KATEGORİK DEĞİŞKEN İÇİN MAKİNE ÖĞRENMESİ ALGORİTMALARININ PERFORMANSLARININ İNCELENMESİ

Yüksek Lisans Tezi

İlkay TUĞ

Eskişehir 2023

KATEGORİK DEĞİŞKEN İÇİN MAKİNE ÖĞRENMESİ ALGORİTMALARININ PERFORMANSLARININ İNCELENMESİ

İlkay TUĞ

YÜKSEK LİSANS TEZİ

Uygulamalı İstatistik Tezli Yüksek Lisans Programı
İstatistik Anabilim Dalı
Danışman: Prof. Dr. Betül KAN KILINÇ

Eskişehir
Eskişehir Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Nisan 2023

JÜRİ VE ENSTİTÜ ONAYI

İlkay TUĞ'un "Kategorik Değişken için Makine Öğrenmesi Algoritmalarının Performanslarının İncelenmesi" başlıklı tezi 25/04/2023 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Eskişehir Teknik Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği"nin ilgili maddeleri uyarınca, İstatistik Anabilim dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Jüri Üyeleri	**Unvanı Adı Soyadı**	**İmza**
Üye	: Prof. Dr. Betül Kan Kılınç	.....
Üye	: Prof. Dr. Barış Aşıkgil	.....
Üye	: Dr. Öğr. Üyesi İlknur Atasever Güvenç	.....

Prof. Dr. Murat TANIŞLI
Lisansüstü Eğitim Enstitüsü Müdürü

25/04/2023

DANIŞMAN ONAYI

Danışmanlığını yürüttüğüm yüksek lisans öğrencisi İlkay TUĞ, "Kategorik Değişken için Makine Öğrenmesi Algoritmalarının Performanslarının İncelenmesi " başlıklı tez çalışmasını tamamlamıştır. Hazırlamış olduğu tez tarafımca incelenmiş ve öğrencinin tez savunma sınavına alınması bilimsel ve etik açıdan uygun görülmüştür.

Tez Danışmanı
Prof. Dr. Betül KAN KILINÇ

ÖZET

KATEGORİK DEĞİŞKEN İÇİN MAKİNE ÖĞRENMESİ
ALGORİTMALARININ PERFORMANSLARININ İNCELENMESİ

İlkay TUĞ
İstatistik Anabilim Dalı
Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Nisan, 2023

Danışman : Prof. Dr. Betül KAN KILINÇ

Makine öğrenmesi yöntemleri ve buna bağlı çalışmalar son yıllarda yaygın bir şekilde kullanım alanı bulmuş buna bağlı olarak gelişime açık bir hale gelmiştir. Bu tez çalışmasında, sınıflandırma problemlerinde karşılaşılan dengesizlik durumları incelenecektir. Buna göre, ikili sınıf dengesizliği problemi ele alınarak farklı oranlarda dengesizlik içeren sınıfları bir benzetim çalışmasıyla, sınıf dengesizliği problemini çözmek için çeşitli yeniden örnekleme yöntemleri kullanılarak, boosting topluluk öğrenmesi yöntemlerinden Adaptive boosting, Gradient Boosting Machines ve Extreme Gradient Boosting yöntemlerinin başarı performansları ve süre karşılaştırılması yapılacaktır.

Anahtar Sözcükler: Kategorik, Boosting, Karmaşıklık matrisi, Makine öğrenmesi

ABSTRACT

COMPARISON OF THE PERFORMANCE OF MACHINE LEARNING METHODS FOR CATEGORICAL VARIABLE

İlkay TUĞ

Department of Statistics

Eskişehir Technical University, Institute of Graduate Programs, April, 2023

Supervisor : Prof. Dr. Betül KAN KILINÇ

In recent years, machine learning algorithms and related studies have become widespread and depending on that its development is being evolved in time. In this thesis, imbalanced data problems in classification studies have been investigated. By conducting a simulation study, the ensemble learning methods such as Adaptive Boosting, Gradient Boosting Machines and Extreme Gradient Boosting were applied to the data sets having imbalanced problems to compare the performance and processing time of those with various ratios of imbalance scenarios using different sampling methods.

Keywords: Categorical, Boosting, Confusion matrix, Machine learning

TEŞEKKÜR

Yüksek Lisans öğrenim sürecimde deneyim ve bilgisiyle beni yönlendiren, anlayışını, sabrını ve desteğini esirgemeyen, daima yoluma ışık tutan saygı değer danışman hocam Prof. Dr. Betül KAN KILINÇ' a, her zaman yanımda olup, beni hep destekleyen değerli annem İlknur TUĞ ve babam Muharrem TUĞ' a, daima yüzümü güldüren canım kardeşim Berkay TUĞ'a, bana hep inanan ve ileriye dönük teşvik eden biricik babaannem Kadriye TUĞ' a ve bu süreçte beni yalnız bırakmayan arkadaşım Mazlum Davut ÇELİK' e sevgi, saygı ve teşekkürlerimi sunuyorum.

İlkay TUĞ

Nisan 2023

25/04/2023

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığım; bu çalışma kapsamında elde edilemeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Eskişehir Teknik Üniversitesi tarafından kullanılan "bilimsel intihal tespit programı"yla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

İlkay TUĞ

İÇİNDEKİLER

Sayfa

Sayfa

TABLOLAR DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

AdaBoost : Adaptive Boosting (Uyarlamalı Arttırma)

ADASYN : Adaptive Synthetic Sampling Method (Uyarlanabilir Sentetik Örnekleme)

ANNs : Artifical Neural Networks (Yapay Sinir Ağları)

AUC : Area Under the ROC Curve (ROC Eğrisinin Altındaki Alan)

Bagging : Bootstrap Aggregating (Torbalama)

Boosting : Arttırma

CART : Classification and Regression Trees (Sınıflandırma ve Regresyon Ağaçları)

CatBoost : Categorical Boosting (Kategorik Artırma)

CHAID : ChiSquared Automatic Interaction Detector (Ki Kare Otomatik Etkileşim Dedektörü)

DT : Decision Tree (Karar Ağacı)

DP : Doğru Pozitif

DN : Doğru Negatif

GBM : Gradient Boosting Machine (Gradyan Arttırma)

KNN : K-Nearest Neighbor (K-En Yakın Komşuluk)

RF : Random Forest (Rastgele Orman)

ROC : Receiver Operating Characteristic (Algılayıcı İşletim Eğrisi)

ROS : Random Over-Sampling (Rastgele Aşırı Örnekleme Yöntemi)

ROSE : Random Over-Sampling Examples (Rastgele Aşırı Örnekleme Örnekleri Yöntemi)

RUS : Random Under-Sampling (Rastgele Alt Örnekleme Yöntemi)

LightGBM : Light Gradient Boosting Machines (Hafif Gradyan Arttırma Makineleri)

LR : Logistic Regression (Lojistik Regresyon)

SMOTE : Synthetic Minority Over-Sampling Technique (Sentetik Azınlık Aşırı Örnekleme Tekniği)

SMOTEBoost	: SMOTE with Boosting
SVM	: Support Vector Machines (Destek Vektör Makineleri)
YP	: Yanlış Pozitif
YN	: Yanlış Negatif
XGBoost	: Extreme Gradient Boosting (Aşırı Gradyan Arttırma)

1. GİRİŞ

Son yıllarda büyük veri araştırmaları öne plana çıkmış ve beraberinde bazı durumları getirmiştir. Örneğin, gerçek hayatta veriler her zaman eksiksiz bulunmayabilir. Kayıp veya eksik değerleri olan verilerde dengesizlik veya doğası gereği bazı sınıf dengesizliği durumları gözlemlenebilir. Dolayısıyla büyük veri ile dengesizlik problemleri daha fazla önem kazanmış, bununla başa çıkmak için bir takım yöntemler geliştirilmiştir.

Dengesizlik sorununa çözüm olması açısından, veriyi dengeli hale getirmek düşünülebilir. Bunun için literatürde bazı yöntemler geliştirilmiştir [1–5]. Bu yöntemler, i) yeniden örnekleme, ii) maliyet duyarlı öğrenme ve iii) hibrit olmak üzere dengesizlik sorununa çözüm aramaktadırlar [6].

Yeniden örnekleme, veri dağılımını değiştirerek yeni gözlemler üretmek veya eksiltmek üzerine kuruludur. Bu yöntemler genellikle üç başlık altında incelenmektedir. Bunlardan ilki "Rastgele Aşırı Örnekleme (Random Over-Sampling-ROS)" dir. Bu yöntemde, eğitim verisinden azınlık sınıfında olan gözlemler yeniden rastgele örneklenerek gözlem sayısı arttırılmaya çalışılır. Diğeri "Rastgele Alt Örnekleme (Random Under-Sampling-RUS" dir. Bu yöntemde ise eğitim verisinden çoğunluk sınıfında olan gözlemler rastgele çıkartılır. Son yaklaşım ise "Hibrit Yöntem"' dir. Bu yöntem hem aşırı örneklemeyi hem de alt örneklemeyi birleştiren bir yöntemdir [7,8].

ROS yönteminin aşırı uyum gibi dezavantajları olabilmektedir bundan dolayı literatürde bazı çalışmalar geliştirilmiştir. Bu çalışmalardan biri de, Chawla vd., (2002) aittir. Sentetik Azınlık Aşırı Örnekleme Yöntemi (Synthetic Minority Over-Sampling Technique - SMOTE) yaklaşımını önermişlerdir. Bu yöntem, dengesiz verilerde aşırı örnekleme yaklaşımını kullanarak, azınlık sınıfı gözlemlerini kopyalar ve yerine “sentetik” gözlemler üretmektedir. Sentetik gözlemler, orijinal veri setlerinde komşuluk dikkate alınarak yapay olarak üretilir, orijinal verinin dağılımını yansıtır [1].

Aydın (2018) iki sınıflı dengesizlik problemleri için simülasyon ile veriler üreterek farklı üç etkinin bulunduğu korelasyon yapısı, gözlem sayısı ve dengesizlik oranlarını farklılaştırılarak 80 farklı senaryo için, Destek Vektör Makineleri (Support Vector Machines-SVM), Sınıflama ve Regresyon Ağaçları (Classification and

Regression Trees-CART) ve Rastgele Orman (Random Forest-RF) algoritmaları ile SMOTE, SMOTEBoost, RUS ve boosting algoritmalarının birleştirilmesi ile oluşan RUSBoost, MW-MOTE, Easy Ensemble, SMOTEBagging ve UnderBagging örnekleme yöntemlerini karşılaştırmış ve sonuçları yorumlanmıştır. Verideki bu değişimlere bağlı olarak kullanılan yöntemlerin performans değerleri farklılık göstermiştir. RUSBoost yönteminin üç korelasyon düzeyinde de etkili olduğu görülmüştür. Buna istinaden açıklanan kriterlerin dikkate alınması gerektiği belirtilmiş ayrıca dengesizlik durumu (%30'un altında) olduğunda çözüm algoritmalarının kullanılmasının sınıflama performansları üzerinde etkisinin olduğu görülmüştür [9].

Brown ve Mues (2012) çalışmalarında, gerçek hayat verileriyle çalışma yapmışlardır. Artan sınıf dengesizlikleriyle yapılan çalışmada çeşitli sınıflandırma yöntemleri karşılaştırılmıştır. Burada, sınıflama yöntemlerinin AUC performans değerleri karşılaştırılmış ve bazı sınıf dengesizliği düzeylerinde Gradyan Arttırma (Gradient Boosting-GBM) ve RF algoritmalarının genel olarak daha iyi performans gösterdiği gösterilmiştir [10].

Diğer çalışmada, Bach vd.' nin (2017) çalışmasında, osteoporozla ilgili tıbbi bir veri setini ele alınarak birkaç farklı makine öğrenmesi yöntemi ile yeniden örnekleme çalışılmıştır. Buna göre RF algoritması ile beraber kullanılan SMOTE yönteminin daha iyi sonuçlar verdiği görülmüştür [11].

Gameng vd., (2019) farklı 6 dengesiz veri seti kullanarak yaptıkları çalışmada, K-En Yakın Komşuluk (K-Nearest Neighbor, KNN) uzaklığını Manhattan uzaklığı ile değiştirmişlerdir. Adaptive Synthtetic (ADASYN) örnekleme yöntemi olmak üzere SMOTE ve modified ADASYN (Adaptive Synthetic Sampling Method) örnekleme yöntemleri kullanılarak doğruluk, kesinlik, duyarlılık ve F-ölçüsü değerlendirilmiştir. Modified ADASYN, öğrenmesi zor olan azınlık sınıfı gözlemlerine odaklanarak ağırlıklı bir dağılım kullanır ve bu gözlemlerden daha fazla üretilmesini sağlamaktadır. Ölçütler göz önüne alındığında Modified ADASYN yönteminin F-ölçütünün daha yüksek verdiği gözlemlenmiştir [12].

Bir diğeri Chakravarthy vd.' nin (2019) çalışmasında, Rastgele Aşırı Örnekleme Örnekleri (Random Over-Sampling Examples - ROSE) ve SMOTE yöntemleri kullanılarak farklı dengesizlikte olan verilere sınıflandırma yöntemleri uygulandığında karar ağacı tabanlı modellerin genel olarak ROSE ile daha iyi sonuçlar verdiği gö-

rülmüştür [13].

Par vd., (2019) çalışmasında, orijinal veri kümesi üzerinden belirlenen oranlarla dengesiz 3 farklı küçük veri seti üzerine Yapay Sinir Ağları (Artifical Neural Networks-ANNs), SVM, Naive Bayes (NB) ve Karar Ağacı (Decision Tree-DT) yöntemleri kullanılarak gözlemler sınıflandırılmıştır. Burada dengesizlik sorununun çözülmesi için 6 farklı yeniden örnekleme yöntemi kullanılmıştır. Kesinlik değerleri karşılaştırıldığında uzaklık tabanlı sınıf çoğaltılması yapıldığında dengesiz ve küçük veri kümelerinde ROSE ve SMOTE yöntemleri başarılı performanslarından dolayı önerilmiştir [14].

Koçaoğlu and Özcan (2018), UCI Machine Repository' den elde edilen veri setine yeniden örnekleme yöntemleri ROS, RUS, SMOTE ve ROSE kullanılarak tek gizli katmanlı öğrenme algoritması uygulamışlardır. ROS ve SMOTE yöntemleri kullanıldığında herhangi bir örnekleme yöntemi olmadan kurulan modele göre daha başarılı olmuşlardır. [15].

Demir and Şahin (2022), dengesiz iki sınıflı deprem verileri ile çalışmışlardır. Bu veri kümesine aşırı öğrenme, SMOTE, ROSE yeniden örnekleme yöntemleri ile SVM, RF ve NB makine öğrenmesi algoritmaları birlikte sırasıyla uygulanmıştır. Sonuç incelendiğinde, SMOTE ile birlikte kullanılan SVM yönteminin daha yüksek doğruluk değerine sahip olduğu görülmüştür [16].

Byeon' nın (2021) çalışmasında, dengesiz bir veri setine uygulanan Aşırı Gradyan Arttırma (Extreme Gradient Boosting-XGBoost), RF ve Lojistik Regresyon (Logistic Regression-LR) algoritmalarının performansları RUS, ROS ve SMOTE yöntemleri ile beraber değerlendirilmiştir. Sonuç olarak SMOTE tabanlı XGBoost modelinin duyarlılık ve seçicilik ölçütleriyle diğer yöntemlere göre daha iyi performans gösterdiği görülmüştür. Bu yüzden hastalık verileri gibi dengesiz veri setleri ile çalışıldığında SMOTE tabanlı XGBoost kullanılması önerilmiştir [17].

AL-Shatnwai and Faris (2020) çalışmalarında, telekomünikasyon şirketlerinde müşteri kaybı tahmini yapmışlardır. SVM, LR, RF ve XGBoost sınıflama yöntemleri arasından, XGBoost algoritmasının dengesiz veride daha iyi performans gösterdiği görülmüştür. Daha sonra XGBoost ile beraber uygulanan SMOTE, ADASYN ve Borderline SMOTE yöntemleri arasından SMOTE yöntemiyle beraber F-ölçütünün daha iyi sonuç verdiği görülmüştür [18].

Cahyana vd.' nin (2019) çalışmasında, GBM uyguladıkları dengesiz veriye daha sonra 4 farklı aşırı örnekleme yöntemi uygulamışlardır. Aşırı örnekleme uygulanarak sınıflandırılan verilerin örnekleme yapılmamış duruma göre performans ölçütlerinin arttığı gözlemlenmiştir [19].

Bentéjac, vd., (2020) RF, GBM, XGBoost, LightGBM (Light Gradient Boosting Machine), CatBoost (Categorical Boosting) için dengesiz veride hiper parametre çalışması yapmışlardır. Bir çok farklı değer üzerinde yapılan çalışmada optimum parametre değerleri belirlenmiştir. GBM ve XGBoost algoritmaların doğrulanmış değerleri varsayılan değerlere göre daha iyi sonuç verdiği görülmüştür. Yine hız açısından karşılaştırıldığında en hızlı LightGBM ve XGBoost olarak kaydedilmiştir [20].

Bir diğer çalışmada Mishra (2017), yeniden örnekleme yöntemleri, RUS, SMOTE ile XGBoost (Extreme Gradient Boosting, XGBoost) ve RF topluluk öğrenmesi yöntemi kullanılarak sınıflandırma yapmışlardır. ROC ölçüleri karşılaştırıldığında XGBoost algoritması RF algoritmasından daha iyi sonuçlar vermiştir. Ayrıca, RUS ile birlikte kullanılan XGBoost algoritmasının seçicilik ve duyarlılık arasında daha dengeli sonuçlar oluşturduğu ve RUS yönteminin ROC ölçüsünün, SMOTE yöntemine göre daha iyi performans gösterdiği kaydedilmiştir [21].

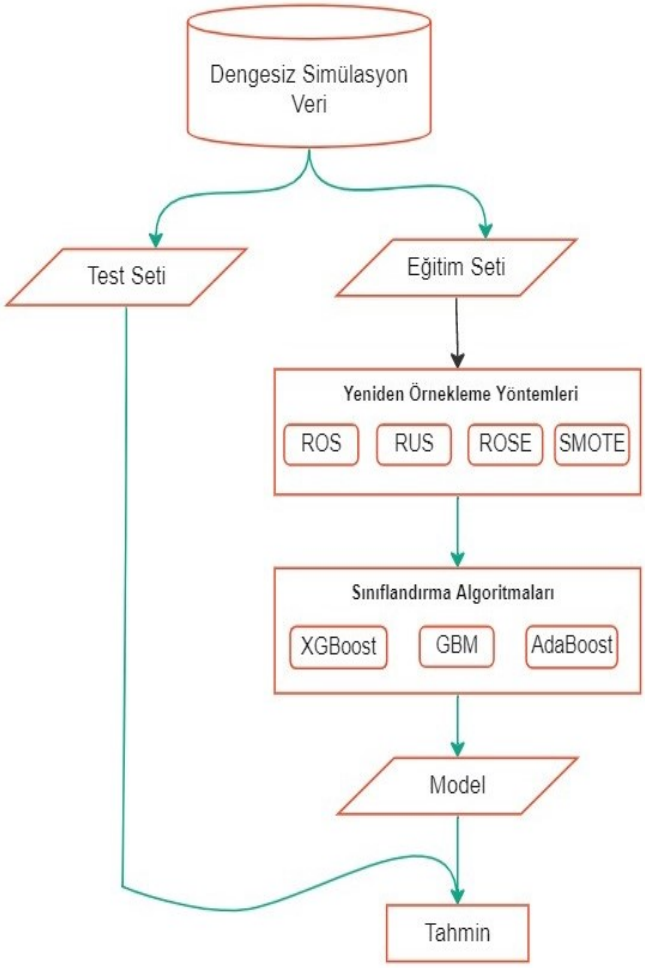
Mohammed vd., (2020) yaptıkları çalışmada veriye ROS ve RUS yöntemleri uyguladıktan sonra sınıflandırma algoritmaları uygulamışlardır. ROS yönteminin performansının daha başarılı olduğu görülmüştür [22].

Bir diğer çalışmada, Lenin ve Chandrasekaran (2021) örnekleme yöntemleri kullanılmadan dengesiz veriye C4.5 kazanç (gain) algoritması ve topluluk öğrenmesi algoritmalarından RF, Uyarlamalı Arttırma (Adaptive Boosting-AdaBoost), GBM ve XGBoost algoritmaları ile analizler yapmışlardır. Sonuç olarak, RF ve AdaBoost' un diğer algoritmalara göre daha iyi performans gösterdiği görülmüştür [23]. Sınıf dengesizliği sorunlarının çözümünde boosting kombinasyonları ile yeni yöntemler öneren çalışmalar da vardır [24–26].

Bu tez çalışmasının amacı, ikili sınıf dengesizliği problemlerini farklı dengesizlik oranlarını ve gözlem sayılarını ele alarak verileri simülasyon ile oluşturmayı, bu verileri (RUS, ROS, ROSE, SMOTE) yöntemleri ile dengeleyerek topluluk öğrenmesi algoritmalarından; AdaBoost, GBM ve XGBoost algoritmalarının başarı performanslarını gözlemlemeyi ve analiz sürelerini karşılaştırmayı hedeflemektedir.

2. YÖNTEMLER

Dengesiz sınıf verilerinde boosting topluluk öğrenmesi algoritmalarından AdaBoost, Gradient Boosting ve Extreme Gradient Boosting' nın performanları değişik senaryolar altında karşılaştırılacaktır. Buna ilişkin akış şeması Şekil 2.1' de gösterilmiştir.



Şekil 2.1. *Akış şeması*

Şekil 2.1' de görüldüğü gibi, simülasyon ile üretilen veri test ve eğitim olmak üzere ikiye ayrılır daha sonra eğitim verileri yeniden örnekleme yöntemleri ile sınıflandırma algoritmaları kullanılarak modellenir.

2.1. Sınıf Dengesizliği

Gerçekte, veri setinin sınıflar arası dağılımları her zaman eşit olmayabilir. Veriler, iki veya daha fazla sınıfa sahip olmakla beraber sınıf dağılımlarının eşit olmadığı durumlarda gözlem sayıları sınıflara göre farklılık gösterecektir. Azınlık sınıfı

daha değerli bilgiye sahip olabilir ve bu durum geleneksel makine öğrenmesi algoritmaları için zorlayıcı olacaktır. Geleneksel makine öğrenmesi algoritmaları genel olarak hatayı azaltmak ve doğruluğu arttırmak öngörüsüne bağlı olduğundan model performansı değerlendirilirken tahminlerin yanlı veya hatalı olmasına neden olabilmektedir. Çünkü tüm örnekler genellikle baskın sınıfa atanmakta ve azınlık sınıfı az örneklenmektedir [27].

Sınıf dengesizliği birçok araştırmacının dikkatini çekmiş ikili ve çoklu sınıflandırmada dengesizlik sorunun çözülebilmesi için birçok yöntem geliştirilmiştir. Bunlar, ilk olarak yeni gözlemler üretmek ve çıkarmak üzerine kurulu olan yeniden örnekleme yöntemidir. Bir diğeri algoritma düzeyinde olan yöntemler, mevcut veri madenciliği yöntemleri değiştirilerek ya da yeni algoritmalar üretilerek çalışır [28]. Son olarak hibrit yöntemler, uygun bir endüktif yanlılığı seçmek üzerine kurulu olan yöntemlerdir [27]. Genel olarak araştırmacılar yeniden örnekleme yöntemleri ve yeniden örnekleme yöntemlerinin kombinasyonları üzerine çalışmışlardır [28].

2.1.1. Yeniden örnekleme yöntemleri

Veri seviyesinde olan yeniden örnekleme yöntemleri, sınıf dengesizliği sorununu çözmek için dengeli sınıflar oluşturmayı amaçlarlar. Yeniden örnekleme yöntemleri: Aşırı Örnekleme (Over Sampling), Alt Örnekleme (Under Sampling) ve Hibrit yöntemler olarak üç grupta incelenebilir. Aşırı örneklemede azınlık sınıfına, çoğunluk sınıfının gözlem sayısıyla dengelenmesi için yeni gözlemler oluşturulur. Alt örneklemede ise tam tersi, çoğunluk sınıfında bulunan mevcut bazı gözlemler kaldırılır. Bu yöntemler, sınıf dengesini nispeten oluşturmak amacıyla verinin modifiye edildiği süreçlerdir. Hem aşırı hem alt örneklemenin kombinasyonlarını içeren yöntemler ise hibrit yöntem olarak bilinir. Bu yöntemin avantajı, aşırı örneklemeye kıyasla daha hızlı ve basit bir yaklaşım olmasıdır. Fakat çoğunluk sınıfından çıkarılan gözlemler potansiyel olarak bazı yararlı bilgilerin kaybolmasına neden olabilmektedir [8]. Hulse vd., (2007) yeniden örneklemeye yönelik faydanın, pozitif ve negatif gözlemlerin oranına, eldeki verinin diğer özelliklerine ve sınıflandırma yöntemlerine bağlı olmak üzere bir dizi faktöre göre değişiklik gösterdiğini öne sürmüştür [7, 29].

2.1.2. Rastgele Aşırı Örnekleme Yöntemi (Random Over-Sampling - ROS)

Aşırı Örnekleme, azınlık gözlem sayısının arttırılması için bu örneklerin rastgele olacak şekilde kopyalanması sürecidir [27]. Çoğunluk sınıfının gözlem sayısı aynı kalırken azınlık sınıfı gözlemleri çoğaltılır. Bununla birlikte, aşırı örnekleme sürecinde gözlem sayısının artmasıyla alt örneklemeye göre bu işlem daha uzun sürmektedir. Bu yöntemde herhangi bir veri kaybı olmaksızın azınlık sınıfı çoğaltılmış olur. Fakat bu durum, azınlık sınıfındaki mevcut gözlemleri tekrarladığı için aşırı uyuma neden olabilmektedir [8]. Daha fazla veriyle çalıştığı için eğitim süresi uzamaktadır.

2.1.3. Rastgele Alt Örnekleme Yöntemi (Random Under-Sampling - RUS)

Alt örnekleme yöntemi ise, azınlık sınıfı gözlem sayısı sabit tutulurken sınıf dengesinin sağlanabilmesi için çoğunluk sınıfına ait rastgele seçilen mevcut gözlemlerin kaldırılmasıdır. Kaldırılan gözlemlerden sonra çoğunluk sınıfının alt kümesiyle devam edilir. Alt örneklemenin dezavantajı, kaldırılan gözlemlerle beraber potansiyel olarak sınıflamaya fayda sağlayacak bilgiler de kaybolabilmektedir [7]. Daha az veriyle çalıştığı için eğitim süresi azalmaktadır.

2.1.4. Sentetik Azınlık Aşırı Örnekleme Yöntemi (Synthetic Minority Over-Sampling Technique - SMOTE)

SMOTE, bir aşırı örnekleme yöntemidir. Algoritma, Chawla vd. (2022) tarafından önerilmiş olup birçok dengesiz veri problemlerinde kullanılmıştır. Bu yöntem, azınlık sınıfı gözlemlerinin rastgele çoğaltılmasının aksine, var olan azınlık sınıfının veri alanı keşfedilerek buna yönelik sentetik gözlemler üretmeye dayanır. Yeni azınlık gözlemleri oluşturmak için, azınlık sınıfından rastgele seçilen gözlem ve yakınındaki gözlemler arasındaki fark alınır.

SMOTE yöntemiyle yeni azınlık sınıfları oluşturulurken, eğitim setindeki sınıflar arası dengesizlik yüzdesi belirlenir. Buna göre azınlık sınıfı için üretilecek gözlem sayısı hesaplanır. Rastgele seçilen her azınlık gözlemi dikkate alınarak, bu gözlemler için belirlenen k en yakın komşuları (varsayılan 5) Öklid uzaklığına göre bulunur.

$$sentetik = gozlem_j + (gozlem_j - komsu_i)c$$

Yukarıda görüldüğü gibi rastgele seçilen $gozlem_j$ ve her seçilen $komsu_i$ için fark hesaplanır. Daha sonra hesaplanan bu fark, 0 ve 1 aralığında bulunan rastgele

bir sayı (c) ile çarpılır ve ilk başta seçilen $gozlem_j$' e eklenir. Böylelikle, gözlem ile komşuyu birleştiren hat boyunca bir nokta seçilerek yeni sentetik gözlemlerin üretilmesi sağlanır [1].

2.1.5. Rastgele Aşırı Örnekleme Örnekleri Yöntemi (Random Over-Sampling Examples - ROSE)

Rastgele Aşırı Örnekleme yöntemi, Menardi vd. (2014) tarafından geliştirilmiştir. Sentetikler gözlemler elde edilirken veri kümesinin özellikleri değiştirilmeden dönüştürülmesi amaçlanır. Örneklerin olasılık yoğunluk fonksiyonları kullanılarak düzleştirilmiş (smoothed) bootstrap temelli bir yaklaşıma sahiptir [4].

$\mathbf{T}$, eğitim seti ve $\mathbf{T}_n$, n örnekli bir eğitim seti olmak üzere, $\mathbf{R}^d$ üzerinden tanımlanan X değişkenler olsun. Değişkenlerin olasılık yoğunluk fonksiyonu $P(\mathbf{x}) = f(\mathbf{x})$ ile ifade edilir. İki sınıflı veriye sahip olunduğu varsayılarak, Y_i' nin boyutu $n_j < n$ olmak üzere, Y_j (j=0,1) etiketler olarak gösterilsin. Buna göre ROSE yönteminin adımları aşağıdaki gibi tanımlanmıştır [4].

1. $\frac{1}{2}$ olasılık ile $y = Y_j \in Y$ seçilir.
2. $\frac{1}{n_j}$ olasılık ile $y_i = y$ olmak üzere $(\mathbf{x}_i, y_i) \in \mathbf{T}_n$ seçilir.
3. $\mathbf{x}_i$ merkezli bir olasılık dağılımı ile $\mathbf{H}_j$ matrisi baz alınarak $K_{\mathbf{H}_j}(., \mathbf{x}_i)$' den $\mathbf{x}$ gözlemi oluşturulur [4].

Olasılık dağılımları kullanılarak seçilen gözlemler $\mathbf{H}_j$ matrisi baz alınarak $\mathbf{x}$ gözlemini oluştururlar.

2.1.6. Maliyet duyarlı öğrenme

Maliyet duyarlı öğrenme, algoritma düzeyinde olan yöntemler kategorisine girmektedir. Mevcut öğrenici, çoğunluk sınıfına yönelik yanlılığı ortadan kaldıracak şekilde değiştirilir [27]. Doğruluk oranını yükseltmek yerine, yanlış sınıflandırılan gözlemlere odaklanarak maliyet hatasını düşürmek amaçlanmaktadır [6]. Bu yanlış sınıflandırılan gözlemlere yüksek ağırlıklar verilerek doğru şekilde sınıflandırılması amaçlanır. Doğru sınıflandırılan gözlemlere herhangi bir ceza verilmemektedir. Eğitim setinin toplam maliyeti küçültülmeye çalışılır. Bununla birlikte maliyet değerlerinin belirlenmesi her zaman kolay olmayabilir [27].

2.1.7. Hibrit Yöntem

Hibrit yöntemlerde, daha önce tanımlanan yöntemleri birleştirerek, uygun bir yanlılık seçmek hedeflenir. Topluluk öğrenmesi, veri ve algoritma seviyesindeki yöntemleri birleştirmektedir.

Topluluk sınıflandırıcıların en çok bilinenleri Bagging (Torbalama) ve Boosting (Arttırma) algoritmalarıdır [27, 30]. Bagging yönteminde, eğitim seti N alt kümeye bölünür. Bölünen bu alt kümelerin her biri sınıflandırma yapmak için kullanılır. Boosting yönteminde ise, birçok öğrenicinin (zayıf öğrenici) bir araya getirilmesiyle güçlü bir tahmin modeli elde edilmek amaçlanır. Zayıf öğreniciler tek bir tahmin modelinde birleştirilir [30].

2.2. Makine Öğrenmesi

Makine öğrenmesi, insan zekası baz alınarak bu yeteneği makinelere aktarma güdüsüyle ortaya çıkmıştır. Günümüz gerekliliklerini karşılamak, veriyi aktif kullanabilmek amacıyla geliştirilmiş ve zamanla kullanım alanı artarak etkin rol oynamıştır.

Veri geçmiş, gelecek ve şu an için önemli bir rol oynamaktadır. Makine öğrenmesi ise gerekli çıkarımların yapılabilmesi, verinin her yönüyle değerlendirilmesine imkan sağlamaktadır. Makinelerin, veriler ile öğrenerek karar verebilmesi veya belli sonuçlar çıkararak uygun alanlarda kullanılması amaçlanır. Bu amaçlara bağlı olarak makine öğrenmesi algoritmaları geliştirilmiş, veriler analiz edilerek yorumlanabilir, modeller oluşturularak anlamlı sonuçlar alınabilir hale gelmiştir.

Makine öğrenmesi her alanda uygulanabilir olmasıyla beraber teknoloji ve yaşamımızın bir parçası olmuş, bir gereklilik halini almıştır [31].

Makine öğrenmesi kabaca üçe ayrılmaktadır: Denetimli, Denetimsiz ve Pekiştirmeli Makine Öğrenmesi.

Denetimli makine öğrenmesi etiketli verilerden oluşmaktadır. Eldeki verilerden model oluşturulur ve modelin görmediği gözlemler üzerinde tahminler gerçekleştirilir. Girdi değerlerine bakılarak çıktı değerleri tahmin edilir. Örneğin, bir bireyin kanser olup olmadığı belli testler ve görüntülerden sonra belli olmaktadır. Buradaki değişkenler; kan değerleri, vücut sıvısı ya da ultrason ve X-ışınları olabilir. Bu değişkenlerden çıkan sonuçlara bağlı olarak tahminler yapılır [32].

Denetimsiz öğrenmede ise, denetimli öğrenmenin aksine çıktı değerleri bulun-

mamaktadır. Girdi değerlerine göre aralarındaki ilişki bulunarak, daha çok verinin genel yapısını anlamak, kümelemek hedeflenir.

Pekiştirmeli öğrenmede ise bir görevi yerine getirmek için doğrudan yazılımların çevreyle etkileşime girerek öğrenmesi söz konusudur. Karşılaşılan durumla alakalı etkileşimden aldığı geri bildirim ile öğrenir. Böylelikle ideal davranışı belirlemeye çalışır. Bu yöntem robotik, oyun programlama, hastalık teşhisi ve fabrika otomasyonu gibi alanlarda sıklıkla kullanılır [32].

Şekil 2.2. *Makine öğrenmesi algoritmaları*

Makine öğrenmesi algoritmaları Şekil 2.2' de olduğu gibi sınıflandırılmıştır. Buna bağlı olarak, eldeki veri dikkate alınarak algoritmalar uygulanmaktadır.

2.3. Sınıflandırma ve Regresyon

Sınıflandırma ve regresyon denetimli öğrenme kategorisine girmektedir ve Breiman vd. (1984) tarafından geliştirilmiştir [33]. Makine öğrenmesinde yaygın olarak kullanılan ve veri yapısına bağlı olarak tahmin şeklini değiştiren iki önemli yaklaşım, sınıflandırma ve regresyondur [34].

Bağımsız değişken vektörü x üzerinden bağımlı değişken veya yanıt değişkeni olan y tanımlanır. Burada tanım kümesi, $x \in \chi$ ve $y \in Y$ olduğu varsayılsın. Eğer yanıt değişkeni değerleri sürekli veya kesikli sayısal değerlerden oluşuyorsa regresyon, kategorik değerlerden oluşuyorsa sınıflandırma problemi olarak söylenir. Regresyon

problemlerinde konut fiyat tahmini, bir arabanın yaptığı kaza sayısı örnek olarak gösterilirken konut mevkisi veya arabanın markası sınıflandırma problemi örneğidir. Ayrıca sınıflandırmanın, hasta tanı tespiti, örüntü-resim tanıma, dolandırıcılık tespiti gibi sıklıkla kullanıldığı alanlar mevcuttur [34].

$D = \{(x_1, y_1), ..., (x_n, y_n)\}$, n gözlemli eğitim seti olmak üzere $f(x)$ her χ örneğini Y ile eşleyen fonksiyondur. Buna bağlı olarak regresyon için ortalama kare hatası Eşitlik 2.1' deki gibi hesaplanır [35].

$$E(f\,(x) - E\,(y|x))^2 \tag{2.1}$$

Sınıflandırma için Y, J' nin farklı sınıflarını içeriyorsa $A_j = \{x : f(x) = j\}$ olarak yazılır. Bu bağlamda sınıflandırma ağacı Eşitlik 2.2' de görüldüğü gibi her A_j' nin bir küme birleşimi olduğu, kümelerin tekrarlı olarak bölünmesiyle oluşan özel bir sınıflandırıcı şeklidir [35].

$$\chi = \bigcup_{j=1}^{J} A_j \tag{2.2}$$

Böylelikle, Eşitlik 2.2, sınıflandırıcının bir karar ağacı olarak gösterilmesine izin verir. Benzer şekilde regresyon modelleri için de karar ağaçları oluşturulmaktadır [35].

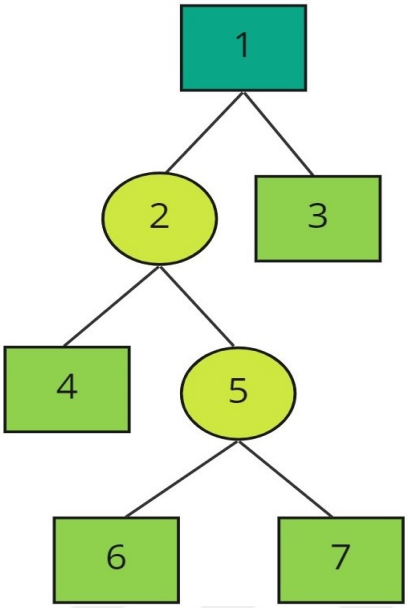
Literatürde sınıflandırma üzerine yapılmış birçok çalışma mevcuttur [36, 37].

2.4. Karar Ağaçları (Decision Trees)

Karar ağaçları, değişkenler arasındaki ilişkinin ortaya konmasını sağlayan yöntemlerden biridir [33]. Karar ağaçları hem regresyon hem de sınıflandırma problemlerinde kullanılabilir. Çıktı buna bağlı olarak sürekli veya kategorik değer olacaktır. Örnek karar ağacı yapısı Şekil 2.3' de verilmiştir.

Bir karar ağacı kök, düğüm ve dallardan oluşmaktadır. Şekil 2.3' de görüldüğü gibi kök düğüm "1" olarak gösterilmiştir. "2" ve "5" karar düğümleri ile dallanmalar gerçekleşmiş ve kalanlar ise yaprak olarak isimlendirilir, çıktı etiketlerini, temsil etmektedir [33].

Bağımsız değişkenlere sorulan sorular ve alınan cevaplarla karar ağacı oluşturulur. Karar ağacı yapısının oluşturulması aşamasında ağacın ne kadar genişleyeceği-



Şekil 2.3. *Karar ağacı yapısı*

nin kararı, bölünmelerin seçimi, düğümlerin devam edip/etmeyeceği veya bir sınıfa atanması gibi kararlar sınanır [38]. Düğüm noktaları karar noktalarını oluşturur. Her düğüm bir değişkenle etiketlendikten sonra dallar değişkenlere karşılık gelen değerlerle etiketlenmektedir. Oluşturulan ağacın bağımlı değişken ile ilişkisi ortaya çıkmaktadır [38]. Karar ağaçlarının oluşturulmasında hangi değişkenlerin seçileceği veya dallanma sırasında kullanılacak kriter ve bu sorulara ilişkin çeşitli algoritmalar geliştirilmiştir. Bunlar; ID3, C4.5, C5.0, CART ve CHAID algoritmalarıdır [38, 39].

Literatürde birden çok karar ağacı oluşturarak daha güçlü performans verebilen çeşitli algoritmalar mevcuttur. Bunlar; Random Forest (Rastgele Orman) ve Boosting (Arttırma) algoritmalarıdır [41, 42]. Karar ağaçlarıyla ile ilgili bir sınıflandırma çalışması da Asfha ve Kan Kılınç (2018) tarafından yapılmıştır [40].

2.5. Topluluk Öğrenmesi

Makine öğrenmesi yöntemleri yaygın bir şekilde kullanım alanı bulmuş ve buna bağlı olarak gelişime açık bir hale gelmiştir. Geleneksel makine öğrenmesi yöntemlerinde zaman zaman yüksek varyans ve yanlılık (bias) sorunları yaşanmaktadır. Bunun yanında eksik veya gürültülü veri gibi model performansını etkileyen sorunlar yaşanmaktadır. Topluluk öğrenmesi yöntemleri bu gibi sorunlarla baş edebilmek için geliştirilen yöntemlerden biridir. Yaygın olarak kullanılan topluluk öğrenmesi yön-

temleri bagging ve boosting' tir. Topluluk öğrenme yönteminin kullanılması daha düşük hata ve yüksek doğruluk performansıyla sonlanacaktır [41, 42].

Geleneksel makine öğrenmesi yöntemleri tek bir model oluşturarak tahmin yapmaktadır. Topluluk modelleri birçok kez veriyi eğiterek birçok öğrenici oluşturur. Topluluk öğrenmesinde örneklerin eğitildiği her bir öğrenene, zayıf öğrenici denir [43]. Zayıf öğreniciler; karar ağacı, destek vektör makineleri, yapay sinir ağları gibi algoritmalardır. Verinin bir kez eğitilerek oluşturulduğu modelin ve bundan elde edilen tahminler yerine, birçok kez eğitilen veriden elde edilen modelin daha güçlü olması beklenir. Örneğin, karar ağacı zayıf öğrenici olarak kullanıldığından, birden çok ağaç oluşturulur ve oluşturulan her öğrenicinin tahminleri birleştirilir [43, 44].

Makine öğrenmesi yöntemlerinde modelin performansını etkileyen durumlardan bazıları aşırı uyum (overfitting) ve yetersiz uyum (underfitting) problemleridir. Aşırı uyum, modelin eğitim setinde yüksek performans göstermesine rağmen test setinde tam tersi düşük performans ile sonuçlanması durumuna denir. Burada modelin eğitim setini ezberlemesi söz konusudur. Dolayısıyla daha önce kullanılmayan veriler üzerinden iyi performans gösteremeyecek ve varyans yükselecektir. Yetersiz uyum, modelin veriyi yeterince iyi öğrenemediği veya doğru olan model seçilmediğinde yanlılık artacak ve model performansını düşürecektir [41].

Topluluk öğrenmesi yöntemleri öğrenici sonuçlarının birleştirilmesiyle aşırı uyum ve yanlılığı azaltmayı amaçlamaktadır. Bu bağlamda, zayıf öğreniciler birleştirilerek oluşturulan nihai modelin doğruluğu arttırması, hatayı azaltması beklenir. Doğru sınıflandırıcı ve doğru birleştirme tekniği ile iyi sonuçlar alınabilir. Bir sınıflandırma problemi ele alındığında öğrenicileri birleştirmek için oy çokluğu (vote), ortalama veya ağırlıklı ortalama gibi yaklaşımlar kullanılır. Sınıf tahmini yapılacak gözlem için, tüm zayıf öğrenici tahmin oyları değerlendirilerek en sık oylanan sınıfa atanması ile gerçekleşir.

Topluluk öğrenmesi yönteminin başarılı olmasını sağlayan özelliklerden biri de, veri setinde yeterli gözlem sayısının bulunmasıdır. Veri setinin yeterli sayıda örneğe ve değişken sayısına sahip olması tahmin performansına olumlu katkı sağlayabilir. Bu başarıyı arttıran bir diğer özellik ise, farklı öğrenicilerin tahmin performansını yükseltmektedir. Aynı öğrenici hatalarını birleştirmek ilerleme yaratmazken topluluk öğrenmesi farklı hataların yapılmasına olanak sağlamaktadır. Öğrenici tahminlerinde

farklılıkların olabilmesi için eğitim verilerinin farklı alt kümeleri veya farklı değişkenleri kullanılabilir. Alt küme örnekleri, bootstrap örnekleme yöntemleri ile oluşturulabilir. Farklı alt kümelerin kullanılması ile öğreniciler farklı hatalar yapacaktır. Aynı şekilde farklı parametre ayarlarının yapılması da performansı etkileyecektir [41, 45].

2.5.1. Bootstrap (Yeniden örnekleme) ve Bagging (Torbalama)

Bootstrap, örneklemleri veri kümesinden bağımsız ve rastgele seçilerek oluşturulan yeniden örnekleme yöntemidir. Bagging, Bootstrap Aggregating teriminin kısaltılmış hali olup, mevcut olan veri kümesinden yeniden örneklemeye dayalı olarak birçok alt örneklem oluşturmakla ilgilidir. Bu alt örneklemlere bootstrap örneklemi denir [46].

Bootstrap, popülasyon dağılımının bilinmediği durumlarda popülasyona yönelik çıkarsamalar yapmak için kullanılır. Dağılımın bilinmediği durumlarda tahminler, bootstrap örneklemlerine bağlı olarak bulunur. İlgilenilen veri kümesi popülasyon olarak kabul edilir. Eldeki mevcut veri setinden iadeli olarak, popülasyonun mevcut gözlem sayısı korunacak şekilde rastgele olarak seçilen gözlemler, alt örneklemleri oluşturur, seçilen her gözlemin tekrar seçilme olasılığı vardır ya da hiç seçilmeme olasılığı da mümkündür. Her gözlemin seçilme olasılığı eşit olasılıktadır ve seçilen gözlemler eğitim setini oluşturur [46, 47]. Örneklem ortalaması alınarak oluşturulan modelde, parametrelerin örnekleme dağılımları oluşturulur. Örnekleme dağılımı, asimtotik olarak normal dağılıma sahiptir. Bu dağılım kullanılarak, popülasyon parametresi tahmin edilebilir. Ayrıca, güven aralıkları oluşturulup, hipotez testleri uygulanabilir. Tahmincinin standart sapması ve standart hatası hesaplanabilir [48, 49].

Varsayılsın ki, gözlem sayısı N olan bir veri kümesi, Bootstrap yeniden örnekleme ile alt örneklemler oluşturulmak istenildiğinde, her örneklemde yine N kadar gözlem olacak fakat süreç iadeli olduğundan, gözlemlerin her defasında seçilme olasılığı $1/N$ kadar olacaktır. Seçilmeme olasılığı ise Eşitlik (2.3)' de olduğu gibi hesaplanır [46].

$$(1 - \frac{1}{N})^N \simeq \exp(-1) \simeq 0.368 \tag{2.3}$$

Burada, verinin %36.8 test için kullanılacak örnekleri oluştururken, kalanı %63.2'

si eğitim seti için kullanılır.

Modelin tahmin hatasını hesaplamak için orijinal veri kümesi veya örneklemeye dahil olmayan gözlemler kullanılır, bu işlem bootstrap örneklem sayısı kadar M kez tekrarlanır ve ortalama tahmin hatası olarak hesaplanır.

Orijinal gözlemler kullanıldığında, eğitim seti ile test seti benzer olacağından aşırı uyum (overfitting) sorunu yaşanabilir. Bu etkiyi azaltmak, Efron ve Tibshirani (1998)' in "tahmin edicisi" kullanılır. Bu tahmin edici, Eşitlik (2.4)' de verilmiştir. Buna bootstrap hata tahmin edicisi denir [46].

$$hata = \frac{1}{M}\sum_{i=1}^{M}[(0.632 \times testh_i) + (0.368 \times toplamh_i)] \tag{2.4}$$

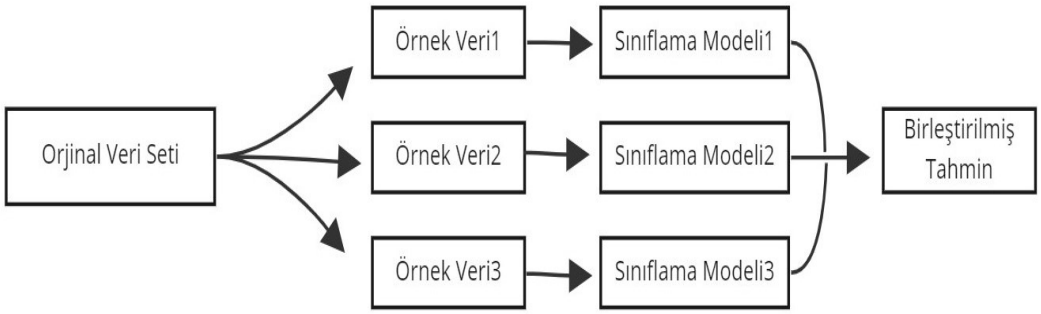
Eşitlik (2.4)' te görüldüğü gibi M, karar verilen bootstrap örneklem sayısı olmak üzere, $testh_i$ i. bootstrap örneklemi ile oluşturulan modelin test setine uygulandığında hesaplanan hata değeri ve $toplamh_i$ ise i. bootstrap örneklemi ile oluşturulan modelin orijinal veri setine uygulandığında hesaplanan hata değeridir. Bu durumda, veri kümesinin % 36.8' i test için kullanılırken % 63.2' si de eğitim için kullanılır [47].

İlgilenilen tahmin için her bootstrap örnekleminden, örnekleme dağılımı elde edilir. Bootstrap örneklem sayısı M olmak üzere, bootstrap gözlem sayısı en fazla teorik olarak n^n kadar olacaktır [48].

Bagging yöntemi, Breiman tarafından (1996) geliştirilmiştir [50]. Bagging, Bootstrap Aggregating teriminin kısaltılmış halidir. Torbalama olarakta bilinen bu yöntem, Bootstrap yeniden örnekleme yöntemini kullanarak alt örneklemler oluşturur. Bootstrap kısmında açıklandığı gibi veri setinden iadeli ve birbirinden bağımsız olarak bootstrap örneklemleri oluşturulur [50, 51]. Bu örneklemler birbirinden bağımsız olmakla beraber birbirine paralel olarak oluşturulur. Bagging yöntemi bootstrap yönteminden farklı olarak, N gözlemli veri setinden k tane gözlem seçer ve yine aynı şekilde iadeli ve rastgele olacak şekilde b tane bootstrap örneklemi oluşturur. Bu oluşturulan örneklemler, gözlem sayısı kadar devam eder. Böylelikle, b tane örneklemden b tane model kurularak tahminleme yapılır. Küçük eğitim verilerinde Bagging iyi bir yöntem olabilir [47]. Bagging yöntemi, hem sınıflandırma hem de regresyon problemlerinde kullanılabilir. Problem, sınıflandırma ise oy çokluğu alınarak, regresyon ise ortalamalar alınarak nihai model elde edilmiş olur. Bu yöntem,

daha çok karar ağaçları ile kullanılır.

Bagging yönteminde, tek bir model yerine birden fazla model kurulur. Bu modeller birleştirilerek nihai model oluşturulduğundan geleneksel yöntemlerden farklı olarak elde edilen örneklemlerin bir araya gelerek varyansı düşürüp, aşırı uyumu azaltması hedeflenir. Hata azaltılarak tahmin performansı bu sayede iyileştirilebilir [47, 51]. Şekil 2.4' de bagging yönteminin bootstrap yeniden örnekleme yöntemini kullanarak torbalardan elde edilen tahminleri birleştirme aşaması gösterilmiştir. Burada torba sayısı (n) kadar örnekleme yapılır ve bu durum n=3 için Şekil 2.4' te gösterilmiştir.



Şekil 2.4. *Bagging yöntemi*

Literatürde sıklıkla kullanılan OOB (Out Of Bag) ifadesi, torba dışı gözlemler olarak adlandırılır. OOB hatası; torba dışı hata, verilerin eğitilirken aynı zamanda test edilmesini sağlar. Orijinal veri kümesi, eğitim seti olarak düşünülsün. Bu durumda, orijinal veriden alt örneklemler oluşturulurken her örneklemde seçilmeyen gözlemler olacaktır. Bu bakımdan OOB, tahminleri test etmek açısından önem taşır [52].

OOB hatası, bir doğrulama setine gerek kalmadan bagging tahminlerini genelleştirmenin kolay bir yoludur. Model oluşturma aşamasında kullanılmayan gözlemlerin oluşturduğu torba dışı gözlemler verinin yaklaşık olarak % 37' lik kısmıdır. Bu torba dışı gözlemler test seti gibi kullanılır. Eğitim seti dışında kalan gözlemler ilgili modellerini test etmek için kullanılabilir [47]. Rastgele ormanlar bu şekilde oluşturulur. Bunlar bagging ile eğitilmiş bir karar ağacı topluluğudur. Orijinal verinin eğitim aşamasında birçok karar ağacı oluşturulur dolayısıyla bootstrap örneklemleri ve OOB seti oluşturulur. Böylelikle, orijinal verinin farklı alt örneklemlerinden oluş-

turulan her ağaç, o ağacın yapımında kullanılmayan gözlemler üzerinde test edilir. OOB, tahmin performansının iyileştirilmesini ve rastgele bir orman modelin doğruluğunun değerlendirilmesini sağlamaktadır [52].

2.5.2. Boosting (Arttırma)

Son yıllarda topluluk öğrenmesi yöntemlerinden olan boosting algoritmaları, popülerlik kazanmış birçok veri bilimi yarışmasında da ilgi odağı olmuştur [53]. Schapire (1990) ve Freund' in (1995) bireysel çalışmalarından sonra Freud ve Schapire' ın (1996) çalışmalarıyla devam etmiştir [30,43]. Daha sonra, Friedman vd. (2000) çalışmasında, kayıp fonksiyonun gradyan optimizasyonu ile ilişkisini göstermiştir [44].

Boosting algoritmasının amacı, zayıf (temel) öğreniciler üzerinden güçlü öğreniciler oluşturma prensibine dayanmaktadır [43]. Boosting, zayıf öğrenicilerden elde edilen tek bir tahminden daha iyi sonuçlar veren ve güçlü bir öğrenici oluşturmak için tekrarlı bir yaklaşım olarak tanımlanabilir. Zayıf öğreniciler daha sınırlı tahmin yeteneğine sahip olma eğilimindedir. Bu yüzden güçlü tahminler yapabilmek için birçok öğreniciyi, sırasıyla, işleyen ve nihayetinde onları birleştiren bir yaklaşıma sahiptir. Burada daha az eğitim hatası ile güçlü tahminler yapılmak istenir. Algoritma, mevcut öğreniciden bir önceki öğrenicinin örnekleri ne kadar doğru tahmin ettiğini dikkate alacaktır. Buna bağlı olarak, yanlış tahmin edilen gözlemlere daha fazla ağırlık vererek hatayı azaltmayı amaçlar [51].

Boosting algoritması tahmini zor olan gözlemlere odaklanarak, performansı yükselterek, varyansı ve yanlılığı azaltabilir. Bagging yöntemindeki gibi bootstrap örneklemlerini içermez. Bunun yerine, ağaçları oluştururken var olan veri setinin değiştirilmiş hali kullanılır.

Boosting algoritmasının farklılaşması, temel öğrenici seçimi ve ağırlık güncelleme aşamalarından kaynaklanmaktadır [47]. Genellikle, tahminleyici model olarak karar ağaçları ile kullanılmaktadır. Boosting, karar ağaçlarında sınıflandırmanın olduğu veri setlerinde sınıflandırma oranlarını iyileştirmektedir [51].

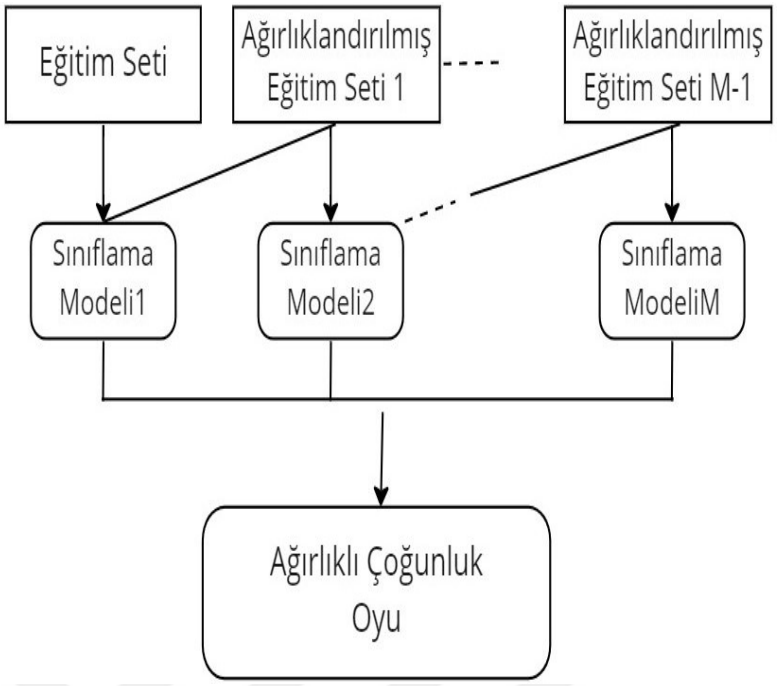
Bagging ve boosting yöntemleri karşılaştırıldığında, iki yöntemde de olduğu gibi tek bir öğreniciden ziyade birçok öğrenici oluşturulmak istenir. Bagging yöntemi bağımsız olarak oluşturulmuş modellerden oluşurken, boosting yöntemi önceki modellerin başarısız olduğu yerlerde başarılı olan yeni modeller oluşturmayı amaçlar [54].

Bagging yöntemi, her modele eşit ağırlık verirken boosting yöntemi, modelleri performanslarına göre ağırlıklandırmaya çalışır. Boosting yöntemi, yanlılığı azaltmaya çalışır. Öte yandan, bagging yöntemi aşırı öğrenme sorununu çözme eğilimindedir. Rastgele orman modeli bagging kullanırken, AdaBoost boosting tekniklerini kullanır.

Bu tez kapsamında literatürde bilinen boosting algoritmalarından, AdaBoost, GBM ve günümüzde yaygın kullanılan XGBoost algoritmaları incelenmiştir.

2.5.3. Adaptive Boosting (AdaBoost)

Boosting algoritmalarından biri olan AdaBoost 1996 yılında Freund ve Schapire tarafından tanıtılmıştır [43]. AdaBoost algoritması iki terminal yapraktan oluşan kütük (stump) adı verilen zayıf öğrenicilerden oluşur. Topluluk algoritmalarının altında boosting yönteminin kullanıldığı ilk pratik algoritmadır. Burada, zayıf öğreniciler bir araya getirilerek güçlü bir öğrenici elde edilmek istenir. Şekil 2.5' te AdaBoost algoritma süreci yer almaktadır. Şekilden de görüldüğü gibi ardışık modellerden oluşan AdaBoost algoritması için iterasyon sayısı belirlenir ve her adımda gözlemlerin ağırlıkları güncellenir. İlk olarak, eşit ağırlık verilen eğitim setindeki gözlemlerden yanlış sınıflandırılanların ağırlıkları arttırılarak bu gözlemlere daha fazla önem verililir. Doğru sınıflandırılan gözlemlerin ağırlıkları ise azaltılır [44]. Dolayısıyla, ilk öğrenici sonucu ağırlıkları artan eğitim verisindeki gözlemlerin seçilme olasılığı artar. Böylelikle, bir diğer eğitim sürecinde farklı veri kümesi eğitilecektir. Son olarak, sınıflandırma performansları dikkate alınarak bu öğreniciler oy çokluğuyla birleştirilir. Sınıflandırma performansı yüksek olan öğrenici, nihai sınıflandırma kararında daha etkili olacaktır [55].



Şekil 2.5. *AdaBoost algoritması*

AdaBoost algoritmasının uygulama kısmında süreci kontrol edebilmek için, kullanıcı tarafından belirlenen hiper parametreler mevcuttur. Bunlara ilişkin açıklamalar aşağıda verilmiştir [56].

- nu: Optimizasyon hızı veya öğrenme oranını kontrol eder.
- maxdepth: Ağacın maksimum derinliğini kontrol eder.
- iter: Oluşturulacak ağaç miktarını kontrol edilir (iterasyon sayısı).

Freund ve Schapire tarafından geliştirilen bu yöntemde etiketli eğitim gözlemleri $(x_1, y_1), .., (x_n, y_n)$ verilmiş olsun. Tanım kümesi, $x_i \in X$ olmak üzere, y etiketleri iki sınıflı $y_i \in \{-1, 1\}$ olarak ele alınsın. Algoritmada zayıf öğreniciler eğitilirken, her m iterasyonda örneklere bir ağırlık dağılımı D_m sağlar. Her bir öğrenme aşamasında, h_m zayıf öğreniciler eğitilir ve zayıf öğrenicinin görevi $h_m : X \to \{-1, +1\}$ sınıflandırmasını yapmaktır [30]. Algoritma adımları şu şekilde tanımlanmaktadır [30].

İterasyon sayısı $m = 1, 2, ..., M$ olmak üzere $i = 1, 2, ..., n$ gözlemlerine Eşitlik (2.5)' de olduğu gibi ilk atamalar (ağırlıklar) yapılır [30].

$$D_1(i) = 1/n \tag{2.5}$$

h_m zayıf sınıflandırıcısı, $h_m : X \to \{-1, +1\}$ yapılan atamalar sonucunda yanlış sınıflandırılan gözlemler için Eşitlik (2.6)' da hata hesaplanır. Bu hata zayıf öğrenicinin eğitildiği dağılıma göre hesaplanır.

$$\epsilon_m = Pr_{i\sim D_m} = [h_m(x_i) \neq y_i] \tag{2.6}$$

Eşitlik (2.7)' de bulunan α_m sınıflandırıcı performansını verir. ϵ_m ' nin bir fonksiyonu olarak seçilir ve bununla beraber bir sonraki eşitlikte ağırlık vektörü güncellenecektir.

$$\alpha_m = \frac{1}{2} ln \left(\frac{1 - \epsilon_m}{\epsilon_m} \right) \tag{2.7}$$

Eşitlik (2.8)' de gösterildiği gibi, yanlış sınıflanan gözlemler $(h_m(x_i) \neq y_i)$ için üstel fonksiyon $exp(\alpha_m)$ olurken, doğru sınıflanan gözlemler $(h_m(x_i) = y_i)$ için $exp(-\alpha_m)$ olarak güncellenir.

Eşitliğin 1' e eşit olması gerektiğinden, Eşitlik (2.8)' de $Z_{(m)}$ normalleştirme faktörü (D_{m+1} dağılım olacak şekilde seçilir) eklenerek gözlemlerin ağırlıkları güncellenir.

$$D_{m+1}(i) = \frac{D_m(i) \exp\left(-\alpha_m y_i h_m\left(x_i\right)\right)}{Z_m} \tag{2.8}$$

Her iterasyonda süreç tekrarlanarak öğreniciler oluşturulur.

Son olarak, zayıf öğrenicilerin çoğunluk oyu alınarak,

$$H\left(x\right) = sign(\sum_{m=1}^{M} \alpha_m h_m(x)) \tag{2.9}$$

Eşitlik (2.9)' da verilen $H(x)$ nihai atamaları yapılmış olur [30].

2.5.4. Gradient Boosting Machine (GBM)

Gradient boosting algoritması, Friedman (2001) tarafından üzerinde çalışılmış daha sonra bu çalışmalar genişletilmiştir [44, 57]. GBM, AdaBoost algoritmasında olduğu gibi zayıf öğrenicilerden güçlü öğrenici oluşturma sürecidir. Artıklar ya da hatalar, kayıp (loss) fonksiyonunun negatif kısmi türevleri üzerinden belirlenir. Algoritmanın arkasındaki temel fikir, yeni zayıf öğrenicileri tüm toplulukla ilişkili kayıp

fonksiyonun negatif gradyanı ile maksimum düzeyde ilişkilendirecek şekilde oluşturmaktır [53]. AdaBoost algoritmasından farklı olarak her iterasyonda gözlemleri ağırlandırmak yerine, türevlenebilir bir kayıp fonksiyonu minimize ederek negatif gradyan vektörüne uydurmak amaçlanır [47]. Kayıp fonksiyon verildiğinde, oluşturulan her ağaç bir önceki ağacın hatalarına (artık) odaklanarak tahminler yapılır ve kayıp fonksiyon minimize edilmeye çalışılır [53]. Ayrıca, kayıp fonksiyonun optimal minimum değerinin bulunmasında hiper parametrelerin değerleri önem taşımaktadır.

Genel olarak, karar ağaçları ile birlikte kullanıldığından dolayı Gradient Boosted Trees olarak da geçmektedir. Kayıp fonksiyon olarak, regresyon problemleri için, ortalama kare hatası veya sınıflandırma için log-olabilirlik kullanılabilir [57].

GBM, sırayla artıklara dayalı olarak yeni ağaçlar inşa etmektedir. Bir önceki ağacın artıkları göz önünde bulundurularak artığı yeniden oluşturmak için ağaçlar sırayla eklenerek, eğitilir. Performansta daha fazla gelişme olmadığı veya ağaç sayısı maksimum seviyeye ulaşana kadar yeni ağaçlar eklenir. Tahminler, artıklar toplamının 0'a yakın veya tahmin edilen değerler, gerçek değerlere yeterince yakın olacak şekilde güncellenir [57].

GBM aşamasında süreci kontrol edebilmek için, kullanıcı tarafından belirlenen parametreler mevcuttur. Aşağıda açıklamaları verilen parametre değerleri için uygun optimizasyonlar yapılabilir [20].

- shrinkage: Optimizasyon hızı veya öğrenme oranını kontrol eder.
- interaction.depth: Ağacın maksimum derinliğini kontrol eder.
- n.trees: Oluşturularacak ağaç miktarını kontrol eder (iterasyon sayısı).
- n.minobsinnode: Yapraklarda olması gereken minimum gözlem sayısını kontrol eder.

GBM algoritması şu şekilde tanımlanmaktadır: y bağımlı değişken ve $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ açıklayıcı değişkenler kümesi olmak üzere, eğitim örneği, $(y_i, \mathbf{x}_i)_i^N$ ve $x_i \in R^P$ ele alınsın, $y_i \in \{-1, 1\}$ sınıf etiketleri olsun. $L(y, F(x))$ türevlenebilir herhangi keyfi bir kayıp (loss) fonksiyon olarak tanımlanabilirken; $y \in R^1$ (regresyon için) olmak üzere, kareli hata $(y - F)^2$ ve mutlak hata $|y - F|$' yı içerir ve

$y \in \{-1, 1\}$ (sınıflandırma) için negatif binomiyal log-likelihood (negative binomial log-likelihood), $log(1+exp(-2yF(x)))$ değerini içerir. Burada sınıflama problemi ele alındığı için kayıp fonksiyon, Eşitlik (2.10)' daki gibi tanımlanır [57].

$$L(y, F(x)) = log(1 + exp(-2yF(x))), \qquad y \in \{-1, 1\} \tag{2.10}$$

Burada y gözlenen ve $F(x)$ tahmin edilen değerleri temsil etmektedir.

Buradan, Eşitlik (2.11)' de olduğu gibi $y = 1$ ve $y = -1$ eşit olasılıklara sahip sınıfları göstermektedir. $P(y = 1|x)$ ifadesi x verildiğinde y' nin 1 olma olasılığını, $P(y = -1|x)$ ifadesi x verildiğinde y' nin -1 olma olasılığını gösterir [58].

$$F(x) = \frac{1}{2} log \left[\frac{P(y = 1|x)}{P(y = -1|x)} \right] \tag{2.11}$$

Kayıp fonksiyon minimum olacak şekilde ilk tahminler yapılır, Eşitlik (2.12)' de görüldüğü gibi F_o değeri hesaplanır.

$$F_o\left(x\right) = \ argmin_\gamma \sum_{i=1}^{N} L\left(y_i, \gamma\right) \tag{2.12}$$

Eşitlik (2.13)' de $s = 1, \ldots, S$ iterasyon sayısı olmak üzere ve bir önceki tahmin $F_{s-1}(x)$, N boyutlu veri uzayında r_i^s=$\{-r_s(x_i)\}_i^N$ sözde artıklar (pseudo residuals), gerçek değer ile tahmin edilen değerin farkı olacak şekilde Eşitlik (2.13)' de olduğu gibi her iterasyonda güncellenir.

$$r_i^s = - \left[\frac{\partial L(y_i, F\left(x_i\right))}{\partial F\left(x_i\right)} \right]_{F(x)=F_{s-1}(x)}, \qquad \forall s = 1, \ldots, N \tag{2.13}$$

Bu aşamada, $h_s(x)$ yeni zayıf öğrenici ile karar ağacı, j yaprak sayısı $j = 1, ..., Js$ olmak üzere ve R_{js}' de terminal bölgeleri temsil etmek üzere ağacın bittiği düğümleri (node) ifade eder. Eşitlik (2.14)' de bulunan γ minimize edilerek, γ_{js} değerleri bulunur.

$$\gamma_{js} = argmin_\gamma \sum_{x_i \in R_{js}} L\left(y_i, \ \ F_{s-1}(x) + \gamma\right) \tag{2.14}$$

Yeni tahminler, Eşitlik (2.15)' de görüldüğü gibi güncellenir.

$$F_s(x) = F_{s-1}(x) + \nu \sum_{j=1}^{J} \gamma_{js} 1(x \in R_{js}) \tag{2.15}$$

Bu eşitlikte $1(x \in R_{\{js\}})$ fonksiyonu, x örneğinin s' inci iterasyonda R_{js} terminal bölgesine ait olup olmadığını gösteren bir indikatör fonksiyondur. ν parametresi ise aşırı uyumu önlemeyi sağlayan, modelin karmaşıklığını azaltan bir etkisi vardır. Bu nedenle kayıp fonksiyon cezalandırılır. Bu parametre, çapraz doğrulama yoluyla tahmin edilebilir [58].

2.5.5. Extreme gradient boosting (XGBoost)

XGBoost, gradient boosting algoritmasının verimli, hızlı bir farklı versiyonudur. Boosting temelli algoritmayı yayınlayan iki araştırmacı, Chen ve Guestrin (2016) olmuştur [59]. XGBoost karar ağaçları regresyon ve sınıflama problemlerinde kullanılan bir yöntemdir. Diğer algoritmalara göre 10 kat daha hızlı performans göstermiştir [59].

XGBoost hem regresyon hem de sınıflandırma problemlerinde kayda değer bir başarı elde etmiş olsa da, performansı genellikle dengesiz sınıflandırma durumu gündeme geldiğinde belirsiz hale gelebilir [24].

XGBoost algoritmasına çeşitli optimizasyonlar eklenmiştir [53]. Türevlenebilir keyfi bir kayıp fonksiyon ve aşırı uyum ile mücadele edebilmek için düzenlileştirme terimi içermektedir. Aşırı uyumu azaltmak için düzenlileştirme terimi eklenir, model cezalandırılarak daha az karmaşık olması istenir [60]. "Blocks for out-of-core computation" özelliği ile CPU üzerinde birden fazla çekirdek kullanılarak, veriyi alt kümelere böler (sub-sampling) ve paralel işlemlerin yapılarak sistem optimizasyonuna imkan sağlar [59]. Fakat paralelleştirme sınırlıdır, çekirdek iç ve dış döngüler arası geçiş yapar. Dış döngü karar ağacının yapraklarını oluştururken iç döngü değişkenlerinin hesaplanmasını sağlar. Dolayısıyla, iç döngü bitmeden dış döngüye geçilemez. Değişkenler hesaplandıktan sonra ağaç yaprakları oluşturulur [61].

Bir diğer özellik "sparsity awareness" özelliği sayesinde XGBoost kayıp gözlemler ile çalışabilir. Algoritmanın bu eksik gözlemlerden haberdar olması için eksik gözlemler ağaçlara eklenir. Bölünmede (split) bu eksik gözlemlere yön verilir [59,60].

Ağaca eklenen bu gözlemler düğüm noktasında sağ ve sol yapraklara eklenerek denenir ve her defasında ağacın kalitesi değerlendirilir.

XGBoost, karar ağaçlarını oluştururken ağaç yapısının üzerinden bir "gain" değeri hesaplar ve en yüksek gain değerini veren bölünme seçilir. Bu da amaç fonksiyonunun minimize edilmesinde etkili olacaktır. XGBoost, her bölünme aşamasında değişken alt kümesiyle çalışılmasına imkan sağlar. Seçilen değişken ile bölünmeler gerçekleşir [59]. Algoritma her düğümde en iyi bölünmeyi bulabilmek için olası tüm değişken değerlerini sıralamaktadır. Bu süreç özellikle büyük veri setlerinde oldukça zaman alabilir. Verileri her düğümde tekrar tekrar sıralamaktan kaçınmak için algoritma yalnızca bir kez değişkenleri sıralar. Sıralanan veri, "blok" adı verilen depolarda saklanmaktadır [20]. Daha sonra bu veri kümesinin alt kümesine karşılık gelen alt bloklar oluşturulur. Bu alt bloklar paralel olarak oluşturulur böylelikle hız kazandırılmış olur.

Veri kümeleri arasında en iyi bölünme noktalarının bulunması için "weighted quantile sketch" kullanılır [59]. Ağırlıklandırma yapılarak kararsız kalmış gözlemlerin daha doğru tahminlerde bulunması amaçlanır.

XGBoost yönteminde Gradient Boosting algoritmasına göre daha çok parametre bulunmaktadır. Bunların açıklamaları aşağıda verilmiş olup optimizasyon süreci, model performansının değerlendirilmesi açısından oldukça önemlidir [20,61].

- eta: Optimizasyon hızı veya öğrenme oranını kontrol eder.
- max_depth: Ağacın maksimum derinliğini kontrol eder.
- nrounds: Oluşturularacak ağaç miktarını kontrol eder (iterasyon sayısı).
- gamma: Aşırı öğrenmeye engel olur.
- colsample_bytree: Karar ağacının eğitilme aşamasında her ağaç için rastgele seçilecek olan değişken sayısını belirtir.
- subsample: Karar ağacının eğitilme aşamasında her ağaç için rastgele seçilecek olan gözlem sayısını belirtir.
- min_child_weight: Bölünmenin devam etmesi için gereken minimum toplam ağırlığı belirtir.

Burada, Chen ve Guestrin (2006) çalışmasında verilen XGBoost algoritması açıklanmaya çalışılmıştır. Buna göre n birim sayısı ve m değişken (feature) olmak üzere tanım kümesi $D = \{(\mathbf{x}_i, y_i)\}$, $(|D| = n, \mathbf{x}_i \in R^m$ ve $y_i \in R)$, $i = 1, 2, ..., n$ olarak eğitim veri setidir. Burada, kayıp karesel hata fonksiyonunu optimize etmek yerine iki parçalı bir amaç fonksiyonu tanımlanır. Öyle ki, burada eğitim seti üzerinden bir kayıp fonksiyon ve modelin karmaşıklığını düzenleyen bir ceza terimi bulunur. Bu durum Eşitlik (2.16)' da verilmiştir. $L(y_i, \widehat{y_i})$ türevlenebilir bir kayıp fonksiyondur. y_i gerçek değerler, $\widehat{y_i}$ tahmin değerlerini gösterirken fonksiyon, bu ikisi arasındaki farkı ölçer [60].

$$AmacFonksiyonu = \sum_{i=1}^{n} L(y_i, \widehat{y_i}) + \sum_{k=1}^{n} \Omega(f_k) \tag{2.16}$$

Eşitlik (2.16)' da $\Omega(f_k)$ terimi f_k ağacının karmaşıklığını ifade eder ve $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda w^2$ dir. Buradaki T, f_k' ıncı ağacın yapraklarının sayısıdır ve w, yaprak ağırlıklarıdır (yaprak düğümlerdeki tahmin değerleridir).

Eşitlik (2.17) ağaç topluluk modeli olarak adlandırılır, K tane toplamsal fonksiyondan oluşur ve bağımlı değişkeni tahmin eder. $F = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}(q : R^m \rightarrow T, w \in R^T)$ regresyon ağaçları uzayını tanımlar. q, ilgili yaprakları gözlemlere eşleyen her ağacın yapısını temsil eder [59].

$$\widehat{y_i} = \sum_{k=1}^{K} f_k(\mathbf{x}_i) \tag{2.17}$$

Her iterasyonda Eşitlik (2.18)' de, $Obj^{(t)}$ değeri azaltılmak istenir. Burada $\widehat{y_i}$, t' inci iterasyondaki i' inci tahmin değerini gösterir.Bu amaç fonksiyonunu minimize etmek için modele f_t terimi eklenir.

$$Obj^{(t)} = \sum_{i=1}^{n} L(y_i, \widehat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{2.18}$$

Eşitlik (2.19)' da eklenilen her ağaç yaprağı (T) için minimum kaybı ifade eden γ parametresi bulunur ve ağırlıkları (w) cezalandıran düzenlileştirme terimi olan λ parametresi bulunur [59].

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda w^2 \tag{2.19}$$

$Obj^{(t)}$ amaç fonksiyonunu azaltmak için yeni ağaçlar (f_t) eklenir. Fakat, Eşitlik (2.18) Öklid uzayında geleneksel yöntemler kullanılarak optimize edilemez. Bu nedenle, amaç fonksiyonunu azaltmak için ikinci dereceden Taylor açılımı Eşitlik (2.20)' de olduğu gibi yazılır.

$$Obj^{(t)} \simeq \sum_{i=1}^{n} [L(y_i,\, \widehat{y}_i^{(t-1)}) + g_i f_t\,(x_i) + \frac{1}{2} h_i\; f_t^2(x_i)] + \Omega(f_t) \tag{2.20}$$

Birinci ve ikinci dereceden kayıp fonksiyonun türevleri sırasıyla Eşitlik (2.21)' de olduğu gibi g_i ve h_i olarak gösterilmiştir.

$$gi = \frac{dL(\; y_i, \widehat{y}_i^{(t-1)})}{d\widehat{yi}^{(t-1)}}, hi = \frac{d^2L(\; y_i, \widehat{y}_i^{(t-1)})}{d^2\widehat{y}_i^{(t-1)}} \tag{2.21}$$

Sabitleri kaldırılan basitleştirilmiş fonksiyon Eşitlik (2.22)' deki gibi yazılır.

$$Obj^{(t)} = \sum_{i=1}^{n} [g_i f_t\,(x_i) + \frac{1}{2} h_i\; f_t^2(x_i)] + \Omega(f_t) \tag{2.22}$$

Eşitlik (2.23)' de $\Omega(f_t)$ eşitlikte yerine yazılır.

$$Obj^{(t)} = \sum_{i=1}^{n} [g_i f_t\,(x_i) + \frac{1}{2} h_i\; f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda w^2 \tag{2.23}$$

$I_j = \{i/q(x_i) = j\}$ eşitliği j yaprağındaki eğitim seti gözlemleri olarak tanımlanır. Tanımdan yola çıkarak Eşitlik (2.24)' de $f_t(x)$ fonksiyonu, w_j olarak yani, j' 'inci yaprağın ağırlığı olarak temsil edilir.

$$Obj^{(t)} = \sum_{j=1}^{T} [(\sum_{i\in\; I_j} g_i) w_j + \frac{1}{2}(\sum h_i + \;\lambda) w_j^2] + \gamma T \tag{2.24}$$

Her w_j değeri $Obj^{(t)}$ amaç fonksiyonunu azaltmaktadır. Buna göre en iyi w_j* değeri Eşitlik (2.25)' de olduğu gibi hesaplanır.

$$w_j* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} \tag{2.25}$$

w_j* değeri $Obj^{(t)}$ fonksiyonuna göre düzenlendiğinde en iyi ağaç yapısı (q) elde edilmek hedeflendiği için Eşitlik (2.26)' daki gibi yazılacaktır.

$$Obj^{(t)}(q) = \frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{2.26}$$

Ağaç kalitesini (q) ölçmek için bir puan fonksiyonu (gain) kullanılır. Ağaç kalitesi arttıkça amaç fonksiyonunda azalma olacaktır. Tek bir yapraktan başlayan ve yinelemeli olarak ağaca dallar ekleyen açgözlü bir algoritma ile ağaç genişletilir her bölünmede gain hesaplanır. Varsayılsın ki, $I = I_L \bigcup I_R$ burada tanımlanan I kök, I_L ve I_R sırasıyla bölünmeden sonraki sol ve sağ düğümlerinin gözlemlerini temsil etsin. Bu ağaç için Eşitlik (2.27)' de olduğu gibi "gain" değeri hesaplanır [59, 60].

$$Gain = \frac{1}{2}[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}] - \gamma \tag{2.27}$$

Böylelikle algoritma mevcut ağaç kalitesini değerlendirir ve bölünme eklenip eklenmeyeceğine karar verilir [59, 60].

2.6. Sınıflandırma Performans Ölçütleri

Eğitilen sınıflandırma problemlerinin değerlendirilmesi için çeşitli performans ölçütleri vardır.

Dengesiz bir veri ile çalışıldığında performans ölçütünün seçilmesi ve buna bağlı olarak değerlendirilmesi gerekir. Sınıflandırma problemlerinde doğruluk önemli bir performans ölçütü olmakla birlikte çoğunluk sınıfı (negatif sınıf) baz alınarak hesaplanır. Dolayısıyla azınlık sınıfı (pozitif sınıf) göz ardı edilmiş olur. Bundan dolayı F-ölçütü, G-ortalama, duyarlılık, seçicilik, kesinlik ve ROC eğrisi altında kalan alan (AUC) gibi performans ölçütleri de dikkate alınmalıdır [28, 62].

İki sınıflı problemlerde sınıflandırma performansının değerlendirilmesinde kullanılan karmaşıklık matrisi Tablo 2.1' de verilmiştir.

Tablo 2.1. *Karmaşıklık matrisi*

		Gerçek Durum	
		Pozitif Sınıf	Negatif Sınıf
Tahmin	Pozitif Sınıf	Doğru Pozitif (DP)	Yanlış Pozitif (YP)
	Negatif Sınıf	Yanlış Negatif (YN)	Doğru Negatif (DN)

Karmaşıklık matrisinde bulunan ifadeler,

- DP: model tarafından doğru şekilde tahmin edilen pozitif gözlemlerin sayısı,
- YP: model tarafından yanlış şekilde tahmin edilen negatif gözlemlerin sayısı,
- DN: model tarafından doğru şekilde tahmin edilen negatif gözlemlerin sayısı,
- YN: model tarafından yanlış şekilde tahmin edilen pozitif gözlemlerin sayısı,

olarak söylenebilir.

Bunun yanında, karmaşıklık matrisinden faydalanılarak elde edilen diğer performans ölçütleri aşağıda açıklanmıştır.

Doğruluk (Accuracy): Doğru olarak tahmin edilen gözlem sayısının tüm gözlem sayısına oranı olarak Eşitlik (2.28)' de görüldüğü gibi hesaplanır.

$$Doğruluk = \frac{DP + DN}{DP + YP + DN + YP} \tag{2.28}$$

Duyarlılık (Recall): Doğru tahmin edilen pozitif sınıf gözlemlerinin gerçekte tüm pozitif gözlemlere oranı olarak Eşitlik (2.29)' da olduğu gibi hesaplanır.

$$Duyarlılık = \frac{DP}{DP + YN} \tag{2.29}$$

Seçicilik (Specificity): Doğru tahmin edilen negatif sınıf gözlemlerinin gerçekte tüm negatif gözlemlere oranı olarak Eşitlik (2.30)' da görüldüğü gibi hesaplanır.

$$Seçicilik = \frac{DN}{DN + YP} \tag{2.30}$$

Kesinlik (Precision): Doğru tahmin edilen pozitif sınıf gözlemlerinin tahminde tüm pozitif gözlemlere oranı olarak Eşitlik (2.31)' deki gibi hesaplanır.

$$Kesinlik = \frac{DP}{DP + YP} \tag{2.31}$$

F-ölçütü: Ölçütlerin tek başına yeterli olmadığı durumlarda F-ölçütü duyarlılık ve kesinlik ölçütlerinin harmonik bir ortalaması olarak Eşitlik (2.32)' de görüldüğü gibi hesaplanır.

$$F = 2 * \frac{Kesinlik * Duyarlılık}{Kesinlik + Duyarlılık} \tag{2.32}$$

G-ortalama: Pozitif ve negatif sınıf performans değerlerinin her ikisini de dikkate alan geometrik ortalama Eşitlik (2.33)' de olduğu gibi hesaplanmaktadır.

$$G = \sqrt{Duyarlılık * Seçicilik} \tag{2.33}$$

Eğri Altında Kalan Alan (AUC): Alıcı çalışma karakteristiği (ROC) görsel olarak sınıflandırma performansını gösteren bir diğer performans ölçütüdür. İki boyutlu bu grafikte AUC, ROC eğrisi altında kalan alan Eşitlik (2.34)' de görüldüğü gibi hesaplanmaktadır.

$$\widehat{AUC} = \int_0^1 \widehat{ROC}\,(t)\,dt \tag{2.34}$$

Grafik üzerinde, x ve y ekseni sırasıyla gerçek pozitif oran ve gerçek negatif oran olarak yer alır. ROC eğrisi, duyarlılık ve seçiciliği dikkate alarak uygun kesim noktasını belirler. Bu değer 1 ise mükemmel sınıflandırıcıyı ifade eder [9].

3. UYGULAMA

Bu tez çalışması kapsamında, dengesiz veri oluşturma ve buna bağlı boosting yöntemlerinin performans analizlerine ilişkin senaryolar ele alınmıştır. Çeşitli dengesizlik oranları %10, %15, %20, %25 ve %30 olacak şekilde 250, 500 ve 1000 gözlemden oluşan veri setleri ile topluluk yöntemlerinden XGBoost, GBM ve AdaBoost algoritmaları kullanılarak, kategorik iki sınıflı bir değişkenin tahmini probleminde, kullanılan örnekleme yöntemlerinin performansları araştırılmıştır. Sınıflar arası dengesizlik sorununu çözmek için literatürde önerilen bu örnekleme yöntemlerinden, ROS, RUS, ROSE ve SMOTE ele alınarak, boosting algoritmalarının tahmin performansları ve süreleri araştırılmıştır.

Simülasyonda kullanılacak verinin üretilmesi aşamasında dünya çapında önemli üniversitelere yapılan başvurular ve bu başvurulara dair kabul/red oranları incelenmiştir (http-1). Bu istatistikler incelenerek oluşturulan veri, 4 bağımsız ve 1 kategorik bağımlı değişkenden oluşmaktadır. Üretilen bağımsız değişkenler analizlerde cinsiyet, yaş, puan ve mülakat puanı olarak yer alırken, kategorik değişken için "kabul" ve "red" miktarlarını ifade eden ve analizlerde y olarak adlandırılmış iki sınıflı bir değişken olarak tanımlanmıştır. Buna göre, adayın üniversiteye yaptığı başvuru "kabul" ise y' nin sınıf etiketi "1" değilse "0" olarak atanmıştır. Yaş değişkeni oransal ölçekli olduğu varsayılarak ortalaması 18 ve standart sapması 2 olan normal dağılımdan üretilmiştir. Üretilen bu yaş değerleri 17' den küçük değerler ise yeniden değer ataması yapılmıştır. Bilim sınavı değerlendirmesi olarak puan değişkeni ortalaması 4 ve standart sapması 0.5 olan normal dağılımdan üretilmiştir. Mülakat değişkeni 4 kategorisi olan nitel bir değişkendir. Kategoriler değerlendirildiğinde, 1=kötü, 2=orta, 3=iyi ve 4= çok iyi durumunu tanımlamaktadır. Bağımsız değişkenler oluşturulduktan sonra bağımlı değişken için sınıf etiketlerini belirlemede koşul tanımlanmıştır. Bu koşula göre, y' nin değerleri; puan değeri 4' e eşit veya büyük ya da mülakat değeri 4' e eşitse "kabul" değilse "red" olarak tanımlanmıştır. Bu koşula uyan tüm gözlemler seçildikten sonra sadece dengesizlik oranı kadar gözlem rassal olacak şekilde seçilmiş ve bu süreç 1000 defa tekrar etmiştir. Daha sonra, veriler eğitim ve test verisi olarak ikiye bölünmüştür. Verinin %70' i eğitim için, %30' u ise yöntemleri test etmek için kullanılmıştır. Örnekleme sadece eğitim verisi üzerinden yapılmıştır.

Tablo 3.1. *Hiper parametre değerleri*

Algoritma	Hiper Parametre Aralık Değerleri	Hiper Parametre Optimizasyonu
XGBoost	eta=[0-1] nrounds=[1-Inf) max_depth=[1-Inf) min_child_weight=[0-1] colsample_bytree=[0-1] gamma=[0-Inf) subsample=[0-1]	eta= 0.1,0.01,0.001 nrounds= 50,150,300 max_depth=1,3,5,7 min_child_weight=0.5,1 colsample_bytree=1 gamma=0 subsample=1
GBM	n.trees=[1-Inf) interaction.depth=[1-Inf) shrinkage=[0-1] n.minobsinnode=[1-Inf)	n.trees=50,150,300 interaction.depth=1,7,2 shrinkage=0.1,0.01,0.001 n.minobsinnode=({4,10}, {1,5})
AdaBoost	max_depth=[1-Inf) nu=[0-1] iter=[1-Inf)	maxdepth= 1,7,2 nu=0.1,0.01,0.001 iter=50,150,300

Ayrıca, 5 katlı çapraz doğrulama yapılarak boosting algoritmalarının hiper parametre optimizasyonu gerçekleştirilmiştir. Hiper parametre optimizasyonu için "Grid Search" yapılarak geniş bir hiper parametre uzayı yaratılmıştır. Buna ilişkin Tablo 3.1' de belirlenen hiper parametre değerleri incelenebilir.

Bu değerler için kesin bir kanı bulunmamakla beraber bazı çalışmalar mevcuttur [19,20]. Belirlenen hiper parametre değerleri kullanılarak model oluşturulmuş ve elde edilen perfomans ölçütleri 1000 sonucun ortalaması olarak tablolara kaydedilmiştir.

Modeller eğitilirken, 5 katlı çapraz doğrulama ile yapılan optimizasyon işleminde en iyi hiper parametreler AUC değeri yüksek olana göre seçilmektedir. Buna göre tahminler yapılarak "1" ve "0" için olasılıklar elde edilmiştir. Eşik değeri 0.5 olarak ayarlandıktan sonra sınıf tahminleri yapılmış ve performans ölçütleri hesaplanmıştır. Daha sonra, XGBoost, GBM ve AdaBoost yöntemleri ile iterasyon sayısı kadar model oluşturulmuş ve tahminler yapılmıştır. Elde edilen performans ölçütlerinin (metriklerin) ortalaması 1000 iterasyon sonucunun ortalaması olarak Tablo 3.2- 3.4 arasında sunulmuştur.

Ayrıca simülasyon verinin oluşturulmasından performans ölçütlerinin hesaplanmasına kadar geçen süre kaydedilmiş ve tüm sonuçlar hem algoritmalar hem örnekleme yöntemleri bazında karşılaştırılmış ve Tablo 3.8-3.10 arasında verilmiştir. Çapraz doğrulama sürecini hızlandırmak için paralelleştirme yapılarak, CPU çekirdek kullanımı optimizasyonu yapılmıştır.

Tablo 3.2' deki sonuçlar gözlem sayısı 250 ve 5 farklı dengesizlik oranı kullanı-

larak elde edilmiştir. Buna göre, XGBoost, GBM ve AdaBoost yöntemlerinin (0.10, 0.15, 0.20, 0.25, 0.30) dengesizlik altındaki performans karşılaştırmaları verilmiştir. Burada, GBM algoritmasında bulunan "n.minobsinnode" parametresi (4,10) olarak belirlendikten sonra sınıflandırma yapılmıştır. Bu parametre yapraklarda kalacak olan gözlem sayılarını belirtmektedir. Bu gözlem sayısına göre ağaç bölünmeye devam edecektir. Gözlem sayısı 250 ve dengesizlik oranı %10 için bu parametre (4,10) olduğunda örneklemin küçük olmasında dolayı yapraklarda yeterli sayıda gözlem oluşturamadığı için hesaplama yapılamamıştır.

Tablo 3.2' de elde edilen sonuçlar incelendiğinde, gözlem sayısı 250 ve %10 dengesizlik oranıyla XGBoost algoritması için en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS yönteminde ve AUC değeri en yüksek SMOTE yönteminde gözlemlenmiştir. GBM algoritması için, en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü, G-ortalama ve duyarlılık ROSE yönteminde, AUC ise SMOTE yönteminde gözlemlenmiştir. GBM için, RUS sonuçları alınamamıştır. AdaBoost algoritmasında ise, en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenmiştir. En yüksek F, G-ortalama, AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında en iyi doğruluk, seçicilik ve kesinlik değerleri ROS yöntemiyle kurulan GBM algoritmasına aittir. En iyi F-ölçütü, G-ortalama, AUC ve duyarlılık değerleri ise RUS yöntemiyle kurulan AdaBoost' a aittir.

Benzer şekilde Tablo 3.2' de %15 ve %20 dengesizlik oranları için elde edilen sonuçlar incelendiğinde, XGBoost ve AdaBoost algoritmaları için en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken en yüksek F-ölçütü, G-ortalama, AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. GBM algoritması için ise %15 dengesizlik oranında , en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, %20 dengesizlik oranında farklı olarak en yüksek kesinlik değeri RUS yönteminde gözlemlenmiştir. Hem %15 hem %20 dengesizlik oranında en yüksek F-ölçütü, G-ortalama, AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında %15 ve %20 dengesizlikte, en iyi doğruluk, seçicilik değeri ROS yöntemiyle kurulan XGBoost' a aittir. Öte yandan %15 ve %20 dengesizlikte, en iyi F-ölçütü, G-ortalama ve

Tablo 3.2. *Dengesizlik oranları ve N=250 için performans karşılaştırmaları-(4,10)*

Dengesizlik: %10		Metrikler, N=250,(4,10)						
Sınıflayıcı	Yöntem	Doğruluk	F	G-Ort	AUC	Duyarlılık	Seçicilik	Kesinlik
XGBoost	ROS	**0.8294**	0.2137	0.3371	0.6886	0.1741	**0.9073**	**0.1838**
	RUS	0.6157	**0.2713**	**0.6338**	0.6944	**0.6953**	0.6066	0.1724
	ROSE	0.6429	0.2503	0.5912	0.6774	0.5695	0.6518	0.1632
	SMOTE	0.7286	0.2538	0.5549	**0.6958**	0.4375	0.7628	0.1791
GBM	ROS	**0.8311**	0.218	0.3299	0.6659	0.1711	**0.9094**	**0.186**
	RUS	-	-	-	-	-	-	-
	ROSE	0.6448	**0.2607**	**0.6114**	0.687	**0.6084**	0.6497	0.1691
	SMOTE	0.7359	0.2484	0.534	**0.6938**	0.4107	0.7745	0.1754
AdaBoost	ROS	**0.8252**	0.2166	0.3459	0.697	0.1848	**0.9011**	**0.182**
	RUS	0.6169	**0.2734**	**0.637**	**0.6991**	**0.6982**	0.6077	0.1729
	ROSE	0.6426	0.2583	0.6075	0.6872	0.6019	0.6478	0.168
	SMOTE	0.7283	0.255	0.5495	0.6978	0.4357	0.7627	0.1775
Dengesizlik: %15		N=250, (4,10)						
XGBoost	ROS	**0.7745**	0.2652	0.4482	0.7182	0.2566	**0.8724**	**0.2737**
	RUS	0.6406	**0.3877**	**0.6675**	**0.7264**	**0.7348**	0.6229	0.2682
	ROSE	0.6372	0.3603	0.6365	0.7102	0.6625	0.633	0.254
	SMOTE	0.7141	0.3357	0.5873	0.7228	0.4711	0.7595	0.2691
GBM	ROS	**0.7734**	0.2639	0.4557	0.7035	0.2621	**0.8702**	**0.2765**
	RUS	0.6457	**0.3889**	**0.6703**	**0.7309**	**0.7289**	0.63	0.2699
	ROSE	0.6387	0.372	0.6502	0.7194	0.6951	0.629	0.2603
	SMOTE	0.7161	0.3393	0.588	0.7218	0.4731	0.7617	0.2716
AdaBoost	ROS	**0.7735**	0.2654	0.4516	0.7269	0.2604	**0.8706**	**0.2751**
	RUS	0.6428	**0.392**	**0.6727**	**0.7342**	**0.7433**	0.6242	0.271
	ROSE	0.6396	0.3671	0.6445	0.7178	0.6791	0.6331	0.2583
	SMOTE	0.7135	0.3421	0.596	0.7272	0.4869	0.7559	0.2723
Dengesizlik: %20		N=250,(4,10)						
XGBoost	ROS	**0.7344**	0.3508	0.5409	0.7472	0.3683	**0.8286**	**0.3533**
	RUS	0.6597	**0.4724**	**0.6907**	**0.7537**	**0.7675**	0.6327	0.347
	ROSE	0.6443	0.4361	0.6529	0.7268	0.6954	0.6313	0.3252
	SMOTE	0.7037	0.4167	0.6254	0.7481	0.5377	0.7461	0.3505
GBM	ROS	**0.7323**	0.3511	0.5421	0.7397	0.3715	**0.8255**	0.3509
	RUS	0.6671	**0.4746**	**0.6925**	**0.7567**	**0.757**	0.644	**0.3511**
	ROSE	0.648	0.4514	0.6686	0.7349	0.7321	0.6268	0.3332
	SMOTE	0.7045	0.4145	0.6219	0.7471	0.531	0.7492	0.3494
AdaBoost	ROS	**0.7318**	0.3533	0.5456	0.7526	0.3784	**0.8232**	**0.3511**
	RUS	0.6607	**0.4774**	**0.6951**	**0.7561**	**0.7783**	0.6308	0.3497
	ROSE	0.6444	0.4452	0.6616	0.7335	0.7223	0.6251	0.3297
	SMOTE	0.7022	0.4212	0.6313	0.7501	0.5514	0.7411	0.3509
Dengesizlik: %25		N=250, (4,10)						
XGBoost	ROS	**0.7172**	0.4415	0.6052	0.7719	0.4703	**0.7986**	**0.432**
	RUS	0.6868	**0.5487**	**0.7139**	**0.7746**	**0.7913**	0.6531	0.4266
	ROSE	0.663	0.5141	0.6801	0.7507	0.7418	0.6372	0.4012
	SMOTE	0.7009	0.4982	0.6664	0.7703	0.619	0.7283	0.4258
GBM	ROS	**0.7174**	0.4441	0.6076	0.7679	0.475	**0.7971**	**0.4319**
	RUS	0.691	**0.5494**	**0.7147**	**0.775**	**0.7828**	0.6614	0.4297
	ROSE	0.6687	0.5274	0.6923	0.7578	0.7707	0.6354	0.4084
	SMOTE	0.7029	0.4987	0.6665	0.7706	0.6164	0.7319	0.428
AdaBoost	ROS	**0.7167**	0.4506	0.6145	**0.777**	0.4905	**0.7914**	**0.4333**
	RUS	0.6878	**0.5521**	**0.7169**	0.7768	**0.7984**	0.6519	0.4282
	ROSE	0.6643	0.5232	0.6875	0.7574	0.7665	0.6308	0.405
	SMOTE	0.7011	0.505	0.6729	0.7734	0.635	0.7236	0.4281
Dengesizlik: %30		N=250,(4,10)						
XGBoost	ROS	0.7165	0.542	0.6651	**0.7986**	0.58	**0.7754**	**0.5221**
	RUS	**0.7175**	**0.6274**	**0.7388**	0.798	**0.8164**	0.6757	0.5156
	ROSE	0.6896	0.596	0.7083	0.7746	0.7843	0.65	0.4886
	SMOTE	0.715	0.5823	0.7016	0.7957	0.6833	0.7286	0.5156
GBM	ROS	0.7159	0.5426	0.6657	0.7973	0.5826	**0.7732**	**0.521**
	RUS	**0.72**	**0.6279**	**0.7399**	**0.7995**	**0.8102**	0.6821	0.5188
	ROSE	0.6958	0.6071	0.7178	0.7801	0.8059	0.6491	0.4942
	SMOTE	0.7178	0.5855	0.7041	0.7953	0.686	0.7311	0.5187
AdaBoost	ROS	0.7166	0.5557	0.6777	**0.8011**	0.6132	**0.7616**	**0.521**
	RUS	**0.7197**	**0.6333**	**0.744**	0.7998	**0.8289**	0.6739	0.5183
	ROSE	0.6935	0.6039	0.7148	0.7787	0.7991	0.6493	0.4933
	SMOTE	0.7174	0.5909	0.709	0.7974	0.7015	0.7241	0.5179

duyarlılık değerleri RUS yöntemiyle kurulan AdaBoost' a aittir. En iyi AUC değeri ise %15 dengesizlikte RUS yöntemiyle AdaBoost iken %20 dengesizlikte ise GBM algoritmasına aittir. %15 dengesizlikte en iyi kesinlik değeri ROS yöntemiyle kurulan GBM algoritmasına ait olurken %20 için XGBoost' a aittir.

Benzer şekilde Tablo 3.2' de %25 dengesizlik oranı için elde edilen sonuçlar incelendiğinde, XGBoost ve GBM algoritmaları birlikte değerlendirildiğinde en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü, G-ortalama AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. AdaBoost kullanıldığında, en yüksek doğruluk, AUC, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken en yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında en iyi doğruluk değeri ROS yöntemiyle GBM algoritmasına aittir. En iyi F-ölçütü, G-ortalama ve duyarlılık RUS yönteminde ve en iyi AUC ve kesinlik değerleri ise ROS yöntemiyle kurulan AdaBoost' a aittir. En iyi seçicilik değeri ROS yöntemiyle kurulan XGBoost' a aittir.

Benzer şekilde Tablo 3.2' de %30 dengesizlik oranı için elde edilen sonuçlar incelendiğinde, XGBoost ve GBM algoritmaları birlikte değerlendirildiğinde en yüksek seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, en yüksek doğruluk, F-ölçütü, G-ortalama, AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. AdaBoost algoritması ile en yüksek AUC, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, en yüksek doğruluk, F, G-ortalama ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında, en iyi doğruluk değeri RUS yöntemiyle kurulan GBM algoritmasına aittir. En iyi F-ölçütü, G-ortalama ve duyarlılık değerleri RUS yöntemiyle kurulan AdaBoost' a aittir. En iyi AUC değeri ise ROS ile kurulan AdaBoost' a olurken, en iyi seçicilik ve kesinlik değerleri ROS ile kurulan XGBoost' a aittir.

Öte yandan, Tablo 3.3' de gözlem sayısı 500 ve tüm dengesizlik oranlarında algoritmaların performans karşılaştırmalarından elde edilen performans ölçütleri verilmiştir. Tablo 3.3' de elde edilen sonuçlar incelendiğinde, gözlem sayısı 500 ve %10 dengesizlik oranında, XGBoost algoritması için en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS yönteminde ve AUC değeri ise en yüksek SMOTE

yönteminde gözlemlenmiştir. GBM algoritması için, en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS ile gözlemlenirken, en yüksek F-ölçütü ve G-ortalama değerleri RUS yönteminde gözlemlenmiştir. En yüksek AUC değeri hem RUS hem SMOTE yönteminde, en yüksek duyarlılık ise ROSE yönteminde gözlemlenmiştir. AdaBoost' da ise, en yüksek doğruluk, seçicilik değerleri ROS yönteminde gözlemlenmiştir. En yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS ile gözlemlenirken en yüksek AUC ve kesinlik değerleri SMOTE ile gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında en iyi doğruluk ve seçicilik değerleri ROS yöntemiyle kurulan AdaBoost'a ait olurken en iyi F-ölçütü, G-ortalama ve duyarlılık RUS yöntemiyle ve en iyi AUC değeri SMOTE yöntemiyle kurulan AdaBoost' a aittir. En iyi kesinlik değeri ise ROS yöntemiyle kurulan XGBoost algoritmasına aittir.

Benzer şekilde, Tablo 3.3' de %15 ve %20 dengesizlik oranları için elde edilen sonuçlar incelendiğinde, XGBoost algoritması için en yüksek doğruluk ve seçicilik ROS yönteminde gözlemlenirken en yüksek F-ölçütü, G-ortalama AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. GBM algoritması için ise, en yüksek doğruluk ve seçicilik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü, G-ortalama ve AUC değerleri ise RUS yönteminde gözlemlenmiştir. En yüksek duyarlılık ve kesinlik değerleri sırasıyla ROSE ve SMOTE yöntemlerinde gözlemlenmiştir. AdaBoost algoritmasında ise, en yüksek doğruluk ve seçicilik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. En yüksek AUC değeri %15 dengesizlikte RUS, %20 ' de ise SMOTE yöntemindedir. En yüksek kesinlik değeri SMOTE yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında %15 dengesizlikte en iyi doğruluk, seçicilik değeri ROS yöntemiyle kurulan AdaBoost' a aittir. En iyi F-ölçütü, G-ortalama ve AUC değerleri RUS yöntemiyle kurulan AdaBoost' a aittir. En iyi duyarlılık ve kesinlik değerleri sırasıyla ROSE ve SMOTE yöntemiyle kurulan GBM algoritmasına aittir. Dengesizlik oranı %20 için farklı olarak en iyi AUC ve kesinlik değerleri SMOTE yöntemiyle AdaBoost' a aittir.

Benzer şekilde Tablo 3.3' de %25 dengesizlik oranı için elde edilen sonuçlar incelendiğinde, XGBoost, GBM ve AdaBoost algoritmaları birlikte değerlendirildiğinde en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü ve G-ortalama değerleri ise RUS yönteminde gözlemlenirken,

Tablo 3.3. *Dengesizlik oranları ve N=500 için performans karşılaştırmaları-(4,10)*

Dengesizlik: %10		Metrikler N=500,(4,10)						
Sınıflayıcı	Yöntem	Doğruluk	F	G-Ort	AUC	Duyarlılık	Seçicilik	Kesinlik
XGBoost	ROS	**0.8346**	0.1863	0.3852	0.7089	0.187	**0.905**	**0.176**
	RUS	0.6164	**0.2685**	**0.6562**	0.7207	**0.7296**	0.6042	0.1665
	ROSE	0.6163	0.2512	0.6309	0.6979	0.6691	0.6114	0.157
	SMOTE	0.7308	0.2444	0.5794	**0.7216**	0.4548	0.7609	0.1705
GBM	ROS	**0.8315**	0.1892	0.3929	0.6986	0.1949	**0.9008**	**0.1752**
	RUS	0.6243	**0.2714**	**0.6604**	**0.7211**	0.7247	0.6133	0.1688
	ROSE	0.5852	0.255	0.6306	0.7069	**0.7334**	0.5696	0.1571
	SMOTE	0.7349	0.2441	0.5751	**0.7211**	0.4469	0.7662	0.1708
AdaBoost	ROS	**0.8415**	0.1771	0.3531	0.7154	0.1624	**0.9152**	0.1717
	RUS	0.6158	**0.2729**	**0.6633**	0.7242	**0.7456**	0.6017	0.1689
	ROSE	0.6157	0.2549	0.6368	0.7027	0.683	0.6091	0.159
	SMOTE	0.7286	0.2495	0.5879	**0.7264**	0.4715	0.7567	**0.1727**
Dengesizlik: %15		N=500, (4,10)						
XGBoost	ROS	**0.7751**	0.2631	0.4793	0.7359	0.2784	**0.8622**	0.2595
	RUS	0.6368	**0.3818**	**0.681**	**0.7435**	**0.7647**	0.6148	0.2571
	ROSE	0.6165	0.3622	0.6585	0.7212	0.7437	0.5947	0.2424
	SMOTE	0.7081	0.3427	0.6164	0.7412	0.5209	0.7411	**0.2597**
GBM	ROS	**0.7699**	0.2721	0.4952	0.7333	0.2988	**0.8527**	0.2605
	RUS	0.6406	**0.3823**	**0.6818**	**0.7438**	0.7589	0.6202	0.2582
	ROSE	0.5931	0.3675	0.6569	0.732	**0.8036**	0.5568	0.2419
	SMOTE	0.7096	0.3443	0.6174	0.7418	0.5222	0.7426	**0.2612**
AdaBoost	ROS	**0.7795**	0.2549	0.4655	0.7416	0.2615	**0.8703**	0.259
	RUS	0.6323	**0.385**	**0.6854**	**0.7451**	**0.7847**	0.6061	0.2577
	ROSE	0.6173	0.3655	0.6625	0.7273	0.7519	0.5942	0.2443
	SMOTE	0.7042	0.3466	0.6228	0.7447	0.5371	0.7336	**0.26**
Dengesizlik: %20		N=500,(4,10)						
XGBoost	ROS	**0.735**	0.3541	0.5523	0.7572	0.3784	**0.8226**	0.342
	RUS	0.6584	**0.4756**	**0.7027**	**0.7615**	**0.7994**	0.6241	0.3415
	ROSE	0.6423	0.4547	0.6811	0.7426	0.7709	0.6111	0.326
	SMOTE	0.7015	0.4274	0.646	0.7609	0.5754	0.7324	**0.3442**
GBM	ROS	**0.7301**	0.366	0.5679	0.7572	0.4051	**0.81**	0.3418
	RUS	0.6587	**0.4767**	**0.7039**	**0.7618**	0.802	0.6241	0.3423
	ROSE	0.6169	0.4618	0.6793	0.7518	**0.8439**	0.5617	0.3223
	SMOTE	0.7022	0.4273	0.6456	0.7616	0.5742	0.7335	**0.3445**
AdaBoost	ROS	**0.7361**	0.3508	0.5482	0.7616	0.3716	**0.8257**	0.3419
	RUS	0.6558	**0.4807**	**0.7079**	0.7614	**0.8206**	0.6156	0.3426
	ROSE	0.6407	0.461	0.6875	0.7468	0.792	0.6041	0.3284
	SMOTE	0.6984	0.4334	0.6535	**0.7628**	0.5959	0.7237	**0.3446**
Dengesizlik: %25		N=500, (4,10)						
XGBoost	ROS	**0.7178**	0.4563	0.6123	0.7835	0.4738	**0.7999**	**0.4448**
	RUS	0.6917	**0.5719**	**0.7264**	0.7845	**0.8201**	0.6485	0.4414
	ROSE	0.6721	0.5535	0.7079	0.766	0.8089	0.626	0.4234
	SMOTE	0.7069	0.5211	0.6792	**0.7852**	0.6356	0.7309	0.4443
GBM	ROS	**0.7166**	0.4696	0.6261	0.7839	0.5018	**0.7888**	**0.4457**
	RUS	0.693	**0.5749**	**0.729**	0.7854	0.8268	0.648	0.4431
	ROSE	0.6568	0.5608	0.7051	0.7751	**0.8671**	0.586	0.4185
	SMOTE	0.7079	0.5197	0.6777	**0.7858**	0.6305	0.7339	0.445
AdaBoost	ROS	**0.7187**	0.4613	0.617	0.7863	0.4824	**0.7982**	**0.447**
	RUS	0.692	**0.5785**	**0.7321**	0.7857	**0.8399**	0.6422	0.4432
	ROSE	0.6734	0.5614	0.7148	0.7715	0.8307	0.6205	0.4262
	SMOTE	0.7062	0.5288	0.6868	**0.7864**	0.6568	0.7228	0.445
Dengesizlik: %30		N=500,(4,10)						
XGBoost	ROS	0.7175	0.5481	0.6689	0.8042	0.5808	**0.7767**	**0.5259**
	RUS	**0.7239**	**0.646**	**0.753**	**0.8053**	**0.8504**	0.6703	0.5241
	ROSE	0.7027	0.623	0.7304	0.7881	0.8307	0.648	0.5026
	SMOTE	0.7198	0.5989	0.7146	0.8045	0.7063	0.7264	0.524
GBM	ROS	0.7173	0.5585	0.6785	0.8043	0.6057	**0.7658**	0.5245
	RUS	**0.7254**	**0.6512**	**0.7571**	**0.8066**	0.8661	0.6661	**0.5256**
	ROSE	0.6928	0.6332	0.7303	0.7975	**0.8923**	0.6077	0.4963
	SMOTE	0.7205	0.5998	0.7153	0.8049	0.7077	0.7267	0.5248
AdaBoost	ROS	0.717	0.5559	0.6762	0.8055	0.6011	**0.7676**	0.5246
	RUS	**0.725**	**0.6522**	**0.7579**	**0.8059**	**0.8697**	0.6636	**0.5247**
	ROSE	0.7054	0.6308	0.7369	0.7937	0.8508	0.6434	0.5049
	SMOTE	0.7205	0.6069	0.7215	0.8056	0.7285	0.7179	0.5241

duyarlılık değeri XGBoost ve AdaBoost için RUS, GBM için ROSE yönteminde gözlemlenmiştir. En yüksek AUC değeri ise SMOTE yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında en iyi doğruluk ve kesinlik değeri ROS yöntemiyle AdaBoost algoritmasına aittir. En iyi F-ölçütü, G-ortalama değerleri RUS, kesinlik değeri ROS ve en iyi AUC değeri SMOTE yöntemiyle kurulan AdaBoost algoritmasına aittir. En iyi duyarlılık değeri ROSE yöntemiyle kurulan GBM algoritmasına ve seçicilik değeri ROS yöntemiyle kurulan XGBoost algoritmasına aittir.

Benzer şekilde Tablo 3.3' de %30 dengesizlik oranı için elde edilen sonuçlar incelendiğinde, XGBoost, GBM ve AdaBoost algoritmaları birlikte değerlendirildiğinde en yüksek seçicilik değeri ROS yönteminde gözlemlenirken, kesinlik değeri XGBoost algoritmasında ROS, diğer algoritmalarda RUS yönteminde gözlemlenmiştir. En yüksek doğruluk, F-ölçütü, G-ortalama ve AUC değerleri ise RUS yönteminde gözlemlenmiştir. En yüksek duyarlılık değeri diğer dengesizlik oranlarında da olduğu gibi GBM algoritması için ROSE yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında, en iyi doğruluk ve AUC değerleri RUS yöntemiyle kurulan GBM' e aittir. En iyi F-ölçütü, G-ortalama ise RUS yöntemiyle kurulan AdaBoost' a ait olurken, en iyi duyarlılık değeri ROSE yöntemiyle kurulan GBM'e, en iyi seçicilik ve kesinlik değerleri ise ROS yöntemiyle kurulan XGBoost algoritmasına aittir.

Tablo 3.4' de gözlem sayısı 1000 ve tüm dengesizlik oranlarında performans karşılaştırmalarından elde edilen sonuçlar sunulmuştur. Tablo 3.4' de elde edilen sonuçlar incelendiğinde, gözlem sayısı 1000 ve %10 dengesizlik oranıyla XGBoost, GBM ve AdaBoost algoritmaları birlikte değerlendirildiğinde en yüksek doğruluk ve seçicilik değerleri ROS, kesinlik değeri GBM algoritması için RUS yönteminde iken XGBoost için ROS ve AdaBoost için her iki yöntemde de gözlemlenmiştir. En yüksek F-ölçütü, G-ortalama, AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında en iyi doğruluk ve seçicilik değerleri ROS yöntemiyle kurulan AdaBoost algoritmasına aittir. En iyi F-ölçütü, G-ortalama, AUC ve duyarlılık RUS yöntemiyle kurulan AdaBoost' a ve en iyi kesinlik değeri ROS yöntemiyle kurulan XGBoost algoritmasına aittir.

Benzer şekilde Tablo 3.4' de %15 dengesizlik oranı için elde edilen sonuçlar in-

Tablo 3.4. *Dengesizlik oranları ve N=1000 için performans karşılaştırmaları-(4,10)*

Dengesizlik: %10		N=1000,(4,10)						
Sınıflayıcı	Yöntem	Doğruluk	F	G-Ort	AUC	Duyarlılık	Seçicilik	Kesinlik
XGBoost	ROS	**0.8323**	0.1773	0.3946	0.7278	0.1805	**0.9061**	**0.1785**
	RUS	0.6148	**0.2857**	**0.6719**	**0.7345**	**0.7639**	0.598	0.1769
	ROSE	0.6194	0.2721	0.6509	0.7173	0.7071	0.6095	0.1697
	SMOTE	0.7231	0.2548	0.591	0.7317	0.4716	0.7516	0.1764
GBM	ROS	**0.8181**	0.1923	0.4306	0.7248	0.217	**0.8863**	0.1771
	RUS	0.6143	**0.2875**	**0.6748**	**0.7345**	**0.7717**	0.5967	**0.1778**
	ROSE	0.6432	0.2701	0.6343	0.7266	0.6678	0.6402	0.1734
	SMOTE	0.7237	0.2558	0.5922	0.732	0.4729	0.7522	0.1771
AdaBoost	ROS	**0.8328**	0.1742	0.3914	0.7347	0.1772	**0.9071**	**0.1773**
	RUS	0.6061	**0.2883**	**0.6768**	**0.7352**	**0.7906**	0.5853	**0.1773**
	ROSE	0.6176	0.2742	0.6548	0.7214	0.7184	0.6062	0.1707
	SMOTE	0.7182	0.2576	0.5978	0.7337	0.4863	0.7445	0.1769
Dengesizlik: %15		N=1000, (4,10)						
XGBoost	ROS	**0.7738**	0.2703	0.4921	0.7485	0.2872	**0.8586**	**0.2603**
	RUS	0.6299	**0.3892**	**0.691**	0.7501	**0.803**	0.6	0.2583
	ROSE	0.6289	0.3728	0.671	0.7351	0.7542	0.6072	0.2498
	SMOTE	0.7063	0.3478	0.6245	**0.7507**	0.5338	0.7364	0.26
GBM	ROS	**0.7576**	0.2959	0.5338	0.7484	0.3486	**0.8289**	**0.2611**
	RUS	0.6267	**0.3919**	**0.6939**	0.7504	**0.8192**	0.5935	0.2591
	ROSE	0.65	0.3669	0.6567	0.7446	0.7123	0.6391	0.2558
	SMOTE	0.7064	0.3491	0.6259	**0.7507**	0.5367	0.736	0.2608
AdaBoost	ROS	**0.7698**	0.2793	0.5057	**0.7523**	0.3054	**0.8507**	**0.2623**
	RUS	0.6234	**0.393**	**0.6955**	0.7509	**0.8291**	0.5877	0.2588
	ROSE	0.6272	0.3763	0.6752	0.7391	0.7669	0.603	0.2513
	SMOTE	0.7025	0.3523	0.6315	0.752	0.5513	0.7289	0.2608
Dengesizlik: %20		N=1000,(4,10)						
XGBoost	ROS	**0.7345**	0.3675	0.5621	0.7674	0.3881	**0.8217**	0.353
	RUS	0.658	**0.4927**	**0.7129**	0.7668	**0.8328**	0.6142	0.3517
	ROSE	0.6489	0.4757	0.6956	0.7554	0.8002	0.611	0.3407
	SMOTE	0.7002	0.4452	0.6595	**0.7694**	0.6028	0.725	**0.3551**
GBM	ROS	**0.7239**	0.3957	0.5976	0.7678	0.4541	**0.7921**	0.3539
	RUS	0.6552	**0.4961**	**0.7154**	0.7685	**0.8521**	0.6061	0.3521
	ROSE	0.665	0.4671	0.6829	0.7627	0.7601	0.6409	0.3495
	SMOTE	0.6992	0.4447	0.6594	**0.7694**	0.6038	0.7234	**0.3542**
AdaBoost	ROS	**0.7296**	0.3807	0.5788	0.7697	0.4179	**0.8083**	0.3536
	RUS	0.6538	**0.4974**	**0.717**	0.7682	**0.858**	0.6027	0.3518
	ROSE	0.6486	0.4785	0.6983	0.7576	0.8087	0.6084	0.3418
	SMOTE	0.6971	0.4485	0.664	**0.77**	0.6175	0.7173	**0.3543**
Dengesizlik: %25		N=1000, (4,10)						
XGBoost	ROS	**0.7166**	0.4644	0.6229	0.7866	0.4929	**0.7912**	**0.4412**
	RUS	0.6898	**0.58**	**0.7354**	0.7864	**0.8581**	0.6337	0.4394
	ROSE	0.6774	0.5681	0.7237	0.7758	0.85	0.6199	0.4281
	SMOTE	0.705	0.5277	0.6883	**0.7876**	0.6601	0.7199	0.4407
GBM	ROS	**0.7131**	0.4917	0.6511	0.7871	0.5563	**0.7653**	**0.4422**
	RUS	0.6889	**0.5846**	**0.7388**	0.7871	**0.8779**	0.626	0.4399
	ROSE	0.6879	0.5581	0.7137	0.782	0.8129	0.6463	0.4345
	SMOTE	0.7047	0.5308	0.6913	**0.788**	0.6691	0.7166	0.4411
AdaBoost	ROS	**0.7148**	0.4832	0.6422	**0.7881**	0.5346	**0.7749**	**0.4427**
	RUS	0.6882	**0.5842**	**0.7388**	0.7867	**0.8776**	0.625	0.439
	ROSE	0.6787	0.5725	0.7276	0.7788	0.8612	0.6178	0.43
	SMOTE	0.7037	0.5336	0.6941	**0.7881**	0.6789	0.7119	0.4407
Dengesizlik: %30		N=1000,(4,10)						
XGBoost	ROS	0.718	0.5563	0.6742	0.8081	0.5893	**0.7742**	0.5302
	RUS	**0.7274**	**0.6579**	**0.7602**	0.8078	0.8739	0.6642	0.5298
	ROSE	0.711	0.6455	0.7466	0.797	**0.8763**	0.6393	0.5131
	SMOTE	0.7237	0.6153	0.7263	**0.8086**	0.7352	0.719	**0.5309**
GBM	ROS	0.7207	0.5819	0.6971	0.8084	0.6474	**0.753**	**0.5312**
	RUS	**0.7291**	**0.6643**	**0.765**	**0.8087**	**0.8945**	0.6576	0.5308
	ROSE	0.7192	0.6438	0.7462	0.8032	0.8589	0.6587	0.5216
	SMOTE	0.7237	0.6168	0.7276	0.8085	0.7401	0.717	0.5308
AdaBoost	ROS	0.7198	0.576	0.6919	0.8087	0.6341	**0.7574**	0.5306
	RUS	**0.7288**	**0.6644**	**0.7653**	0.8088	**0.8948**	0.6571	0.5303
	ROSE	0.7131	0.6511	0.7511	0.7999	0.8912	0.6358	0.5148
	SMOTE	0.7246	0.6222	0.7322	**0.809**	0.7546	0.712	**0.5312**

celendiğinde, XGBoost ve GBM algoritmaları için en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenirken en yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. En yüksek AUC değeri SMOTE yönteminde gözlemlenmiştir. AdaBoost algoritmasında ise, en yüksek doğruluk, AUC, kesinlik ve seçicilik değerleri ROS yönteminde gözlemlenirken, en yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında en iyi doğruluk, seçicilik değerleri ROS yöntemiyle kurulan XGBoost' a aittir. En iyi F, G-ortalama ve duyarlılık değerleri RUS yöntemiyle kurulan AdaBoost' a aittir. En iyi AUC ve kesinlik değerleri ROS yöntemiyle kurulan AdaBoost' a aittir.

Benzer şekilde Tablo 3.4' de %20 ve %25 dengesizlik oranı için elde edilen sonuçlar incelendiğinde, XGBoost, GBM ve AdaBoost algoritması birlikte değerlendirildiğinde en yüksek doğruluk ve seçicilik değerleri ROS yönteminde gözlemlenirken en yüksek F-ölçütü, G-ortalama ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. En yüksek AUC değeri SMOTE yönteminde gözlemlenirken %25' te AdaBoost için hem SMOTE hem de ROS yöntemindedir. En yüksek kesinlik değeri %20 dengesizlikte SMOTE, %25' te ROS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında en iyi doğruluk, seçicilik değeri ROS yöntemiyle kurulan XGBoost' a aittir. En iyi F-ölçütü %20 dengesizlikte RUS yöntemiyle AdaBoost' a ve %25' te RUS ile GBM' e aittir. En yüksek G-ortalama RUS yöntemiyle kurulan AdaBoost' a aittir. Dengesizlik oranı %20' de duyarlılık değeri RUS yöntemiyle kurulan AdaBoost algoritmasına ait iken %25' te ise GBM' e aittir. En iyi AUC değeri SMOTE yöntemiyle kurulan AdaBoost algoritmasında ve kesinlik değeri ise SMOTE yöntemiyle XGBoost algoritmasına ve ROS yöntemiyle kurulan AdaBoost algoritmasına aittir.

Benzer şekilde Tablo 3.4' de %30 dengesizlik oranı için elde edilen sonuçlar incelendiğinde, XGBoost algoritması değerlendirildiğinde en yüksek doğruluk, F-ölçütü, G-ortalama değerleri RUS yönteminde gözlemlenirken, en yüksek AUC ve kesinlik değerleri ise SMOTE yönteminde gözlemlenmiştir. En yüksek duyarlılık ve seçicilik değerleri sırasıyla ROSE ve ROS yöntemlerinde gözlemlenmiştir. GBM algoritması için, en yüksek doğruluk, F, G-ortalama, AUC, duyarlılık değerleri RUS yönteminde gözlemlenirken en yüksek seçicilik ve kesinlik değerleri ROS yönteminde gözlem-

lenmiştir. AdaBoost algoritmasında ise, en yüksek doğruluk, F-ölçütü, G-ortalama ve duyarlılık değerleri RUS yönteminde gözlemlenirken, en yüksek AUC ve kesinlik değeri SMOTE ve en iyi seçicilik değeri ROS yönteminde gözlemlenmiştir. Tüm algoritmalar birlikte karşılaştırıldığında, en iyi doğruluk değeri RUS yöntemiyle kurulan GBM' e aittir. En iyi F-ölçütü, G-ortalama ve duyarlılık ise RUS yöntemiyle kurulan AdaBoost' ait olurken, en iyi seçicilik değeri ROS yöntemiyle kurulan XGBoost' a aittir. En iyi AUC değeri SMOTE yöntemiyle kurulan AdaBoost' a ve en iyi kesinlik değeri ROS yöntemiyle GBM ve SMOTE yöntemiyle AdaBoost'a aittir.

Öte yandan, GBM algoritmasının, "n.minobsinnode" parametresi sonraki aşamada (1,5) olarak belirlenmiş ve tekrar analizler yapılmıştır. Bu sonuçlara ilişkin elde edilen performans değerleri Tablo 3.5- 3.7 arasında verilmiştir.

Hiper parametre doğrulaması veriye bağlı olarak farklılık gösterebilir. Bu çalışmada "n.minobsinnode" hiper parametre değeri ilk olarak (4,10) ve daha sonra (1,5) olarak ele alındığında kaydedilen değerler birbirine oldukça yakın olarak bulunmuştur.

Tablo 3.5' de gözlem sayısı 250 ve farklı dengesizlik oranları altındaki performans karşılaştırmaları verilmiştir. Tablo 3.5' de elde edilen sonuçlar incelendiğinde, gözlem sayısı 250 ve %10, %15, %20, %25, %30 dengesizlik oranlarıyla en yüksek değerler Tablo 3.2 ' de olduğu gibi gösterilmiştir. GBM algoritmasında bulunan "n.minobsinnode" parametresi (1,5) olarak belirlendikten sonra sınıflandırma yapılmıştır.

Tablo 3.5' de elde edilen sonuçlar incelendiğinde gözlem sayısı 250 ve %10 dengesizlik oranıyla GBM algoritması değerlendirildiğinde en yüksek doğruluk, seçicilik ve kesinlik değerleri ROS yönteminde gözlemlenmiştir. En yüksek F-ölçütü, G-ortalama, AUC ve duyarlılık değerleri ise RUS yönteminde gözlemlenmiştir. Dengesizlik oranı %15, %20, %25, %30 bulunan performans ölçütleri incelendiğinde yöntemler bazında en yüksek değerler Tablo 3.2' de olduğu gibi aynıdır. Farklı olarak, kesinlik değeri %20 dengesizlikte RUS yönteminde değil SMOTE yönteminde daha yüksektir.

Tablo 3.5 ve 3.2 için elde edilen GBM algoritmasının değerleri iki tabloda da birbirine oldukça yakındır. İki tablo için performans ölçütleri karşılaştırıldığında, Tablo 3.5' de %15 dengesizlikte F-ölçütü, G-ortalama ve duyarlılık değerleri hariç,

Tablo 3.5. *GBM dengesizlik oranları ve N=250 için performans karşılaştırmaları-(1,5)*

Dengesizlik	Yöntem	Doğruluk	F	G-Ort	AUC	Duyarlılık	Seçicilik	Kesinlik
%10	ROS	**0.8317**	0.2153	0.3292	0.6675	0.1690	**0.9103**	**0.1837**
	RUS	0.6195	**0.2712**	**0.6355**	**0.698**	**0.6855**	0.6117	0.1725
	ROSE	0.6419	0.2608	0.6127	0.6893	0.6136	0.6461	0.1691
	SMOTE	0.7393	0.2491	0.5252	0.6938	0.4028	0.7788	0.1751
%15	ROS	**0.775**	0.2642	0.45	0.6998	0.2566	**0.873**	**0.2776**
	RUS	0.6446	**0.3878**	**0.6689**	**0.7299**	**0.7283**	0.6291	0.2692
	ROSE	0.6375	0.3744	0.6535	0.7214	0.7044	0.6258	0.2612
	SMOTE	0.7177	0.3401	0.5827	0.7214	0.4705	0.7643	0.2726
%20	ROS	**0.7333**	0.3479	0.5382	0.7327	0.3644	**0.8284**	0.3509
	RUS	0.6654	**0.4758**	**0.6939**	**0.7557**	**0.7641**	0.6404	0.3511
	ROSE	0.6464	0.454	0.6707	0.7361	0.7423	0.6224	0.334
	SMOTE	0.7047	0.4172	0.6262	0.7443	0.5384	0.7476	**0.3514**
%25	ROS	**0.7173**	0.4419	0.6056	0.7637	0.4717	**0.7985**	**0.4319**
	RUS	0.6891	**0.553**	**0.7177**	**0.776**	**0.7989**	0.6535	0.4292
	ROSE	0.6665	0.5326	0.6964	0.7617	0.7901	0.6268	0.4093
	SMOTE	0.7027	0.5015	0.6693	0.769	0.6244	0.729	0.4281
% 30	ROS	0.716	0.5423	0.6657	0.7934	0.5831	**0.7735**	**0.5207**
	RUS	**0.7192**	**0.63**	**0.7413**	**0.8003**	**0.8203**	0.6768	0.5177
	ROSE	0.6973	0.6102	0.7206	0.7822	0.8125	0.6488	0. 4959
	SMOTE	0.7187	0.5892	0.7074	0.7957	0.6943	0.729	0.5198

AUC değeri %15 ve %20 dengesizlikler hariç ve tüm dengesizlik oranları için seçicilik değerinin Tablo 3.5' de biraz daha yüksek olduğu görülmüştür. Doğruluk değeri ise %10, %15 ve %20 dengesizliklerde, kesinlik değeri %15 ve %20 dengesizlik oranlarında Tablo 3.5' de biraz daha yüksek olduğu görülmüştür. Ayrıca, kesinlik değeri %25 dengesizlikte iki tabloda da aynı değeri vermiştir.

Tablo 3.2' de bulunan XGBoost ve AdaBoost değerleri ile karşılaştırılan Tablo 3.5' e göre GBM değerleri; %15 dengesizlikte doğruluk ve seçicilik değerleri, %25 dengesizlikte F-ölçütü, G-ortalama ve duyarlılık değerleri diğer iki algoritmadan daha yüksek olduğu görülmüştür. Tersi olarak, %10 dengesizlikte kesinlik değeri, %20 dengesizlikte AUC değeri ve %30 dengesizlikte ise doğruluk değeri Tablo 3.5' de diğer iki algoritmadan daha düşük olduğu görülmüştür.

Tablo 3.6' daki sonuçlar gözlem sayısı 500 ve 5 farklı dengesizlik oranları altındaki performans karşılaştırmaları verilmiştir. Tablo 3.6' da elde edilen sonuçlar incelendiğinde yöntemler bazında en yüksek değerler Tablo 3.3' da olduğu gibi aynıdır.

Tablo 3.6- 3.3 için elde edilen GBM algoritmasının değerleri iki tabloda da birbirine oldukça yakındır. İki tablo karşılaştırıldığında Tablo 3.6' da tüm dengesizlik oranları için doğruluk, F-ölçütü, G-ortalama, duyarlılık ve seçicilik değerlerinin Tablo 3.6' da biraz daha yüksektir. AUC değeri ise %10 ve %15 dengesizliklerde, kesinlik değeri %20 dengesizlik oranında Tablo 3.6' da biraz daha yüksek olduğu

Tablo 3.6. *GBM dengesizlik oranları ve N=500 için performans karşılaştırmaları-(1,5)*

Dengesizlik	Yöntem	Doğruluk	F	G-Ort	AUC	Duyarlılık	Seçicilik	Kesinlik
%10	ROS	**0.832**	0.1858	0.3912	0.7024	0.1929	**0.9015**	**0.1735**
	RUS	0.6209	**0.2726**	**0.6627**	**0.722**	0.7354	0.6085	0.1691
	ROSE	0.5802	0.2567	0.6315	0.7085	**0.7473**	0.5626	0.1577
	SMOTE	0.7337	0.2455	0.5778	0.721	0.4525	0.7643	0.1716
%15	ROS	**0.7703**	0.2677	0.4895	0.734	0.2915	**0.8543**	0.2582
	RUS	0.6378	**0.3847**	**0.6845**	**0.7445**	0.7725	0.6146	0.2587
	ROSE	0.5862	0.3676	0.6547	0.733	**0.8157**	0.5465	0.241
	SMOTE	0.7074	0.3452	0.6186	0.7413	0.5275	0.7391	**0.2604**
%20	ROS	**0.7316**	0.3618	0.5624	0.7563	0.3956	**0.8141**	0.3417
	RUS	0.6576	**0.48**	**0.707**	**0.7615**	0.8153	0.6194	0.3432
	ROSE	0.6147	0.4619	0.6783	0.7521	**0.8486**	0.5579	0.3218
	SMOTE	0.7013	0.43	0.6489	0.7604	0.5826	0.7304	**0.3451**
%25	ROS	**0.7168**	0.4657	0.622	0.7828	0.4937	**0.7918**	**0.4454**
	RUS	0.6926	**0.5775**	**0.7312**	0.7852	0.8366	0.6442	0.4433
	ROSE	0.6539	0.5605	0.7031	0.7755	**0.8716**	0.5807	0.4175
	SMOTE	0.7066	0.5228	0.6808	**0.7853**	0.6409	0.7287	0.4443
%30	ROS	0.7165	0.5543	0.6749	0.8032	0.5967	**0.7686**	0.5236
	RUS	**0.7257**	**0.6532**	**0.7583**	**0.806**	0.873	0.663	**0.5254**
	ROSE	0.6928	0.6332	0.7303	0.7975	**0.8923**	0.6077	0.4963
	SMOTE	0.7205	0.6027	0.7178	0.8046	0.7164	0.723	0.5244

görülmüştür. Ayrıca, duyarlılık değeri %30 dengesizlikte iki tabloda da aynı değeri vermiştir.

Tablo 3.3' de bulunan XGBoost ve AdaBoost değerleri ile karşılaştırılan Tablo 3.6' a göre GBM değerleri; %10 dengesizlikte duyarlılık değeri, %20 dengesizlikte kesinlik değeri, %30 dengesizlik oranında ise F-ölçütü değeri ve G-ortalama değeri AdaBoost algoritmasından daha yüksek olduğu görülmüştür.

Tablo 3.7' deki sonuçlar gözlem sayısı 1000 ve 5 farklı dengesizlik oranları altındaki performans karşılaştırmaları verilmiştir. Tablo 3.7' de elde edilen sonuçlar incelendiğinde, yöntemler bazında en yüksek değerler Tablo 3.4' de olduğu gibi aynıdır. Farklı olarak, kesinlik değeri %30 dengesizlikte ROS değil SMOTE yönteminde daha yüksektir.

Tablo 3.7- 3.4 için elde edilen GBM algoritmasının değerleri iki tabloda da birbirine oldukça yakındır. İki tablo karşılaştırıldığında %30 dengesizlik hariç doğruluk, F-ölçütü, G-ortalama değerleri ve tüm dengesizlik oranları için seçicilik değeri tablo 3.7' de biraz daha yüksektir. AUC ve duyarlılık değerleri ise %10 ve %15 dengesizliklerde ve kesinlik değeri %10, %15 ve %20 dengesizlik oranlarında Tablo 3.7' de biraz daha yüksek olduğu görülmüştür. Ayrıca, AUC değeri %10 ve %15 dengesizliklerde, F-ölçütü değeri %25 dengesizlikte iki tabloda da aynı değeri vermiştir.

Tablo 3.4' de bulunan XGBoost ve AdaBoost değerleri ile karşılaştırılan Tablo 3.7' e göre GBM değerleri; %15 dengesizlik oranında F-ölçütü değeri ve %25 denge-

Tablo 3.7. *GBM dengesizlik oranları ve N=1000 için performans karşılaştırmaları-(1,5)*

Dengesizlik	Yöntem	Doğruluk	F	G-Ort	AUC	Duyarlılık	Seçicilik	Kesinlik
%10	ROS	**0.8202**	0.1904	0.4263	0.7272	0.2125	**0.8891**	0.1778
	RUS	0.6123	**0.2883**	**0.676**	**0.7345**	**0.7792**	0.5936	**0.178**
	ROSE	0.6419	0.2714	0.6379	0.7269	0.6759	0.6382	0.1745
	SMOTE	0.722	0.257	0.595	0.7308	0.479	0.7496	0.1774
%15	ROS	**0.7606**	0.2919	0.5272	0.7493	0.3379	**0.8344**	**0.2613**
	RUS	0.6247	**0.3931**	**0.6951**	0.7497	**0.8275**	0.5896	0.2593
	ROSE	0.6497	0.3675	0.6576	0.7444	0.7159	0.6384	0.2564
	SMOTE	0.7046	0.3509	0.629	**0.7507**	0.5441	0.7327	0.261
%20	ROS	**0.7257**	0.3902	0.591	0.7677	0.4412	**0.7976**	0.3534
	RUS	0.6559	**0.4963**	**0.7156**	0.7681	**0.8505**	0.6073	0.3525
	ROSE	0.6613	0.4722	0.6881	0.763	0.7795	0.6316	0.3481
	SMOTE	0.6975	0.4473	0.6626	**0.7692**	0.6138	0.7188	**0.3541**
%25	ROS	**0.7132**	0.4865	0.646	0.7869	0.5448	**0.7693**	**0.4413**
	RUS	0.6891	**0.5846**	**0.7389**	0.7874	**0.8775**	0.6263	0.44
	ROSE	0.6883	0.5585	0.7139	0.7819	0.8122	0.647	0.4351
	SMOTE	0.7041	0.5339	0.6944	**0.7878**	0.6786	0.7127	0.4412
%30	ROS	0.7196	0.5759	0.6919	0.8082	0.6345	**0.7569**	0.5301
	RUS	**0.7289**	**0.6632**	**0.7643**	**0.8086**	**0.8907**	0.6591	0.5307
	ROSE	0.7199	0.6447	0.7468	0.8033	0.8595	0.6592	0.5222
	SMOTE	0.7244	0.6207	0.7309	0.8084	0.7507	0.7132	**0.5311**

sizlik oranında ise G-ortalama değeri AdaBoost algoritmasından daha yüksek olduğu görülmüştür. Tersi olarak, %25 dengesizlikte duyarlılık değeri ve %30 dengesizlikte kesinlik değeri Tablo 3.7' de diğer iki algoritmadan daha düşük olduğu görülmüştür.

Simülasyon verisinin oluşturularak, yeniden örnekleme yöntemlerinin uygulanması ve tüm algoritma sonuçlarının alınmasına kadar geçen süre ele alınan iki ayrı "n.minobsinnode" parametre değerine göre kaydedilmiştir. Bu sürelerin alınması için R programlamada "proc.time" fonksiyonu kullanılmıştır. Tüm denemeler, dört çekirdekli Intel® Core™ i7-4790K CPU@3.60GHz, 8GB RAM özelliklerine sahip 20 bilgisayar kullanılarak kaydedilmiştir.

Farklı gözlem sayısı ve dengesizlik oranları için analizlerde geçen süreler Tablo 3.8, Tablo 3.9 ve Tablo 3.10' da olduğu gibi gösterilmiştir. Süreler "user", "system" ve "elapsed" olmak üzere 3 şekilde kaydedilmiştir. Buna göre: "user", R programlama da mevcut işlemler yürütülürken çekirdek dışında harcanan CPU süreyi, "system" işlem için gereken çekirdekte harcanan CPU süreyi ifade eder. Son olarak "elapsed" ise aynı anda çalışan diğer işlemleride dahil edilerek geçen toplam süreyi ifade eder.

Tablo 3.8' de görüldüğü gibi "n.minobsinnode değeri" (1,5) veya (4,10) olduğunda, gözlem sayısı 250 ve %10, %15 ve %20 dengesizlik oranlarında ROS yöntemi diğer yöntemlere göre daha uzun sürmüştür. Daha sonra SMOTE ve ROSE yöntemleri uzun sürmüştür. En kısa süren RUS yöntemidir. Dengesizlik oranı %25 ve %30 olduğunda ise bu sefer SMOTE yöntemi daha uzun sürmüş ve ROS ikinci sırada yer

Tablo 3.8. *Dengesizlik oranları ve N=250 için süre karşılaştırmaları*

Algoritma/süre	user		system		elapsed	
Dengesizlik %10	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1526.1	-	176.3	-	47118.7	-
ROS	2833.6	2869.8	375.9	382.6	61065.3	61088.4
ROSE	2153.4	2129.5	285.6	282.9	57860.5	57497.4
SMOTE	2266	2236.8	298	291.6	54767	54262.9
Dengesizlik %15	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1702.7	1719.3	209.3	194.5	50067.6	49977.9
ROS	3274.5	3271.7	438.3	432.2	62231.3	62532.2
ROSE	1985.6	2030.1	272.3	260.6	56598.9	57384.3
SMOTE	2641.2	2606.2	353.7	347.3	57946.6	57727.7
Dengesizlik %20	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1757.2	1745	218.7	205	52060.6	51303
ROS	3393.6	3438	471.2	469	62117.9	62546
ROSE	2073.3	2070.7	270.4	267.7	57759.4	57642.2
SMOTE	2973.3	2950	395.4	394	60129.4	61194
Dengesizlik %25	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1813.4	1799.1	216.8	214.1	52484.3	52134.3
ROS	3189.2	3192.3	444.5	449.3	61137.3	61031.4
ROSE	2039.5	2044.3	257.1	253.8	57387.7	57119.7
SMOTE	3285.3	3308.9	458.8	445.5	64153.4	63283.6
Dengesizlik %30	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1844.2	1837.3	214.8	221.7	53046.1	53038.8
ROS	2906.7	2873.3	407.9	413.7	60297.4	59231.6
ROSE	2017.0	2050.9	256.1	254.2	56548.6	57745.0
SMOTE	3487.8	3513.7	478.6	475.4	64889.5	64828.0

almıştır, daha kısa süren yine RUS yöntemidir.

Tablo 3.9' da görüldüğü gibi, gözlem sayısı 500 ve dengesizlik %10, %15 ve %20 olduğunda Tablo 3.8' de olduğu gibi ROS yöntemi diğer yöntemlere göre daha uzun sürmüştür. Daha sonra SMOTE ve ROSE yöntemleri gelmiştir. En kısa süren RUS yöntemidir. Dengesizlik oranı %25 ve %30 olduğunda ise Tablo 3.8' de olduğu gibi SMOTE yöntemi daha uzun sürmüş ve ROS ikinci sırada yer almıştır, daha kısa süren yine RUS yöntemidir.

Tablo 3.9. *Dengesizlik oranları ve N=500 için süre karşılaştırmaları*

Algoritma/süre	user		system		elapsed	
Dengesizlik %10	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1740.0	1736.9	212.5	213.6	51406.2	51454.8
ROS	4331.2	4255.8	538.1	528.2	72571.3	72221.0
ROSE	2442.0	2439.9	287.5	298.9	64396.6	64470.0
SMOTE	2751.1	2774.5	347.8	351.6	60321.4	60560.8
Dengesizlik %15	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1818.6	1876.0	239.4	251.6	52705.0	54081.2
ROS	4388.1	4402.1	581.9	583.9	71869.4	71715.6
ROSE	2371.6	2366.2	265.9	268.9	64378.0	64500.0
SMOTE	3336.0	3334.8	425.1	432.2	64876.9	65157.2
Dengesizlik %20	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1916.3	1919	239.2	240	55639.4	55977
ROS	4371.8	4419.1	602.4	607.1	71574.0	71380.7
ROSE	2382.3	2403.1	276.9	281.7	64714.7	64795.8
SMOTE	3803.2	3850	500.2	491	68376.8	69374
Dengesizlik %25	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1962	1977.0	245	259.8	56955	56318.4
ROS	4258.0	4232.2	597.8	595.1	69731.2	69753.2
ROSE	2321.2	2295.5	278.1	267.7	64355.0	64014.7
SMOTE	4256.2	4263.5	533.8	529.5	72562.6	72498.7
Dengesizlik %30	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	2006.4	2018.3	243.4	250.8	57692.8	58036.5
ROS	4070.4	4040.2	564.8	558.7	68659.3	68711.4
ROSE	2284.9	2294.7	261.4	262.3	63811.0	63890.9
SMOTE	4709.1	4689.7	613.1	595.6	76285.4	75877.7

Tablo 3.10' da görüldüğü gibi, gözlem sayısı 1000 ve dengesizlik %10, %15 ve %20 olduğunda Tablo 3.8' de ve Tablo 3.9' da olduğu gibi ROS yöntemi diğer yöntemlere göre daha uzun sürmüştür. Daha sonra SMOTE ve ROSE yöntemleri gelmiştir. En kısa süren RUS yöntemidir. Dengesizlik oranı %25 ve %30 olduğunda ise yine aynı şekilde SMOTE daha uzun sürmüş ve ROS ikinci sırada yer almıştır, daha kısa süren yine RUS yöntemidir.

Tablo 3.10. *Dengesizlik oranları ve N=1000 için süre karşılaştırmaları*

Algoritma/süre	user		system		elapsed	
Dengesizlik %10	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	1941.5	1960.8	251.9	251.2	55841.2	56308.3
ROS	5958.9	6063.1	634.4	637.8	90918.8	91565.3
ROSE	3040.6	3062	304.6	308	75472.5	75786
SMOTE	3609.6	3677.6	409.2	402.3	69246.0	68957.0
Dengesizlik %15	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	2053.9	2040	257.8	258	58977.9	58761
ROS	6051.1	6078.1	660.4	679.7	90457.4	90485.7
ROSE	2933.2	2915.8	290.5	293.2	75828.6	75368.0
SMOTE	4563	4633.0	513	512.2	77298	77054.0
Dengesizlik %20	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	2103.8	2125.8	266.1	267.7	60936.7	60782.1
ROS	5813.0	5926.7	672.3	685.6	88104.8	88323.1
ROSE	2986.0	2991.7	291.7	293.8	75937.6	75677.0
SMOTE	5320	5457.3	623	616.4	82621	83655.7
Dengesizlik %25	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	2254.0	2309.0	263.8	285.3	63213.8	65488.7
ROS	5535.6	5725.7	679.7	678.5	85215.5	86446.3
ROSE	2867.5	2957.9	268.3	278.9	75110.5	75887.4
SMOTE	6226.7	6134.8	683.8	670.5	91022.7	90957.6
Dengesizlik %30	(1,5)	(4,10)	(1,5)	(4,10)	(1,5)	(4,10)
RUS	2332.8	2324.3	281.9	278.2	65623.2	65574.0
ROS	5512.4	5559.0	671.4	664.3	84269.9	84390.9
ROSE	2931.2	2938.0	271.2	271.1	75922.6	75295.4
SMOTE	6856.1	6806.9	706.8	696.4	98015.4	98079.1

Gözlem sayısı arttıkça tüm yöntemlerde süre uzamıştır. Yeniden örnekleme yöntemleri kendi içinde değerlendirildiğinde, dengesizlik oranı azaldıkça RUS ve SMOTE yöntemlerinin süreleri artmıştır. ROS yönteminde genel olarak %25 ve %30 dengesizlik oranlarında süre azalmıştır. ROSE yönteminde ise birbirine yakın süreler elde edilmiş olup genel olarak dengesizlik oranıyla birlikte azalmıştır. Analiz süreleri değerlendirildiğinde, aşırı örnekleme yöntemlerinin daha uzun sürdüğü görülmüştür.

4. SONUÇ

Bu çalışma kapsamında dengesiz veri durumunda, XGBoost, GBM ve AdaBoost sınıflandırma yöntemleri ile ROS, RUS, ROSE ve SMOTE örnekleme yöntemleri beraber uygulandığında elde edilen performans sonuçları incelenmiştir. Bununla birlikte, sınıflandırma performansını etkileyen etmenler, seçilen örnekleme yöntemi, kullanılan gözlem sayısı ve sınıflandırma yöntemleri üzerinde durulmuştur. Analizlerde kullanılmak üzere simülasyon yoluyla veri üretilmiş ve analiz süreleri kaydedilmiştir.

Performans ölçütleri değerlendirildiğinde doğruluk değeri, gözlem sayısı 250 için dengesizlik azaldıkça RUS ve ROSE yöntemlerinde artarken, ROS yönteminde azalmıştır. SMOTE yönteminde ise doğruluk değerinin tüm dengesizlik oranlarında genel olarak birbirine yakın olduğu görülmüştür. Gözlem sayısının artmasıyla beraber doğruluk değeri %30 dengesizlik hariç genel olarak azalmıştır. Gözlem sayısı 500 için tüm dengesizliklerde doğruluk değeri ROS yöntemiyle kurulan XGBoost ve AdaBoost ile artarken, %25 ve %30 dengesizlik oranlarında ise tüm yöntemlerde artmıştır. Gözlem sayısı 1000 olduğunda, doğruluk değeri sadece ROSE yöntemiyle tüm dengesizlik oranlarında artarken, %30 dengesizlikte ise gözlem sayısı 500' de olduğu gibi tüm yöntemlerde artış göstermiştir.

F-ölçütü değeri, gözlem sayısı 250 için dengesizlik azaldıkça tüm yöntemlerde artış göstermiştir. Genel olarak gözlem sayısının artmasıyla ve dengesizlik oranın azalmasıyla artmaktadır. Gözlem sayısı 500 için, F-ölçütü değeri %20, %25 ve %30 dengesizlik oranlarında tüm yöntemlerde artmıştır. Gözlem sayısı 1000 olduğunda F-ölçütü tüm dengesizlik oranlarında artış göstermiştir.

G-ortalama ve AUC değerleri, gözlem sayısı 250 için dengesizlik azaldıkça tüm yöntemlerde artış göstermiştir. Gözlem sayısının artmasıyla ve dengesizlik oranın azalmasıyla artmaktadır. Bu iki performans ölçütü gözlem sayısı 500 ve 1000 olduğunda tüm yöntemlerde artış göstermiştir.

Duyarlılık değeri, gözlem sayısı 250 için dengesizlik azaldıkça tüm yöntemlerde artış göstermiştir. Genel olarak gözlem sayısının artmasıyla ve dengesizlik oranın azalmasıyla artmaktadır. Gözlem sayısı 500 için duyarlılık değeri dengesizlik azaldıkça artarken, ROS yöntemiyle kurulan AdaBoost algoritması sadece %15 dengesizlik oranında artış göstermiştir. Gözlem sayısı 1000 olduğunda ise duyarlılık değeri,

ROSE yöntemiyle kurulan GBM algoritması tüm dengesizlik oranlarında azalış göstermiştir.

Seçicilik değeri, gözlem sayısı 250 için dengesizlik azaldıkça SMOTE ve ROS yöntemlerinde azalmış, RUS yönteminde artmıştır. ROSE yönteminde ise seçicilik değerinin tüm dengesizlik oranlarında genel olarak birbirine yakın olduğu görülmüştür. Genel olarak gözlem sayısının artmasıyla azalmıştır. Gözlem sayısı 500 için seçicilik değeri sadece %25 ve %30 dengesizliklerde ROS yöntemiyle kurulan XGBoost ve AdaBoost algoritmalarında artmaktadır. Gözlem sayısı 1000 olduğunda seçicilik değeri ROSE yöntemiyle kurulan GBM algoritması ile tüm dengesizlik oranlarında artış göstermiştir.

Kesinlik değeri, gözlem sayısı 250 için dengesizlik azaldıkça tüm yöntemlerde artış göstermiştir. Dengesizlik oranın azalmasıyla artmış, gözlem sayısının artmasıyla genel olarak tutarsız kalmıştır. Gözlem sayısı 500 için kesinlik değeri, sadece %25 ve %30 dengesizliklerde tüm yöntemlerde artmaktadır. Gözlem sayısı 1000 olduğunda ise kesinlik değeri %25 dengesizlikte ROSE yöntemi hariç azalırken, diğer dengesizlik oranlarında artış göstermiştir.

Sonuç olarak, dengesizlik oranı %30 olduğunda doğruluk değeri RUS yöntemiyle kurulan GBM algoritmasında yüksek gelirken, diğer dengesizlik oranlarında ise ROS yöntemi daha yüksektir. RUS yöntemiyle kurulan AdaBoost algoritmasının F-ölçütü ve G-ortalama değerleri tüm dengesizlik oranlarında yüksektir, ikinci sırada ROSE yöntemi gelmektedir, en düşük F-ölçütü ve G-ortalama değerleri ROS yöntemine aittir. Yine aynı şekilde duyarlılık değeri, RUS yöntemiyle kurulan AdaBoost ile daha yüksek gelirken gözlem sayısı 500 olduğunda GBM algoritmasıyla kurulan ROSE yönteminin performansının da etkili olduğu görülmüştür. ROS yönteminin duyarlılık değerinin belirgin bir şekilde diğer yöntemlere göre daha düşük olduğu görülmüştür. AUC değeri değerlendirildiğinde, gözlem sayısı arttıkça RUS yöntemiyle kurulan AdaBoost algoritmasından sonra SMOTE yönteminin de performansının yükseldiği görülmüştür. Tüm dengesizliklerde ROS yönteminin seçicilik değeri belirgin şekilde yüksek çıkarken, bu değerde genel olarak XGBoost algoritmasının diğer algoritmalara göre performansının biraz daha yüksek olduğu görülmüştür. Yine aynı şekilde ROS yönteminin kesinlik değerinin de yüksek olduğu, bazı dengesizlik oranlarında ise SMOTE yönteminin performansının da etkili olduğu görülmüştür.

AdaBoost algoritması genel olarak GBM ve XGBoost algoritmalarına göre daha başarılı sonuçlar vermiştir. RUS yöntemi F-ölçütü, G-ortalama, AUC ve duyarlılık değerlerinde etkili olmuştur, ikinci olarak ROSE yöntemi gelmektedir. ROSE yöntemi GBM algoritması ile kullanıldığında duyarlılık ve seçicilik arasında belirgin farkın olduğu görülmüştür. Seçicilik ve kesinlik değerleri için genel olarak ROS ve SMOTE yöntemleri daha yüksek sonuç verirken, RUS bu yöntemlerden sonra gelmiştir. SMOTE yönteminin ise tüm dengesizlik oranlarında genel olarak dengeli sonuçlar verdiği görülmüştür. Simülasyon çalışması yapılarak oluşturulan veriye, uygulanan örnekleme yöntemleri genel olarak değerlendirildiğinde daha tutarlı ve yüksek sonuçlar veren yöntemin RUS olduğu görülmüştür.

5. TARTIŞMA

Literatürde, dengesiz simülasyon veri üretme ve sınıflandırma bazında çok fazla çalışma bulunmamakla beraber Aydın (2018) yaptığı bir çalışma örneği mevcuttur [9]. Bu çalışmayla benzer olarak çeşitli dengesizlik ve gözlem sayıları ile oluşturulan veriler 1000 kez tekrarlanarak ortalama sonuçlar alınmıştır. Farklı olan örnekleme yöntemleri ve üç boosting sınıflandırıcı kullanılarak (XGBoost, GBM, AdaBoost) performanslar karşılaştırılmıştır. Farklı gözlem sayısı ve dengesizlik oranları içeren simülasyon verilere, yeniden örnekleme sonucunda boosting algoritmalarının performansları ve süreleri karşılaştırılmıştır. Bunun sonucunda performans ölçütleri için elde edilen sonuçlar verilmiştir.

Aynı zamanda çıktı alıncaya kadar geçen sürenin kaydedilmesi yöntemler arasındaki süre farkını görebilmek açısından önemli olmuştur. Bunun bir örneği de algoritma düzeyinde kaydedilen süreler Singh vd. (2021) çalışmasında da mevcuttur. [63].

Sari vd. (2022) gerçek hayat verileri ile yaptıkları çalışmada aşırı ve alt örnekleme yöntemleri ROS, RUS ve aynı zamanda ikisinin birleştirildiği yöntemler kullanılmıştır. Örnekleme yapıldıktan sonra veriye AdaBoost, XGBoost gibi topluluk sınıflandırıcılarının olduğu 5 farklı sınıflandırıcı uygulanmıştır. Sonuç olarak bu çalışmada da olduğu gibi AdaBoost algoritması duyarlılık ve F-ölçütü değerlerinin daha yüksek olduğu görülmüştür [64].

Varotta vd. (2021) yaptıkları çalışmada LR, KNN ve SVM dahil 10 farklı sınıflandırıcı kullanılarak sınıflandırma yapmışlardır. Örnekleme yöntemlerinden benzer olarak RUS ve ROS dahil olmak üzere aşırı örnekleme ve alt örnekleme yöntemlerinden 5 farklı örnekleme yöntemi seçilerek yapılan sınıflandırma sonuçları değerlendirilmiştir. Performans ölçütü olarak AUC, F-ölçütü ve G-ortalama gibi ölçüler kullanılarak sonuçlar kaydedilmiştir ve bu çalışmada olduğu gibi RUS yönteminin daha başarılı olduğu görülmüştür [65].

Singh vd. (2022) ise RF, AdaBoost ve XGBoost algoritmalarının aralarında olduğu farklı sınıflandırıcılar kullanılarak yapılan çalışmada AdaBoost algoritmasının ROS yöntemi ile daha iyi sonuçlar verdiği görülmüştür [63]. Yine aynı şekilde Hanafy ve Ming (2021) farklı veri setleri ile yaptıkları çalışmada ROS, RUS, SMOTE

ve hibrit modeller ile yeniden örnekleme uygulandıktan sonra AdaBoost, XGBoost, CART gibi sınıflandırıcılar kullanılmıştır. Sonuç olarak, bu çalışmadan farklı olarak AdaBoost algoritmasının aşırı örnekleme yöntemi olan ROS yöntemi ile daha başarılı olduğu gözlenmiştir [66].

Performans sonuçları incelendiğinde, bu çalışma kapsamında da kullanılan örnekleme yöntemleri bazında litaratürde genel olarak ROS ya da SMOTE yöntemleri başarılı bulunmuştur [11, 15, 17, 22, 63, 66]. Buna rağmen, RUS yöntemi ya da alt örnekleme yöntemlerinin de başarılı olduğu sonuçlar da mevcuttur [9, 21, 65]. Sınıflandırıcı bazında değerlendirildiğinde boosting yöntemlerinden genel olarak, bu çalışmada da belirtilen, literatürde de örnekleri olan AdaBoost algoritmasının başarılı olduğu görülmüştür [23,63,64,66].

Gelecek çalışmalar için, simülasyon çalışması genişletilerek, daha hızlı bilgisayarlar kullanılarak gözlem sayısı, bağımsız değişken sayısı arttırılabilir veya farklı yeniden örnekleme ve sınıflandırma yöntemleri ile hiper parametre alanı genişletilerek performans sonuçları araştırılabilir.

KAYNAKÇA

[1] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W. P. (2002). SMOTE:Synthetic Minority Over-sampling Technique.*Journal of Artificial Intelligence Research*, 16, 321–357.

[2] Chawla, N. V., Lazarevic, A., Hall, L. O., Bowyer, K. W. (2003). SMOTEBoost:Improving prediction of the minority class in boosting. *The 7th European Conf on Principles and Practice of Knowledge Discovery in Databases*. Berlin:Springer, s.107-119.

[3] Han, H., Wang, W.Y. ve Mao. B.H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. International Conference on Advances in Intelligent Computing. In: Huang, DS., Zhang, XP., Huang, GB. (Ed.), *Advances in Intelligent Computing*. ICIC 2005. Lecture Notes in Computer Science, vol 3644. Springer, Berlin, Heidelberg.

[4] Menardi, G. ve Torelli, N. (2014). Training and assessing classification rules with imbalanced data. *Data mining and knowledge discovery*, 28(1), 92-122.

[5] Hanifah, F.S. (2015). SMOTE bagging algorithm for imbalanced dataset in logistic regression analysis (case: credit of bank X). *Applied Mathematical Sciences*, 9(138), 6857 - 6865.

[6] Sağlam, F. (2018).*Gürültülü Gözlemler Durumunda Dengesiz Veride Öğrenme İçin Yeni Bir Yaklaşım*. Yüksek Lisans Tezi. Samsun: Ondokuz Mayıs Üniversitesi, Fen Bilimleri Enstitüsü.

[7] Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4), 42–47.

[8] Kaur, P. ve Gosain, A. (2018). Comparing the behavior of oversampling and undersampling approach of class imbalance learning by combining class imbalance problem with noise. A.K. Saini vd., (Ed.), *ICT Based Innovations: Proceedings of CSI 2015*(s. 23-30). Springer.

[9] Aydın Haklı, D. (2018). *Sınıf dengesizliği sorununu çözmek için kullanılan algoritmaların farklı sınıflandırma yöntemlerinde performanslarının karşılaştırılması*. Bütünleşik Doktora Tezi. Ankara: Hacettepe Üniversitesi, Sağlık Bilimleri Enstitüsü.

[10] Brown, I. ve Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3), 3446-3453.

[11] Bach, M., Werner, A., Żywiec, J., Pluskiewicz, W. (2017). The study of under- and over-sampling methods utility in analysis of highly imbalanced data on osteoporosis. *Information Sciences*, 384, 174–190.

[12] Gameng H.A., Gerardo, B.B. ve Medina, R.P. (2019). Modified adaptive synthetic SMOTE to improve classification performance in imbalanced datasets. 6^{th} *IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Kuala Lumpur, Malaysia, s.1-5.

[13] Chakravarthy, A.D., Bonthu, S., Chen, Z., Zhu, Q. (2019). Predictive models with resampling: a comparative study of machine learning algorithms and their performances on handling imbalanced datasets. 18^{th} *IEEE International Conference On Machine Learning And Applications (ICMLA)*, Boca Raton, FL, USA. s.1492-1495.

[14] Par, Ö.E., Akçapınar Sezer, E. ve Sever, H. (2019). Small and unbalanced data set problem in classification. 27^{th} *Signal Processing and Communications Applications Conference (SIU)*, Sivas, Turkey, 2019, s.1-4.

[15] Önay Koçoğlu, F. ve Özcan, T. (2018). Dengeli-dengesiz veri seti dağılımının aşırı öğrenme makinesi yöntemi performansına etkisi. *Mühendislik ve Teknoloji Yönetimi Zirvesi 2018– ETMS2018*, İstanbul: İstanbul Teknik Üniversitesi ve Bahçeşehir Üniversitesi, s.201-208.

[16] Demir, S. ve Şahin, E.K., (2022). Evaluation of oversampling methods (OVER, SMOTE, and ROSE) in classifying soil liquefaction dataset based on SVM, RF, and Naive Bayes. *European Journal of Science and Technology*, (34), 142-147.

[17] Byeon, H. (2021). Development of a physical impairment prediction model for korean elderly people using synthetic minority over-sampling technique and XGBoost. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 12(1), 36-41.

[18] AL-Shatnwai, A.M. ve Faris, M. (2020). Predicting customer retention using XGBoost and balancing methods. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 11(7), 704-712.

[19] Cahyana, N., Khomsah, S. ve Aribowo, A.S., (2019). Improving imbalanced dataset classification using oversampling and gradient boosting. 5^{th} *International Conference on Science in Information Technology (ICSITech)*, IEEE, s.217-222.

[20] Bentéjac, C., Csörgő, A. ve Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3), 1937–1967.

[21] Mishra, S. (2017). Handling Imbalanced Data: SMOTE vs. random undersampling. *International Research Journal of Engineering and Technology (IRJET)*. 4(8), 317-320.

[22] Mohammed, R., Rawashdeh, J. ve Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: overview study and experimental results. 11^{th} *International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, s.243-248.

[23] Lenin, T. ve Chandrasekaran, N. (2021). Learning from imbalanced educational data using ensemble machine learning algorithms. *Webology* 18, s.83-195.

[24] Chen, W., Deng, C. ve Wang, S. (2020). Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. *Pattern Recognition Letters*, 136, 190-197.

[25] Lu, W., Li, Z. ve Chu, J. (2017). Adaptive ensemble undersampling-boost: a novel learning framework for imbalanced data. *Journal of Systems and Software*, 132, 272-282.

[26] Sun, Z., Song, Q., Zhu, X., Sun, H., Xu, B., Zhou, Y. (2015). A novel ensemble method for classifying imbalanced data. *Pattern Recognition*, 48(5), 1623-1637.

[27] Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., Asadpour, M., (2020). Boosting Methods for multi-class imbalanced data classification: An experimental review. *Journal of Big Data*, 7(70), 1-47.

[28] Sarmanova, A. (2013). *Veri madenciliğindeki sınıf dengesizliği sorununun giderilmesi*. Yüksek Lisans Tezi. İstanbul:Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[29] Hulse, J., Khoshgoftaar, T. ve Napolitano, A. (2007). Experimental perspectives on learning from imbalanced data. *In: Proceedings of the* 24^{th} *International Conference on Machine learning*, Corvallis, Oregon, USA.

[30] Freund, Y. ve Schapire, R.E. (1999). A Short Introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(771-780), s.1612.

[31] Turhan, S. (2019). *Kolektif öğrenmede sınıf dengesizliği problemi: hastalık tanısı sınıflandırma*. Yüksek Lisans Tezi. Muğla: Muğla Sıtkı Koçman Üniversitesi, Fen Bilimleri Enstitüsü.

[32] Görmez, B. (2021). *Adli bilişimde makine öğrenmesi: Makine öğrenmesi algoritmaları ile terör olaylarının tahmin edilmesi çalışması*. Yüksek Lisans Tezi. Ankara: Ankara Üniversitesi, Sağlık Bilimleri Enstitüsü.

[33] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984). *Classification and RegressionTrees*. Wadsworth, Belmont.

[34] Kuyucu, Y.E. (2012). *Lojistik regresyon analizi (LRA), yapay sinir ağları (YSA) ve sınıflandırma ve regresyon ağaçları (CART) yöntemlerinin karşılaştırılması ve tıp alanında bir uygulama*. Yüksek Lisans Tezi. Tokat: Gaziosmanpaşa Üniversitesi, Sağlık Bilimleri Enstitüsü.

[35] Loh, W.Y. (2008). Classification and Regression Tree Methods. *In Encyclopedia of Statistics in Quality and Reliability*, 1, 315-323.

[36] Kirazlı, C., Kan Kılınç, B. ve Yamaç, E., (2017). The relationship between parental defense intensity and nest site characteristics in eurasian magpie (PICA PICA L.) - an assessment with the classification methods. *Applied Ecology and Environmental Research* , 15(3), 1293-1308.

[37] Kan Kılınç, B. ve Mirgen, S. (2022). Mining housing features to classify housing unit price. Süleyman Demirel University: Fen Bilimleri Enstitüsü Dergisi, 26(3), 420-426.

[38] Mirgen, S. (2019). *Sıralı lojistik regresyon ve sınıflandırma ağaçlarının performans karşılaştırması*.Yüksek Lisans Tezi. Eskişehir: Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü.

[39] Kavzoğlu, T. ve Çölkesen, İ. (2010). Karar ağaçları ile uydu görüntülerinin sınıflandırılması: Kocaeli örneği. *Electronic Journal of Map Technologies*, 2(1), 36-45.

[40] Asfha, H.D. ve Kan Kılınç, B. (2018). Appraisal of Performance of three tree-based classification methods. In: Tez, M., von Rosen, D. (Ed.), *Trends and Perspectives in Linear Statistical Inference. Contributions to Statistics*. Springer, Cham.

[41] Doğaner, A. (2020). *Topluluk öğrenme yöntemleri ile renal hücreli karsinom'un tahmin edilmesi*. Doktora Tezi. Malatya: İnönü Üniversitesi, Sağlık Bilimleri Enstitüsü.

[42] Zhou, Z.H., Wu, J. ve Tang, W. (2002). Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137, 239-263.

[43] Freund, Y. ve Schapire, R. (1996), Experiments with a new boosting algorithm. *In: Proceedings of the Thirteenth International Conference on Machine Learning Theory*, MorganKaufmann Publishers Inc. San Francisco, s.148-156.

[44] Friedman, J., Hastie, T. ve Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 28(2), 337-407.

[45] Huang, F., Xie, G. ve Xiao, R. (2009). Research on ensemble learning. *International Conference on Artificial Intelligence and Computational Intelligence*, Shanghai, China, 2009, s.249-252.

[46] Efron, B. ve Tibshirani R.J. (1993). *An Introduction to the bootstrap*. Chapman and Hall, London. 430.

[47] Tuğ Karoğlu, T. (2018). *Lisans yerleştirme sınavında yerleşme başarısının karar ağaçlarına göre 'bagging' ve 'boosting'*. Doktora Tezi. Van:Van Yüzüncü Yıl Üniversitesi, Fen Bilimleri Enstitüsü.

[48] Efron, B., Tibshirani R.J. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical measures of statistical accuracy. *Statistical Science*, 1(1), 54-75.

[49] Erişkin, L. (2020). *Konum parametrelerinin bootstrap tahminleri ile ilişkili varyasyona ampirik bir bakış*. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 26(1), 174-183.

[50] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123-140.

[51] Yangın, G. (2019). *XGboost ve karar ağacı tabanlı algoritmaların diyabet veri setleri üzerine uygulaması*. Yüksek Lisans Tezi. İstanbul: Mimar Sinan Güzel Sanatlar Üniversitesi, Fen Bilimleri Enstitüsü.

[52] Janitza, S. ve Hornung, R. (2018). On the overestimation of random forest's out-of-bag error, *PloS One*, 13(8), 1-3.

[53] Sagi, Ö. ve Rokach, L. (2018). Ensemble learning: a survey. *Wiley Interdiscip. Rev.: Data Min. Knowl. DiscROSy*, 8(4), e1249.

[54] Bühlmann, P. (2010). Bagging, boosting and ensemble methods. *Handbook of Computational Statistics*. Berlin: Springer.

[55] Kurt, A. (2021). *Ağ tabanlı saldırı tespit sistemlerinde topluluk öğrenme yöntemlerinin karşılaştırmalı performans analizi*. Yüksek Lisans Tezi. Sakarya: Sakarya Üniversitesi, Fen Bilimleri Enstitüsü.

[56] Krithiga, R. ve Ilavarasan, E. (2020). Hyperparameter tuning of AdaBoost algorithm for social spammer identification. *International journal of pervasive computing and communications*, 17(5), 462-482.

[57] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189-1232.

[58] Rawi, R., Mall, R., Kunji, K., Shen, C-H., Knoung, P.D., Chuang, G-Y. (2017). PaRSnIP: sequence- based protein solubility prediction using gradient boosting machine. *Bioinformatics*, 34(7), 1092–1098.

[59] Chen, T., Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *KDD '16 Proceedings of the* 22^{nd} *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 13-17, San Francisco, CA, USA, s.785-794.

[60] Mitchell. R., Frank R. (2017). Accelerating the XGBoost algorithm using GPU computing. *PeerJ Computer Science*, 3(e127), 1-36.

[61] Yıldırım, E. (2022). *Hızlandırılmış Makine Öğrenmesi Algoritmaları İle Türkçe Sahte Haber Tespiti*. Yüksek Lisans Tezi. Karabük:Karabük Üniversitesi, Lisansüstü Eğitim Enstitüsü.

[62] Şahin Pir, M., (2022). *Dengesiz veri setlerinde sınıflandırma problemlerinin çözümünde melez yöntem uygulaması*. Yüksek Lisans Tezi. Bursa: Uludağ Üniversitesi. Fen Bilimleri Enstitüsü.

[63] Amit, S., Ranjeet, R. ve Abhishek, T. (2021). Credit card fraud detection under extreme imbalanced data: a comparative study of data-level algorithms. *Journal of Experimental & Theoretical Artificial Intelligence*. 34(4), 571-598.

[64] Sari, F. I. E., Edlim, F. W., Ramadhan, F. A., Muhtadin, Navastara, D.A. (2022). Performance analysis of resampling and ensemble learning methods on diabetes detection as imbalanced dataset. *2022 Fifth International Conference on Vocational Education and Electrical Engineering (ICVEE)*, Surabaya, Indonesia, 2022, s.1-5.

[65] Varotto, G., Susi, G., Tassi, L., Gozzo, F., Franceschetti, S., Panzica, F. (2021). Comparison of resampling techniques for imbalanced datasets in machine learning: Application to epileptogenic zone localization from interictal intracranial EEG recordings in patients with focal epilepsy. *Front Neuroinform*, 15(715421), 1-19.

[66] Hanafy, M. ve Ming, R. (2021). Improving imbalanced data classification in auto insurance by the data level approaches. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 12(6), 493-498.

http-1:https://ingeniusprep.com/blog/2020-college-acceptance-rates/ (Erişim tarihi: 06.06.2020).

ÖZGEÇMİŞ

Adı-Soyadı : İlkay Tuğ
Yabancı Dili : İngilizce

Eğitim ve Mesleki Geçmişi:

- 2015-2019, Anadolu Üniversitesi, Fen Fakültesi, İstatistik Bölümü, Lisans

Yayınlar:

- Mathematical Methods for Engineering Applications/Performance of Machine Learning Methods Using Tweets Yayın Evi: Springer Yazar: Tuğ İlkay, KAN KILINÇ BETÜL (Bölüm/Ünite yazarlığı)

Konferanslar ve Bildiriler:

- Performance of Machine Learning Methods Using Tweets Basım Tarihi: 27.07.2022 Etkinlik Adı: III. International Conference on Mathematics and its Applications in Science and Engineering-ICMASE 2022ICMASE 2022 Yazar: Tuğ İlkay, KAN KILINÇ BETÜL Ülke: ROMANYA
- Emotion Classification based on Twitter Data Basım Tarihi: 27.12.2021 Etkinlik Adı: 10th International Eurasian Conference on Mathematical Sciences and Applications Yazar: KAN KILINÇ BETÜL, Tuğ İlkay Ülke: TÜRKİYE
- The Examination of Real Estate Prices in Istanbul by Using Hybrid Hierarchical K-Means Clustering Basım Tarihi: 10.12.2019 Etkinlik Adı: YBIS2019 Yazar: KAN KILINÇ BETÜL, Tuğ İlkay

Sunumlar:

- "SQLite veritabına bağlanma SQL ve dplyr ile sorgulama", https://www.youtube.com/watch?v=bBZuQD8LzCk/ (Erişim tarihi: 29.03.2023). R-ladies Eskişehir.

Projeler:

- Proje no:20ADP179, Proje adı: "Veri Madenciliği Kullanarak Toplumsal Sağlık Sorunlarının İncelenmesi", Görevi: Bursiyer (7ay).