancak, LSTM tabanlı arıza tespiti modeli başarılı sonuçlar üretmiş olsa da, sürekli bir iyileştirme süreci gerektirmektedir. Veri kalitesi, hiperparametre ayarları, güncel veri kullanımı ve işbirliği gibi faktörler, bu alanda daha iyi sonuçlar elde etmek için dikkate alınmalıdır. Bu öneriler, arıza tespiti süreçlerini optimize etmek ve işletme sürekliliğini sağlamak için kritik öneme sahiptir. Günümüzde, endüstriyel tesislerde veri toplama ve analiz teknolojileri hızla gelişmektedir. Bu gelişmeler, tesislerin daha verimli ve güvenilir bir şekilde çalışmasına yardımcı olmaktadır. Özellikle, LSTM (Uzun Kısa Süreli Bellek) tabanlı arıza tespiti, endüstriyel sorunları önceden tahmin etme konusunda büyük bir öneme sahiptir.

## 6. KAYNAKÇA

(1) Check Point. (y.y.). Tarihinde 04 Ekim 2022, adresinden erişildi https://www.checkpoint.com/cyber-hub/network-security/what-is-iot/what-is-an-iot-gateway/

Chin, A., Wolf, P., & Tian, J. (2019). A cloud IoT edge framework for efficient data-driven automotive diagnostics. IEEE Vehicular Technology Conference, 2019-September. https://doi.org/10.1109/VTCFALL.2019.8891492

Deorankar, A. V., & Thakare, S. S. (2020). Survey on Anomaly Detection of (IoT)- Internet of Things Cyberattacks Using Machine Learning. Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC 2020, 115–117. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-00023

Hsieh, R. J., Chou, J., & Ho, C. H. (2019). Unsupervised online anomaly detection on multivariate sensing time series data for smart manufacturing. Proceedings - 2019 IEEE 12th Conference on Service-Oriented Computing and Applications, SOCA 2019, 90–97. https://doi.org/10.1109/SOCA.2019.00021

Huang, Q., Kang, Z., Zhang, Y., & Yan, D. (2020). Tool remaining useful life prediction based on edge data processing and LSTM recurrent neural network. Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM, 2020-June. https://doi.org/10.1109/ICPHM49022.2020.9187037

IoT nedir? - Yeni Başlayanlar İçin Nesnelerin İnterneti Kılavuzu - AWS. (y.y.). Tarihinde 04 Ekim 2022, adresinden erişildi https://aws.amazon.com/tr/what-is/iot/

Kars, B. N. (2021). Edge Computing Security with an IoT device. 1(June), 14–17.

Kayode, O., & Tosun, A. S. (2019). LIRUL: A Lightweight LSTM based model for Remaining Useful Life Estimation at the Edge. Proceedings - International Computer Software and Applications Conference, 2, 177–182. https://doi.org/10.1109/COMPSAC.2019.10203

Lee, S. H., Lee, T., Kim, S., & Park, S. (2019). Energy consumption prediction system based on deep learning with edge computing. 2019 2nd International Conference on Electronics Technology, ICET 2019, 473–477.

https://doi.org/10.1109/ELTECH.2019.8839589

Liu, Y., Garg, S., Nie, J., Zhang, Y., Xiong, Z., Kang, J., & Hossain, M. S. (2021). Deep Anomaly Detection for Time-Series Data in Industrial IoT: A Communication-Efficient On-Device Federated Learning Approach. IEEE Internet of Things Journal, 8(8), 6348–6358. https://doi.org/10.1109/JIOT.2020.3011726

Liu, Y., Kumar, N., Xiong, Z., Lim, W. Y. B., Kang, J., & Niyato, D. (2020). Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things. 2020 IEEE Global Communications Conference, GLOBECOM 2020 - Proceedings, 2020-January. https://doi.org/10.1109/GLOBECOM42002.2020.9348249

Lu, S., Tang, X., Zhu, Y., & She, J. (2021). A Cloud-Edge Collaborative Intelligent Fault Diagnosis Method Based on LSTM-VAE Hybrid Model. Proceedings - 2021 8th IEEE International Conference on Cyber Security and Cloud Computing and 2021 7th IEEE International Conference on Edge Computing and Scalable Cloud, CSCloud-EdgeCom 2021, 207–212. https://doi.org/10.1109/CSCLOUD-EDGECOM52276.2021.00045

Ou, C. H., Chen, Y. A., Huang, T. W., & Huang, N. F. (2020). Design and Implementation of Anomaly Condition Detection in Agricultural IoT Platform System. International Conference on Information Networking, 2020-January, 184–189. https://doi.org/10.1109/ICOIN48656.2020.9016618

Oyekanlu, E. (2018). Distributed Osmotic Computing Approach to Implementation of Explainable Predictive Deep Learning at Industrial IoT Network Edges with Real-Time Adaptive Wavelet Graphs. Proceedings - 2018 1st IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018, 179–188. https://doi.org/10.1109/AIKE.2018.00042

Teh, H. Y., Wang, K. I. K., & Kempa-Liehr, A. W. (2021). Expect the Unexpected: Unsupervised Feature Selection for Automated Sensor Anomaly Detection. IEEE Sensors Journal, 21(16), 18033–18046. https://doi.org/10.1109/JSEN.2021.3084970

Verma, A., Singh, A., Anand, D., Aljahdali, H. M., Alsubhi, K., & Khan, B. (2021). IoT Inspired Intelligent Monitoring and Reporting Framework for Education 4.0. IEEE Access, 9, 131286–131305. https://doi.org/10.1109/ACCESS.2021.3114286

What is IoT? The internet of things explained | Network World. (y.y.). Tarihinde 04 Ekim

2022, adresinden erişildi https://www.networkworld.com/article/3207535/what-is-iot-the-internet-of-things-explained.html

What Is the Internet of Things (IoT)? (y.y.). Tarihinde 04 Ekim 2022, adresinden erişildi https://www.oracle.com/internet-of-things/what-is-iot/

OASIS Standard. (2014). MQTT Version 3.1.1. [Erişim Tarihi: 2023, 6 Mayıs]. URL: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

HiveMQ. (Y.y.). MQTT Essentials - A Lightweight IoT Protocol. [Erişim Tarihi: 2023, 6 Mayıs]. URL: https://www.hivemq.com/mqtt-essentials/

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for VM-based cloudlets in mobile computing. IEEE pervasive computing, 8(4), 14-23.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: vision and challenges. IEEE Internet of Things Journal, 3(5), 637-646.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. Communications of the ACM, 53(6), 50-58.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50

Satyanarayanan, M. (2017). The Emergence of Edge Computing. Computer, 50(1), 30-39.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

Geoffrey Hinton, et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups." IEEE Signal Processing Magazine (2012).

Mitchell, T. (1997). Machine Learning. McGraw-Hill Education.

Alpaydin, E. (2020). Introduction to Machine Learning (3rd ed.). MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. MIT Press.

Marsland, S. (2014). Machine Learning: An Algorithmic Perspective. CRC Press.

Rakhlin, A., & Sridharan, K. (2014). Statistical Learning Theory: A Primer. Chapman and Hall/CRC.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.

Graves, A. (2012). Supervised sequence labelling with recurrent neural networks. Springer Science & Business Media.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 15.

Hawkins, S., He, H., Williams, G., & Baxter, R. (2002). Outlier detection using replicator neural networks. In Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on (pp. 309-316). IEEE.

Aggarwal, C. C. (2013). Outlier analysis. Springer Science & Business Media.

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A review. arXiv preprint arXiv:1901.03407.

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management, 35(2), 137-144.

**EKLER**

**EK-1 Uç Bilişim ile Lstm Tabanli Arıza Tespit Kodu**

```
from tensorflow import keras
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import numpy as np
from numpy.ma.core import size
import tensorflow as tf
import pandas as pd
import plotly.graph_objects as go
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, RepeatVector, TimeDistributed
print('Tensorflow version: ', tf.__version__)
df = pd.read_csv('dataset.csv',sep=';')
df.head()
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['zaman'], y=df['enerji'], name='W'))
fig.update_layout(showlegend=True, title='Enerji Kullanımına Ait Genel Grafik')
fig.show()
plt.figure(figsize=(8,8))
plt.hist(df['enerji'])
plt.title("CPU Değer Histogramı")
plt.show()
```

```
train, test= train_test_split(df, test_size=0.20, shuffle=False)
print(f" Eğitim veri sayısı = {len(train)}", "\n", f"Test veri sayısı {len(test)}")
train.shape,test.shape
#scaler = StandardScaler()
#scaler = scaler.fit(train[['cpu_usage']])
#train['cpu_usage'] = scaler.transform(train[['cpu_usage']])
#test['cpu_usage'] = scaler.transform(test[['cpu_usage']])
TIME_STEPS=48
def create_sequences(X, y, time_steps=TIME_STEPS):
    Xs, ys = [], []
    for i in range(len(X)-time_steps):
        Xs.append(X.iloc[i:(i+time_steps)].values)
        ys.append(y.iloc[i+time_steps])
    return np.array(Xs), np.array(ys)
X_train, y_train = create_sequences(train[['enerji']], train['enerji'])
X_test, y_test = create_sequences(test[['enerji']], test['enerji'])
print(f'Training shape: {X_train.shape}')
print(f'Testing shape: {X_test.shape}')
model = Sequential()
model.add(LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(rate=0.1))
model.add(RepeatVector(X_train.shape[1]))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(rate=0.1))
```

```
model.add(TimeDistributed(Dense(X_train.shape[2])))

model.compile(optimizer='adam', loss='mae')

model.summary()

history = model.fit(X_train, y_train, epochs=500, batch_size=128, validation_split=0.1,

                    callbacks=[keras.callbacks.EarlyStopping(monitor='val_loss', patience=3,
mode='min')], shuffle=False)

converter = tf.lite.TFLiteConverter.from_keras_model(model)

converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS,
tf.lite.OpsSet.SELECT_TF_OPS]

tflite_model = converter.convert()

with open('model.tflite', 'wb') as f:

    f.write(tflite_model)

epochs = range(1, len(train_loss) + 1)

# Eğitim loss grafiği

plt.plot(epochs, train_loss, 'b', label='Eğitim Kaybı')

# Doğrulama loss grafiği

plt.plot(epochs, val_loss, 'r', label='Doğrulama Kaybı')

# Grafik başlığı ve etiketler

plt.title('Eğitim ve Doğrulama Loss Grafiği')

plt.xlabel('Adım')

plt.ylabel('Kayıp Değeri')

plt.legend()

# Grafiği göster

plt.show()
```

```
plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
plt.legend();
model.evaluate(X_test, y_test)
X_train_pred = model.predict(X_train, verbose=0)
train_mae_loss = np.mean(np.abs(X_train_pred - X_train), axis=1)
plt.hist(train_mae_loss, bins=50)
plt.xlabel('Tahmin Aralığı')
plt.ylabel('Örnek sayısı');
threshold = np.max(train_mae_loss)
print(f'Güven Aralığı: {threshold}')
X_test_pred = model.predict(X_test, verbose=0)
test_mae_loss = np.mean(np.abs(X_test_pred-X_test), axis=1)
plt.hist(test_mae_loss, bins=50)
plt.xlabel('Güven Aralığı')
plt.ylabel('Örnek sayısı')
print(f'Reconstruction error threshold: {threshold}')
test_score_df = pd.DataFrame(test[TIME_STEPS:])
test_score_df['loss'] = test_mae_loss
test_score_df['threshold'] = threshold
test_score_df['anomaly'] = test_score_df['enerji'] > test_score_df['threshold']
test_score_df['enerji'] = test[TIME_STEPS:]['enerji']
test_score_df
fig = go.Figure()
```

```
fig.add_trace(go.Scatter(x=test_score_df['zaman'], y=test_score_df['enerji'], name='Test loss'))

fig.add_trace(go.Scatter(x=test_score_df['zaman'], y=test_score_df['threshold'], name='Threshold'))

fig.update_layout(showlegend=True, title='Test loss vs. Threshold')

fig.show()

anomalies = test_score_df[test_score_df['anomaly'] == True]

anomalies.head()

fig = go.Figure()

fig.add_trace(go.Scatter(x=test_score_df['zaman'], y=(test_score_df['enerji']), name='Enerji Değeri'))

fig.add_trace(go.Scatter(x=anomalies['zaman'], y=(anomalies['enerji']), mode='markers', name='Anormallik'))

fig.update_layout(showlegend=True, title='Algılanan Anormaller')

fig.show()
```