Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-16
Sayı-2
2023

Yazılım Hata Tahmininde Farklı Alt Örnekleme ve Üst Örnekleme Yöntemlerinin Karşılaştırılması

Comparison of Different Oversampling and UndersamplingMethods in Software Defect Prediction

Özge ŞEN KAYA
Eskişehir Osmangazi Üniversitesi
Bilgisayar Mühendisliği
Eskişehir, Türkiye
ozgeesenn1995@gmail.com
ORCID: 0000-0002-4713-7536

Sinem OZKURT KESER
Eskişehir Osmangazi Üniversitesi
Bilgisayar Mühendisliği
Eskişehir, Türkiye
sbozkurt@ogu.edu.tr
ORCID: 0000-0002-8013-6922

Öz

Bilgisayarları ve makineleri çalıştırmak üzere belirli fonksiyonların işletilebilmesi için kullanılan komutlar bütünü yazılım olarak adlandırılmaktadır. Günümüzde birçok alanda yapılan faaliyetler ve kullanılan uygulamalar, içerisinde farklı algoritmalarla tasarlanmış yazılımlar barındırır. Bu yazılımların kusursuz ve ihtiyaçları karşılayacak şekilde olması büyük önem teşkil etmektedir. Yazılımın kalitesi, yazılımın içerisinde hata barındırmaması hem yazılımı geliştiren kişilerin hem de yazılımı kullanan son kullanıcıların önem verdiği konulardır. Yazılım hata tahmini doğası gereği dengesiz sınıf problemi içerir. Bu çalışmada, öncelikle dengesiz sınıf problemi çözülmeye çalışılmıştır. Bu doğrultuda, farklı alt örnekleme ve üst örnekleme yöntemleri, literatürde araştırmacıların kullanımına açık NASA'nın PROMISE veri deposundan alınan CM1, KC1, KC2, JM1 ve PC1 veri kümelerinin üzerinde uygulanmıştır. Yazılım hata tahmini aşamasında ise farklı sınıflandırma algoritmaları karşılaştırılarak her bir veri kümesi için en uygun algoritma belirlenmiştir. Deney sonuçlarında on farklı örnekleme yöntemi ile veri kümelerindeki dengesiz sınıf problemi giderilmiş; on üç farklı sınıflandırma algoritması ile sınıflandırma işlemi yapılmıştır. 0,92 oranında AUC ölçütü ile en iyi sınıflandırma sonucu PC1 veri kümesinde elde edilmiştir. Bu çalışma ile yazılım hata tahmininde örnekleme yöntemleri ve uygun sınıflandırıcılar ile hata tahmininin başarımının daha iyi olabileceği gösterilmiştir. Elde edilen sonuçlar, literatürde yapılan çalışmalar ile karşılaştırılarak önerilen yöntemin üstünlüğü ve etkinliği kanıtlanmıştır.

Gönderme, düzeltme ve kabul tarihi: 15.01.2023 - 02.05.2023 – 22.05.2023
Makale türü: Araştırma

Anahtar sözcükler: Yazılım hata tahmini, Sınıf dengesizliği problemi, Alt örnekleme yöntemleri, Üst örnekleme yöntemleri, Sınıflandırma

Abstract

The set of commands used to operate certain functions to operate computers and machines is called software. Today, activities and applications used in many fields contain software designed with different algorithms. It is of great importance that these softwares are perfect and in a way thatmeets the needs. The quality of the software and the absenceof errors in the software are issues that both the developers of the software and the end users of the software attach importance to. Software defect prediction inherently involvesan imbalanced class problem. In this study, first of all, the imbalanced class problem was tried to be solved. In this direction, different undersampling and oversampling methodswere applied on the CM1, KC1, KC2, JM1 and PC1 datasets taken from NASA's PROMISE data repository, which is open toresearchers in the literature. In the software defect prediction phase, different classification algorithms were compared and the most suitable algorithm was determined for each data set.In the experimental results, the imbalanced class problem in the datasets was resolved with ten different sampling methods; classification was done with thirteen different classification algorithms. With an AUC of 0.92, the best classification result was obtained in the PC1 dataset. With this study, it has been shown that the performance of defectprediction can be better with sampling methods and appropriate classifiers in software defect prediction. The results obtained were compared with the studies in the

literature and the superiority and effectiveness of the proposed method were proven.

Keywords: Software defect prediction, Class imblance problem, Oversampling methods, Undersampling methods, Classification

1. Giriş

Bilgisayar ve bilişim teknolojileri günlük yaşamda birçok alanda kullanılan ve kullanımı da her geçen gün yaygınlaşan teknolojilerdir. Bu teknolojilerin işletilebilmesi ve içerisinde yer alan belirli fonksiyonlarının çalışabilmesi için kullanılan komutlar bütünü yazılım olarak adlandırılmaktadır. Her yazılım farklı ihtiyaçları karşılaması için farklı algoritmalar ile tasarlanır. Bu yazılımların kusursuz ve ihtiyaçları karşılayacak şekilde olması büyük önem teşkil eder. Yazılım kalitesi, yazılımın içerisinde hata barındırmaması hem yazılımı geliştiren kişilerin hem de yazılımı kullanan son kullanıcının önem verdiği konulardır.

Yazılım kalite standartları olarak adlandırılan ISO 25010, yazılım kalitesini belirlemede temel ögedir. Bir yazılım kalitesi standart içerisinde belli başlıklara bölünerek değerlendirilmiştir [1]. Taşınabilirlik, bakım yapılabilirlik, güvenirlik, güvenlik, başarım, uyumluluk, işlevsel uygunluk ve kullanılabilirlik gibi sekiz kalite özelliğini içerisinde barındırır. Bu özellikler kapsamında bir yazılımın hatasız ve doğru çalışır durumda olması kalitesini göstermektedir.

Yazılımlar, içerisinde bazı mantıksal kusurlar barındırabilir. Bu kusurlar yazılımın kalitesini ve son kullanıcının yazılımı tercih etme düzeyini düşürebilir. Bir yazılımın kalitesi, yazılım son kullanıcıya ulaşmadan önce yapılan yazılım testleri ile ölçülür. Bu testler ile yazılımın ana fonksiyonlarının çalışabilirliği, yazılımda meydana gelebilecek hata durumları tespit edilir. Gerçekleştirilen yazılım sınamaları ile yazılımda ortaya çıkabilecek hatalı durumların erken aşamada önüne geçilmesi büyük önem taşımaktadır.

Yazılım hatalarını otomatik olarak tespit edebilmek amacıyla makine öğrenmesi yöntemleri önem teşkil etmektedir. Makine öğrenmesi yöntemleri birçok alanda kullanılan, literatür çalışmalarında da oldukça tercih edilen yöntemlerdir. Makine öğrenmesi yöntemleri insan aracılığı olmadan otomatik olarak büyük oranda veri analizini sağlamaktadır [2]. Yazılım test aşamasında makine öğrenmesi yöntemleri de birçok literatür çalışmasında incelenmiş oldukça popüler bir konudur. Literatür çalışmaları incelendiğinde farklı makine öğrenmesi yöntemleri ile yazılım hata tahmininde başarılar elde edilmiştir.

Bu çalışmada, yazılım hatalarının tespit edilmesi problemi üzerinde makine öğrenmesi yöntemleri ile çalışılmıştır. NASA'nın PROMISE veri deposunda bulunan CM1, KC1, KC2, JM1 VE PC1 veri kümelerinde yer alan yazılım hatalı ve hatasız yazılım kayıtları çalışmanın veri kaynağını oluşturmuştur. Bu veri kümelerinde hatalı ve hatasız kayıtların eşit dağılmaması sınıf dengesizliği problemini meydana getirmiştir. Dengesiz sınıf problemini çözmek için alt ve üst örnekleme yöntemleri uygulanmıştır. Bu doğrultuda Sentetik Azınlık Örneklem Arttırma Yöntemi (SMOTE – Synthetic Minority Oversampling Technique), Rastgele üst örnekleme, Borderline SMOTE, SVM- SMOTE, ADASYN, SMOTE + Tomek, SMOTE-NC, Rastgele alt örnekleme yöntemi, Küme Merkezli Örnekleme (Cluster Centroid), SMOTE+ENN yöntemleri kullanılmıştır. Sınıflandırma aşamasında ise K-En Yakın Komşu Algoritması, Destek Vektör Makineleri, Karar Ağacı Algoritması, Rastgele Orman Algoritması, Adaboost, Gradyan Artırma Algoritması, Ekstra Gradyan Artırma algoritması, Hafif Gradyan Artırma algoritması, Catboost algoritması, Naive Bayes Algoritması, Lineer Diskriminant Analizi, Kuadratik Diskriminant Analizi, Ekstra Ağaçlar algoritmaları ile deneyler gerçekleştirilmiştir. Doğruluk ve AUC/AUROC başarım ölçütleri kullanılarak algoritmalar değerlendirilmiştir. Deneylerde, her bir veri kümesi için en iyi başarım gösteren algoritmalar tespit edilmiştir.

2. Yayın İncelemesi

Yazılımda yer alan hataları tahmin etmek hem yazılımın kalitesinin artırılması hem de son kullanıcıya ulaşmadan önce doğru ve istenilen şekilde sunmak açısından önemlidir. Yazılım hata tahmininde sınıflandırma, derin öğrenme ve makine öğrenmesi gibi yöntemler kullanılır. Makine öğrenmesi algoritmaları kullanılarak sınıflandırma işlemi yapan birçok çalışma literatürde yer almaktadır.

Aydilek [3] SMOTE algoritması ile NASA PROMISE veri deposunda yer alan veri kümelerindeki sınıf dengesizliğinin iyileştirilmesi üzerinde çalışmıştır [1]. Bu çalışmada çeşitli karar ağacı tabanlı algoritmalar kullanarak iyileştirilmiş bilgi kazancı sonuçları elde etmiştir. Yazılım hata tahmininde kullanılan metrikler ile bilgi kazancı yönünden daha yüksek sayısal değerli sonuçları elde edilmiştir. Bu çalışma ile, kural tabanlı sınıflayıcı ve karar ağaçlarının başarımlarının yazılım hata tahmininde ön plana çıktığı ve bazı eksikliklerin de mevcut olduğu görülmüştür. Bu eksikliklerin ve problemlerin giderilmesi için bu çalışma yapılarak yüksek başarıda sonuçlar literatüre katkı sağlamıştır.

Çetiner [4] yazılım hata tahmin sistemleri hakkında on farklı makine öğrenmesi algoritmasını kullanarak bu algoritmaların karşılaştırmalı analizini yapmıştır. Bu analiz için NASA'nın PROMISE veri kümesinden CM1, KC1, KC2, JM1 ve PC1 veri kümelerinden yararlanmıştır. Yapılan çalışmada veri setleri ön işleme sonrasında normalizasyon işleminden geçirilmiş daha sonra çapraz doğrulama yöntemi ile makine öğrenmesi algoritmaları eğitilmiştir. Çalışmada doğruluk başarım ölçünü kullanılarak sonuçlar kıyaslanmıştır. Yapılan çapraz doğrulama yaklaşımı ile rastgele orman öğrenme modelinin birleştirilmesinde elde edilen sonuç, bu şekilde önerilen diğer modellere göre daha iyi bir doğruluğa ulaşmıştır.

Sun ve ark. [5] tarafından yazılım hata tahmini veri kümesinde yer alan sınıf dengesizliği problemi örnekleme, maliyete duyarlı örnekleme, torbalama ve arttırma gibi yöntemler ile modellenmiştir. Makine öğrenmesi algoritmaları ile yazılım hata tahmini yapmıştır. Deneylerde NASA veri kümesi üzerinde 4 farklı sınıflandırma algoritması karşılaştırılmıştır.

Önerilen yöntemin rastgele üst örnekleme, rastgele alt örnekleme, SMOTE, maliyete duyarlı öğrenme gibi yöntemlerden çok daha önemli ölçüde iyi başarım gösterdiği sonucu elde edilmiştir.

Eivazpour ve ark. [6] yazılım hata tahmininde veri kümesinin dengesiz dağılımının makine öğrenimi algoritmaları için zorlayıcı bir etkeni olduğunu belirterek, veri dengesizliği problemi üstünde durmuştur. Bu çalışmada, veri kümesindeki azınlık sınıfından örnekleme teknikleri ile artırılmış bir eğitim kümesi elde edilmiştir. Dengesiz veri kümesinin örneklenmesi için Rastgele Örnekleme, Sentetik Azınlık Örneklem Arttırma Yöntemi (SMOTE), Borderline-SMOTE ve ADASYN gibi örnekleme teknikleri kullanılmıştır. Daha sonra önerilen yöntemde hangi modelin kullanılan tekniklerden hangisi ile daha iyi başarım elde ettiği analiz edilmiştir. Elde edilen sonuçlarda önerilen yöntemin, literatürdeki örnekleme yöntemlerine göre daha iyi başarım gösterdiği görülmüştür.

Malhotra ve ark. [7] NASA PROMISE veri kümesi üzerindeki sınıf dengesizliği problemi için SMOTE, SVM-SMOTE ve ADASYN yöntemlerini kullanmış, daha sonra elde edilen sonuçlarda Lineer Diskriminant Analizi yöntemi ile verilerini iyileştirmiştir. Bu veri kümesi üzerinde ise Naive Bayes ve Destek Vektör Makineleri (SVM) sınıflandırma algoritmaları ile yazılım hataları tahmin edilmiştir. Yapılan çalışmada elde edilen sonuçlar ile, SVM-SMOTE yönteminin Lineer Diskriminant Analizi ile iyileştirilmesi mekanizmasının yapılan tahmin kalitesini iyileştirmede önemli bir katkısı olduğu sonucuna varılmıştır.

Choirunnisa ve ark. [8] yapmış oldukları çalışmada NASA veri kümesi kullanılmış ve bu veri kümesinde ön işleme olarak veri dengeleme yöntemini kullanmışlardır. Ön işleme yapılmış veri kümesinde sınıf dengesizliği problemi için SMOTE, Borderline SMOTE, Safe-SMOTE ve A-SUWO (adaptive semi-unsupervised oversampling) yöntemleri uygulanmıştır. Veri kümesindeki sınıf dengesizliği problemi iyileştirildikten sonra K-En Yakın Komşu Algoritması, SVM, Rastgele Orman ve J48 sınıflandırma algoritmaları ile yazılım hata tahmininde bulunulmuştur. En gelişmiş veri dengeleme tekniği olarak nitelendirdikleri A-SUWO yöntemi ile yazılım hata tahmininde yeni bir yaklaşım önermişler ve bu yaklaşımı denetimli sınıflandırma algoritmaları ile birleştirmişlerdir.

Elahi ve ark. [9] çalışmalarında veri kümesindeki sınıf dengesizliği problemi için rastgele üst örnekleme ve komşu temizlik kuralı yöntemlerini önermişlerdir. Bu yöntemler ile veri kümesinin sınıf dengesizliği problemi çözüldükten sonra K- En Yakın Komşu Algoritması, Naive Bayes, Karar Ağacı, Lineer Regresyon, Ortalama Mutlak Hata (MAE) algoritmaları ile sınıflandırma işlemi yapılmıştır. Çalışmada elde edilen başarım sonuçlarının sınıf örtüşme yönteminin kullanıldığı veya kullanılmadığı ve yeniden örnekleme yönteminin de kullanılmadığı çalışmalara kıyasla çok daha iyi başarım gösterdiği açıklanmıştır.

Goyal [10] yazılım hata tahmininde sınıf dengesizliği problemini alt örnekleme yöntemi ile gidermiştir. Yapılan çalışmada eğitim kümesinde sınıf dengesizliği problemini gidermek için çeşitli alt örnekleme yöntemlerini kullanmıştır. Eğitim veri kümesinde uygulanan bu yöntemler ile sınıflandırma modeli eğitildikten sonra test veri kümesi kullanılarak sınıflandırma modelinin başarımı elde edilmiştir. Sınıflandırma aşamasında Yapay Sinir Ağları (ANN), Destek Vektör Makineleri (SVM), Karar Ağacı (DT), KNN, Naive Bayes (NB) algoritmaları kullanılmıştır. Bu çalışmada sınıf dengesizliği problemini etkili bir şekilde ele alarak yeni bir alt örnekleme yöntemi önerilmiş ve literatüre katkıda bulunulmuştur. Bu yöntem 3 alt örnekleme tekniği ile kombinasyon yapılarak önerilmiştir. Bu yöntemler rastgele alt örnekleme, tomek-link örnekleme ve ıskalama alt örnekleme yöntemleridir.

Jacob ve ark. [11] NASA Promise veri kümesindeki sınıf dengesizliği problemi için öncelikli olarak veri kümesine 2 farklı ön işleme uygulamıştır. Bu ön işlem yöntemleri "Wrapper-Based Feature Selection" ve "Heuristic-Based Feature Selection" yöntemleridir. Sınıf dengesizliği problemi, bu yöntemler ile çözüldükten sonra; Naïve Bayes, Lineer Regresyon, Çok Katmanlı Perceptron, Destek Vektör Makineleri, K-En Yakın Komşu Algoritması, Torbalama, AdaBoost, Rastgele Orman (RF), Karar Ağacı algoritmaları ile sınıflandırma işlemi yapılmıştır. Bu çalışma ile oylama tabanlı topluluk modellerinin daha küçük kümeleri daha fazla etkiledikleri ve daha iyi başarım gösterdikleri çıkarımı yapılmıştır.

Çetiner ve ark. [12] NASA Promise veri kümesi içerisinde yer alan CM1, KC1, KC2, JM1 ve PC1 veri setleri ile 10 farklı makine öğrenmesi algoritmasından yararlanarak yazılım hata tahmininde makine öğrenmesini algoritmalarını kıyaslamıştır. Bu çalışmada SVM, KNN, NB, RF, Ekstra Ağaçlar, Adaboost, Gradyan Arttırma, Torbalama ve Multi Layer Perceptron algoritmalarını karşılaştırmışlardır. Yapılan karşılaştırma sonucunda algoritmaların yazılım hata tahmininde kullanılan veri setlerinden PC1 veri kümesi için daha ortalama doğruluk oranlarının elde edildiğini göstermiştir.

Pelayo ve ark. [13] yapmış oldukları çalışmada dengesiz veri kümelerinde makine öğrenmesi üzerinde durarak yazılım hata tahmini konusunu incelemiştir. Bu çalışmada SMOTE yöntemine odaklanılarak veri kümesinin iyileştirilmesi konusunda çalışılmıştır. Bu çalışmada, yazılım hata tahmininde çok az ilgi gören, dengesiz verileri öğrenmek için kullanılan katmanlaşma (stratification) incelenmiş; SMOTE yönteminin hata eğilimli modüllerin iyileştirilmesinde rolü araştırılmıştır.

Aleem ve ark. [14] makine öğrenmesi tekniklerinin yazılım hata tahmininde yararlı olduğunun üstünde durarak kullanıma açık veri setlerinde karşılaştırmalı analiz yapmışlardır. Makine öğrenmesi yöntemlerinin yazılım hata tahmininde iyi başarım gösterdiğini açıklamışlardır. Nöro-bulanık (neuro-fuzzy) teknikleri ve yazılım aracılarının test senaryolarını etkinliğini artırmada önemini ortaya koyarak doğru sonuçlar elde etmede birden fazla tekniğin birleştirilebileceği bu çalışmada önerilmiştir. Hata tespitlerinde destek vektör makineleri, MLP ve torbalama tekniklerinin de veri kümelerinde iyi başarım gösterdiği ortaya konmuştur.

Wang ve ark. [15] sınıf dengesizliği problemini ele alarak yazılım hata tahminine fayda sağlayıp sağlamayacağını tartışmışlardır. Yeniden örnekleme, topluluk algoritmaları olmak üzere farklı sınıf dengesizliği yöntemlerini araştırmışlardır. Bu çalışmada farklı sınıf dengesizliği yöntemlerinin araştırılarak incelenmesi yazılım hata

tahmininde literatüre katkı sağlamıştır.

İbrahim ve ark. [16] arama algoritmaları ve rastgele orman algoritmasını kullanarak yazılım hata tahmininde yüksek başarım elde edebilmek için bir yaklaşım önermişlerdir. Önerilen bu yaklaşımın başarımı, yayın taramasında bahsedilmiş olan diğer yaklaşımlar ile değerlendirilmiş ve önerilen yaklaşımın literatüre göre çok daha iyi sonuçlar elde ettiği görülmüştür.

Akmel ve ark. [17] yapmış oldukları çalışmada, yazılım hata tahmininde yayınlardaki çalışmaları inceleyerek açıklamışlardır. Hazırladıkları özet tabloda, yıl bazlı yapılan çalışmaları kullanılan yöntemleri, teknikleri yaklaşımları analiz etmişlerdir. Bu tabloda yapılan çalışmaları sınıflandırarak ayırmışlardır. Sınıflandırma tabanlı yaklaşımlar, kümeleme tabanlı yaklaşımlar ve topluluk öğrenmesi tabanlı yaklaşımlar olmak üzere analizi yapmışlardır. Bu çalışmanın yazılım hata tahmini çalışmalarında, kaynakların tahsisine rehberlik edecek yapıda olması literatüre katkı sağlamıştır.

Naidu ve ark. [18] yapmış oldukları çalışmada, yazılım kusurlarını gruplandırmak için karar ağacı tabanlı bir sistem önermiştir. Bu sistem içerisinde ID3 ve C4.5 algoritmalarını kullanarak sınıflandırma işlemi yapmıştır. Yapılan sınıflandırma işlemi sonrasında model madenciliği tekniği kullanarak sınıflandırılan kusur modelleri ölçmüştür. Çalışmada zaman ve maliyeti azaltmak için toplam kusur sayısına odaklanılmış ve beş farklı parametre ile sınıflandırma yapılmıştır. Model madenciliği tekniği ile yazılım kusurları sınıflandırılmıştır.

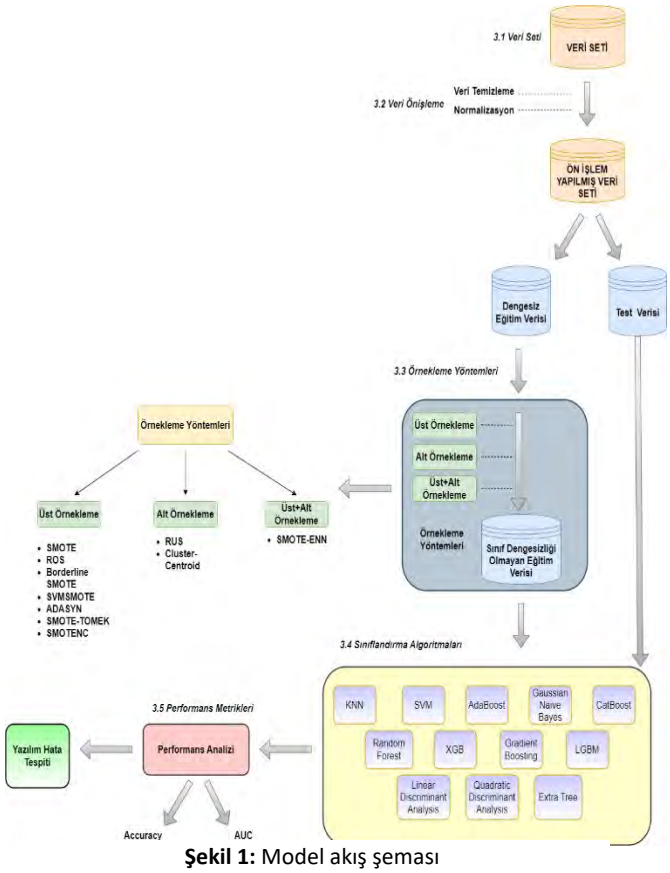
Yazılım hata tahmininde örnekleme ve sınıflandırma algoritmaları ile NASA PROMISE veri kümesini kullanan son 4 yılda yapılan çalışmalar Çizelge-1 içerisinde özetlenmiştir. Bu çizelgede örnekleme yöntemleri, sınıflandırma algoritmaları ve başarım metrikleri detaylandırılmıştır.

3. Materyal ve Yöntem

Bu bölümde çalışma kapsamında kullanılan veri kümesine ve model aşamalarına değinilip akış modeli ayrıntılı biçimde sunulacaktır. Gerçekleştirilen bu çalışmada genel anlamda veri ön işleme, örnekleme, sınıflandırma ve başarım analizi aşamaları bulunmaktadır. Çalışmada kullanılan veri kümesi öncelikli olarak belirli ön işleme adımlarından geçirilmiştir. Bu önişleme adımlarından sonra veri kümesi üzerinde çalışılabilecek duruma gelmiştir. Daha sonra veri kümesi eğitim ve test verisi olarak ikiye ayrılmıştır. Burada veri kümesinde bulunan sınıf dengesizliği problemi nedeniyle eğitim veri kümesinde örnekleme yöntemleri kullanılmıştır. Kullanılan örnekleme yöntemleri sınıf dengesizliği problemini gidermiş; veri setlerinden elde edilecek sonuçların daha doğru olmasını sağlamıştır. Seçilen farklı alt ve üst örnekleme yöntemleri ile sınıf dengesizliği problemi çözülen eğitim kümesine ve ana veri kümesinden alınan test veri kümesine 12 farklı makine öğrenmesi sınıflandırma algoritması ile sınıflandırma işlemi yapılmış, sınıf dengesizliği probleminde en çok kullanılan başarım ölçünleri ile başarım analizi yapılmıştır. Her veri kümesindeki hatalı yazılım kayıtları da göz önüne alınarak elde edilen değerler analiz edilerek kıyaslaması yapılmıştır. Veri kümelerinin hangi örnekleme yöntemi ve hangi sınıflandırma algoritması ile daha iyi başarım gösterdiği tespit edilmiş sonuçları değerlendirilmiştir. Önerilen yöntemin model akışı Şekil-1 içerisinde ayrıntılı biçimde özetlenmiştir.

Çizelge -1: **Özet Yayın Analizi**

Referans	Yıl	Sınıf Dengesizliği içinKullanılan Yöntem	Veri Kümesi	Başarım Ölçünleri	Kullanılan Algoritmalar
[3]	2018	SMOTE algoritması	NASA PROMISE	F-Ölçütü, Tutturma, Bulma, G-Ortalama, Özgüllük	JRip C4.5 PART
[4]	2020		NASA PROMISE	Doğruluk, Bulma, Tutturma	NB, Karar Ağacı, KNN, SVM, Gradyan Artırma, Artırma, RF, Ekstra Ağaç, MLP
[5]	2012	1-1, Orig, Rastgele alt örnekleme, Rastgele üst örnekleme, SMOTE, Cost,Torbalama, Artırma	NASA PROMISE	AUC	Rastgele Orman, C4.5, Ripper, Naive Bayes
[6]	2019	Rastgele üst örnekleme, SMOTE, Borderline-SMOTE,ADASYN	NASA PROMISE	AUC	Karar Ağacı, Rastgele Orman
[7]	2021	SMOTE, SVM-SMOTE,ADASYN	NASASDP	F-Ölçütü, AUC	Destek VekD Makineleri, Naive Bayes
[8]	2018	SMOTE, Borderline-SMOTE, SafeSMOTE, A-SUWO	NASA PROMISE	Doğruluk, TPR, FPR	KNN, SVM, Rastgele Orman, J48
[9]	2021	Rastgele üst örnekleme, NCL	NASA PROMISE	AUC	KNN, Naive Bayes, Karar Ağacı, Lineer Regresyon, MAE
[10]	2022	Çeşitli alt örneklemeteknikleri	NASA PROMISE	Doğruluk, AUC	Yapay Sinir Ağları (ANN), SVM, Karar Ağacı, KNN, Naive Bayes
[11]	2021	“Wrapper-Based Approach “ “Heuristic Based Approach “	NASA PROMISE	Doğruluk, Kesinlik, Duyarlılık, F-Ölçütü, AUC	Naive Bayes, Lineer Regresyon, MLP, SVM, KNN, Torbalama, AdaBoost, Rastgele Orman, Karar Ağacı



Şekil 1: Model akış şeması

3.1 Veri Kümesi

Yazılım hata tahmininde modelleme yapabilmek için NASA PROMISE veri deposunda yer alan 5 veri kümesi kullanılmıştır. Bu çalışmada, NASA PROMISE veri deposunda yer alan 5 veri kümesinden yararlanılmıştır [19]. Veri deposu içerisinden CM1, JM1, KC1, KC2, PC1 veri kümeleri seçilmiştir. Bu veri kümelerine ait 22 parametrenin açıklaması Çizelge-2 içerisinde detaylandırılmıştır. Seçilen bu veri kümeleri içerisinde, hatalı ve hatasız olarak etiketlenmiş kayıtları barındırır. Hatalı kayıtlar ilgili kısımda değişiklik yapılması gerektiğini ifade eder. Kullanılan veri kümelerinden CM1, JM1 ve PC1 veri kümesi C yazılım dili ile; KC1 ve KC2 ise C++ yazılım dili ile geliştirilmiştir.

Çalışmada kullanılan veri kümeleri içerisinde toplam kayıt sayısı en fazla olan veri kümesi JM1 iken, toplam kayıt sayısı en az olan veri kümesi CM1'dir. Veri kümelerindeki hatalı kayıt sayıları incelendiğinde ise en fazla hatalı kayıt içeren veri kümesi JM1, en az hatalı kayıt içeren veri kümesi ise PC1'dir. Veri kümelerine ait kayıt sayıları ve hata oranları detaylı olarak Çizelge-3 içerinde açıklanmıştır.

Çizelge–2: Veri Kümesi Parametreleri

Parametre	Açıklama/Formül
loc	Toplam Satır Sayısı
v(g)	Çevrimsel Karmaşıklığı
ev(g)	Esas Karmaşıklığı
iv(g)	Tasarımsal Karmaşıklığı
N	Toplam Operatör ve İşlenen Sayısı
V	Hacim
L	Program Uzunluğu
D	Zorluk
I	Kabiliyet
E	Efor
B	Hata Sayısı
T	Zaman
lOCode	Kod Satırı Sayısı
lOComment	Yorum Satırı Sayısı
lOBlank	Boş Satır Sayısı
lOCodeAndComment	Yorumlu Kod Satır Sayısı
uniq_Op	Tekil Operatör
uniq_Opnd	Tekil İşlenen
total_Op	Toplam Operatör Sayısı
total_Opnd	Toplam İşlenen Sayısı
branchCount	Akış Grafiğindeki Dal Sayısı
Hata (Sınıf)	Evet / Hayır

Çizelge–3: Veri Kümesi Hata Oranları

Veri Kümesi	Toplam Kayıt Sayısı	asız Kayıt Sayısı	talı Kayıt Sayısı	Hata Oranı
CM1	498	449	49	%9,83
JM1	10885	8779	2106	%19,35
KC1	2109	1783	326	%15,45
KC2	522	415	105	%20,50
PC1	1109	119	11	%8,46

3.2 Veri Ön İşleme

Bu çalışmada önerilen yöntem yazılım içerisinde yer alan hataların tespitini hedeflemektedir. PROMISE açık kaynaklı veri deposundan alınan veriler, örnekleme yöntemleri ve makine öğrenmesi algoritmaları ile analiz edilmiş ve başarım değerleri elde edilmiştir. Yapılan çalışmada öncelikli olarak NASA PROMISE veri deposundan toplanan veriler analiz edilerek makine öğrenmesi yöntemlerine uygun olacak şekilde düzenlenmiştir. Daha sonra ikinci adım olarak veri kümelerine standart normalizasyon işlemi uygulanmıştır. Normalize edilmiş verilerde yer alan sınıf dengesizliği problemi incelenerek veri setlerine örnekleme yöntemleri uygulanmıştır. Burada veri kümesi eğitim ve test verileri olarak 2'ye ayrılarak sadece eğitim kümesinde örnekleme yöntemleri uygulanmıştır. Ön işlemden geçirilen ve eğitim/test olarak ayrılan veriler çalışmada kullanılan örnekleme yöntemleri ve sınıflandırma algoritmalarına hazır hale getirilmiştir.

3.3 Örnekleme Yöntemleri

Bir veri kümesinde sınıf etiketlerindeki gruplardan birinde bulunan gözlem sayısının diğer grupta bulunan gözlem sayısından fazla ya da az olması durumunda ortaya çıkan

problem sınıf dengesizliği problemidir. Veri kümesi içerisinde yer alan az sayıdaki veri grubu azınlık sınıfı diğer grup ise çoğunluk sınıfı olarak adlandırılır. Sınıf dengesizliği problemini çözebilmek için 3 farklı yaklaşım izlenebilir [20]. Bu yaklaşımlardan ilki verilerden yapay veriler üreterek sınıflar arası dengesizliğin çözülmesidir. Diğer bir yaklaşım ise çoğunluk sınıfında yer alan verilerden azınlık sınıfına yakın olacak şekilde örnekler seçerek daha az örnekleme yapılmasıdır. Sınıflandırma işlemi esnasında, sınıfların eşit dağılmadığı dengesiz sınıf problemlerinde örnekleme yöntemleri kullanılmaktadır. Bir diğer yaklaşım ise azınlık sınıfından yeni azınlık sınıfları oluşturarak çoğunluk sınıfına denk gelecek şekilde verilerin artırılmasıdır. Bu çalışmada sınıf dengesizliği problemini gidermek için az örnekleme ve aşırı örnekleme yöntemleri kullanılmıştır.

3.3.1 Üst Örnekleme Yöntemleri

Sentetik Azınlık Örneklem Arttırma Yöntemi (SMOTE – Synthetic Minority Oversampling Technique), en başarılı örnekleme yöntemlerinden birisi olarak bilinmektedir ve 2022 yılında geliştirilmiştir [21]. Bu yöntem, veri kümesindeki azınlık elemanlarından yeni örnekler oluşturarak eleman sayısını dengeli bir hale getirmeyi temel alır. Bu yöntem ile çoğunluk sınıfının eleman sayısında bir değişim yaşanmaz [22]. SMOTE yönteminde, her azınlık sınıfının bir örneği alınır ve bu örneğe ait k komşusunun herhangi birine ya da tümüne bakılarak sentetik örnekler oluşturulmaktadır [23]. Sentetik örnekler üretmek için interpolasyon tekniği kullanılır. Bu sentetik örnekler, sınıflandırıcının daha küçük ve daha spesifik bölümler yerine daha büyük ve daha az spesifik karar bölgeleri oluşturmasını sağlamaktadır [24].

Rastgele üst örnekleme yöntemi, sınıf dengesizliği probleminde kullanılan en eski ve en basit yöntemdir [25]. Sınıf dengesizliği içeren veri kümesinde azınlıktaki sınıftan rastgele farklı örneklerin seçilip tekrar azınlık sınıfına eklenmesi ve bu mantıkla veri kümesinin dengelenmesi akışını izleyen aşırı örnekleme yöntemlerinden biridir. Küçük bir veri kümesinde uygulanması aşırı öğrenme gibi durumlara yol açabilen rastgele örnekleme yöntemi, yaygın olarak kullanılan örnekleme yöntemlerinden birisidir.

Borderline SMOTE yöntemi, SMOTE yönteminden esinlenerek oluşturulmuş bir diğer üst örnekleme yöntemlerinden birisidir. Borderline SMOTE yönteminde, azınlıkta olan veri sınıfının sınır değerlerinde yer alan örnekler SMOTE yöntemi ile arttırılmaktadır. SMOTE yöntemi ile bu yöntem arasındaki temel fark ise, SMOTE yönteminde azınlık sınıfında yer alan tüm örnekler arttırılırken Borderline SMOTE yönteminde sınırlarda yer alan değerlerin arttırılmasıdır [26].

SVM-SMOTE yöntemi, Borderline SMOTE yönteminde olduğu gibi sınır çizgileri yakınında yer alan azınlık sınıfının örneklerini arttırmaya odaklanan bir yöntemdir. Ancak bu yöntemde SVM yöntemi ile sınır çizgilerinin yakınında yeni azınlık sınıfı örnekleri arttırılır.

SMOTE yönteminin bir varyasyonu olarak bilinen ADASYN yönteminde veri yoğunluğunun daha az olduğu alanlarda daha çok örnekler oluşturulur. Bu yöntemde sadece sınıf dengesizliği problemi çözülmeyip aynı zamanda öğrenilmesi zor olan birimlere odaklanmak için karar sınırını uyarlamalı olarak değiştirir. Bu yöntem ile veri dağılımlarındaki öğrenme iki şekilde geliştirilmektedir. Bunlardan ilki sınıf dengesizliğinin getirdiği sapmanın azaltılması diğeri ise sınıflandırma karar sınırının zor örneklere uyarlanabilir şekilde kaydırılmasıdır [27].

Tomek bağlantıları, Tomek tarafından geliştirilmiş yoğun en yakın komşu örnekleme tekniğinin değiştirilmiş bir biçimidir. SMOTE + Tomek yöntemi ise SMOTE ve Tomek bağlantılarının kombinasyonu olarak bilinmektedir. SMOTE yönteminin azınlık sınıfını için sentetik üretmesi ve Tomek bağlantıları olarak tanımlanan verilerin çoğunluk sınıfından kaldırılması işlemlerini birleştiren bir yöntemdir. SMOTE+Tomek, bir diğer tanımla, örnek uzayda dağıtılmış sınıfların her biri için örtüşen veri noktalarını temizlemeyi amaçlayan melez bir örnekleme tekniğidir. SMOTE yöntemi ile yüksek hızda örnekleme yapılmış sınıf kümeleri, birbirlerinin alanını işgal etmiş olabilirler. Bu durum ile SMOTE tarafından yapılan aşırı örneklenmiş azınlık sınıfı örneklerine Tomek bağlantıları uygulanır. Bu işleyişle ile yöntem veri kümelerine uygulanır.

SMOTE yönteminin türlerinden biri olan SMOTE-NC hem nominal hem de sürekli özelliklere sahip veri kümeleri için tasarlanmış, verilerdeki sınıf dengesizliği problemine çözüm olan aşırı örnekleme yöntemlerinden biridir. Veri kümesinde önemli sayıda nominal özellik olduğunda ve kategorik özellikle ile hedef sınıf arasında bir ilişki olduğunda bu yöntemin daha iyi tahmin sağladığı görülmektedir [28].

3.3.2 Alt Örnekleme Yöntemleri

Alt örnekleme, sınıf dengesizliği olan bir veri kümesinde azınlık sınıfındaki verileri tutarak, çoğunluk sınıfının boyutu küçültüp eşit olmayan çoğunluk ve azınlık sınıflarını dengelemeye odaklanan bir tekniktir. Potansiyel olarak verilerde yer alan önemli bilgilerin kaybı gibi dezavantajları olmasına rağmen birçok veri bilimcisi tarafından kullanılan tekniklerden birisidir. Rastgele alt örnekleme yöntemi, sınıf dengesizliği olan bir veri kümesinde çoğunluk sınıfında yer alan verilerden rastgele veriler kaldırarak sınıf dengesizliğini çözen örnekleme yaklaşımlarından birisidir. Bu yöntemde veri kümesinin sayısı azaltılacağından, sınıflandırma süresi de azalmış olur. Bu yöntemin en büyük dezavantajı ise sınıflandırmada önemli değere sahip olabilecek verilerin rastgele kaldırılma olasılığının olmasıdır.

Küme Merkezli Örnekleme, kümeleme yöntemleri ile merkezlere dayalı yeni bir küme oluşturarak alt örnekleme yapan bir tekniktir. Bu teknik en yakın komşu algoritmasının küme merkezine göre yeni bir küme oluşturmaktadır. Çoğunluk sınıfındaki örneklerden oluşan bir kümeyi en yakın komşu algoritmasının küme merkeziyle değiştirerek çoğunluk sınıfının altında örnekler oluşturan bir yöntemdir. Azınlık sınıfındaki değerler bu yöntemde korunurken çoğunluk sınıfındaki veriler en yakın komşu algoritması ile yeniden sentezlenir.

3.3.3 Alt ve Üst Örnekleme Yöntemleri

SMOTE+ENN yöntemi, 2004 yılında önerilen, SMOTE ve en yakın komşu kuralını birleştiren bir örnekleme yöntemidir. SMOTE aşırı bir örnekleme yöntemidir ve temelinde azınlık

sınıfından yeni örnekler oluşturarak sınıf dengesizliği problemini çözmeye çalışır. Burada sınıf dengesizliği problemi çözülse de yeni örneklerin yanında gürültü ve sınır örnekleri de oluşabilir. Daha iyi örnekler oluşturabilmek için ENN, sınıf etiketi en yakın üç komşusundan en az ikisinin sınıfından farklı olan herhangi bir örneği kaldırabilen bir veri temizleme yöntemi kullanır. Bazı çoğunluk sınıfındaki örnekler, azınlık sınıfının alanını işgal edebileceği için SMOTE+ENN sentetik örneklerin ortaya çıkardığı fazla uydurma olasılığını azaltır.

3.4 Sınıflandırma Algoritmaları

Yazılım hata tahmininde yapılan çalışmalarda farklı teknikler kullanılmıştır. Bu teknikler dikey hata sınıflandırması, makine öğrenmesi teknikleri, derin öğrenme teknikleri, yapay sinir ağları teknikleri olarak sıralanabilir. Çalışmalarda farklı birçok makine öğrenmesi algoritması kullanılmış ve başarım sonuçları elde edilmiştir.

Yazılım hata tahmininde makine öğrenmesi algoritmaları ile sınıflandırma birçok çalışmanın konusu olmuştur. Makine öğrenmesi yöntemlerinin kullanılmasının nedeni, kolay anlaşılır ve iyi başarımı yüksek hızda elde edebilecek birçok algoritmayı içermesindendir. Bu çalışmada da kullanılan veri kümesindeki sınıf dengesizliği problemi örnekleme yöntemleri çözülmüş, örnekleme işlemi sonrasında ise veri kümesine 13 adet aşağıda yer alan sınıflandırma algoritması uygulanmıştır.

K-En Yakın Komşu Algoritması, sık kullanılan makine öğrenmesi algoritmalarından en yakın komşu algoritması, birbirine yakın durumda olan değerlerin benzer olacağı mantığına dayanır [29]. Öğrenme tabanlı algoritmalardan biri olan bu algoritmada, yeni karşılaşılan X bir değer, eğitim kümesindeki örnekler ile arasındaki benzerliğe göre sınıflandırılır [30]. Sınıflandırma işlemi esnasında bir değerin diğer bir değere olan uzaklığı hesaplanır. Bir değerin diğer bir değere uzaklığının k adedi kadar komşusu dikkate alınır. Bu en yakın komşular içerisinden en çok benzerlik gösterdiği sınıfa ataması yapılır ve algoritma bu şekilde sınıflandırma işlemini yapar [31].

Destek vektör makineleri, iki farklı sınıfı bir doğru, bir düzlem üzerinde birbirinden ayırmaya odaklanan sınıflandırma problemlerinde oldukça yaygın kullanılan algoritmalardan birisidir. İki sınıfı birbirinden ayırırken sınırda yer alan değerler ile bu işlemi gerçekleştirir. Ayrılma marjının net olduğu zamanlarda çok iyi başarım gösteren algoritmalardan birisidir. Bu yöntem, iki farklı gruba göre etiketlenmiş bir eğitim verisi için geliştirilen algoritması ile yeni bir değer verildiğinde bu değerin hangi gruba ait olması gerektiğini belirleyen bir model oluşturmaktadır ve bu grup ayrımını yaparken hiper düzlem üzerinde çalışmaktadır. [32].

Karar ağacı algoritması, sınıflandırma işlemini bir ağaç yapısı oluşturarak yapan bir algoritmadır. Oluşturulan ağaç yapısında yapraklar sınıf etiketlerini temsil ederken, kökten yapraklara giden yollar ise özniteliklere göre dallanmaları temsil etmektedir [33]. Karar ağacı algoritması böl ve yönet yaklaşımdan yaralanarak sınıflandırma gerçekleştiren bir yöntemdir. Karar ve bitiş düğümlerinden oluşan bir ağaçta, her bir karar düğümü dalları etiketleyen ayrık bir fonksiyonu gerçekleştirmektedir. Verilen değerin bu fonksiyonların ile sınıflandırma işlemi yapılmaktadır [34].

Rastgele orman algoritması, Karar ağaçlarının birleşmesine dayanan, öğrenme sürecinde her bir ağacın sınıflandırma işlemini rastgele seçimler yaparak yapan bir algoritmadır [35]. Rastgele orman algoritması, daha doğru ve daha kararlı tahminler elde edebilmek için daha fazla karar ağacını birleştirir. Denetimli öğrenme yöntemlerinden biri olan bu algoritmada kullanılan ağaç sayısı ile doğruluk arasında doğrusal bir ilişki vardır yani ağaç sayısı arttıkça algoritmanın doğruluk oranı artmaktadır [36].

Topluluk öğrenme algoritmalarından birisi olan Adaboost, “Boosting” yani verilerden çıkan zayıf sonuçları birleştirerek güçlü bir işlem elde etmeyi sağlayan bir algoritmadır [37]. Sınıflandırma problemi için birden fazla öğrenici ile eğitim gerçekleştirerek işlem yapmaktadır. Başlangıçta tüm verilere eşit ağırlık verip daha sonra oluşturulan sınıflandırıcılar arasından en zayıf sınıflandırıcıyı seçer [38].

Gradyan Artırma algoritması temel, güçlü bir tahminci elde edebilmek için zayıf tahminleri kademeleri olarak güçlü tahmincilere dönüştürmeye odaklanır. Karar ağacı, kök noktasından başlayarak aşağıya doğru dallanmalar yapar ve yapraklar oluşturur. Algoritmada ilk olarak oluşturulan karar ağacı ile bir tahmin elde edilir. Elde edilen tahmin ile hedef arasındaki fark hesaplanır. Her yeni yinelemede, hesaplanan fark ile yeni bir ağaç oluşturulur.

Ekstra gradyan artırma algoritması (XGBoost), gradyan arttırma ve karar ağacı algoritmasına dayanan makine öğrenmesi algoritmalarından birisidir. Yüksek tahmin gücü olan bir algoritma olarak bilinen bu algoritma diğer algoritmalardan genel başarımı iyileştiren, fazla öğrenmeyi azaltan bir düzenleme içermektedir [39]. Bu yöntem karar ağacı tabanlı bir yöntem olduğu için, karar ağacı oluştururken maksimum derinlik değerini kullanmaktadır. Oluşturulan ağaç sayısı aşağıda yönde aşırı bir ilerleme gösterir ise budama gerçekleştirilerek aşırı öğrenmenin önüne geçilmektedir.

Hafif gradyan artırma makineleri algoritması (LightGBM), XGBoost algoritmasının eğitim süresinin başarımını arttırmak için geliştirilen başarımı yüksek bir algoritma türüdür. Yüksek doğruluk, düşük kaynak kullanımı hızlı ve verimli oluşu nedeniyle sık tercih edilen algoritmalardan birisidir. Bu yöntemde yaprak odaklı ağaç oluşturma sistemi olduğu için hata oranı azalır ve daha hızlı öğrenme işlemi gerçekleşir [40]. Bu yöntemi diğer gradyan artırma algoritmalarından ayıran yanı ise, karar ağaçlarının eğitilmesi aşamasında kullandığı büyüme stratejisidir [41].

Catboost algoritması, gradyan artırma karar ağacına dayalı sınıflandırma algoritmalarından birisidir. Yüksek hızda öğrenme yeteneği olan bu algoritma kategorik ve sayısal verilerde daha hızlı çalışabilmektedir [42]. Veri hazırlığı aşamasını kısaltması, boş veriler ile başa çıkabilmesi önemli özelliklerindendir. Simetrik ağaç yapısı kurarak çok derin ağaç yapısı olmadan öğrenme işlemini gerçekleştirir.

Naive Bayes sınıflandırma algoritması şartlı olasılık kurallarına göre düzenlenmiş, hesaplama ve kullanım kolaylığı ile makine öğrenmesi algoritmaları arasında en sık kullanılan bir

algoritmadır [43]. Bu algoritma Bayes teoremine göre sınıflandırma işlemini hesaplamaktadır. Bir sınıfa ait öznitelik değerinin etkisinin, diğer öznitelik değerlerinden bağımsız olduğu varsayılır ve bu durum sınıf koşul bağımsızlığı olarak adlandırılır [44]. Naive Bayes yöntemi ulaşılması istenen hedef etiket ile problemde uygulanan giriş parametreleri arasındaki ilişkiyi tahmin etmeye uyum sağlayan sınıflandırma yöntemidir [45].

Lineer diskriminant analizi, denetimli sınıflandırma yöntemlerinde sık kullanılan bir boyut indirme tekniği olan bir yöntem olup iki sınıfı birbirinden ayırırken farklılıkları modeller. Daha yüksek boyutlu uzaydaki özellikleri daha düşük boyutlu bir uzaya yansıtmak için kullanılır. Bu algoritma sınıflar arasındaki varyansı en aza indirerek, sınıflar arasındaki mesafeyi en üst düzeye çıkarıp iz düşüm hiper düzlemini bulmayı hedeflemektedir [46]. Popüler sınıflandırma algoritmalarından biri olan Kuadratik Discriminant Analizi, farklı sınıfların bilinen bir olasılık fonksiyonuna ait olduğunu varsayan bir sınıflandırıcıdır [47]. Bu algoritmada sınıfların ortalama ve kovaryans matrisleri bulunur ve yeni örnek en yüksek olasılığa sahip olan sınıfa etiketlenir.

Rastgele orman algoritması ile aynı yapıda olan Ekstra Ağaçlar algoritması, veri setlerindeki kopyaları kullanarak modelleri eğitmektedir. Rastgele orman algoritmasının farklı bir mimarisini oluşturmaktadır. Rastgele ağaç algoritmasından farkı oluşturulan düğümlerin dallara ayrılması aşamasında karar alma ölçütü yerine rastgele dallanma yolunun seçilmesidir.

3.5 Başarım Ölçünleri

Bu çalışmada makine öğrenmesi algoritmaları ile yapılan sınıflandırmaların başarımlarını ölçmek için doğruluk (doğruluk) ve AUC/AUROC ölçütleri kullanılmıştır. Sınıflandırma algoritmalarının başarım testinde karmaşıklık matrisi kullanılır. Bir karmaşıklık matrisi doğru pozitif (DP), yanlış pozitif (YP), doğru negatif (DN) ve yanlış negatif (YN) adı verilen dört değerden oluşmaktadır. DP ve DN değerleri karmaşıklık matrisinde gerçekten tahmin edilmiş hatalı ve hatasız kayıtları temsil eder. YP ve YN değerleri ise yanlış tahmin edilen hatalı ve hatasız kayıtların temsili gerçekleştirir. Makine öğrenmesi başarım metriklerinden biri olan karmaşıklık matrisi sınıf etiketlerinin görselleştirilmesi için bir araç görevindedir.

Bu çalışmada kullanılan veri kümesinde sınıf dengesizliği problemi bulunmaktadır. Bu tür veri setlerinde sınıflandırma yapılırken en iyi başarımı verecek ölçünlerin seçimi önem teşkil etmektedir. Bu kapsamda sınıf dengesizliği probleminde başarım karşılaştırması için genellikle doğruluk değeri kullanılmaktadır. Bir diğer yandan doğruluk başarım değerlendirilmesinde tek ölçüt olmamalıdır [48]. Sınıf dengesizliği probleminde AUC/AUROC metriği ise sınıfları ayırt edebilmede başarısı göstermekte ve doğruluk metriği yanında kullanan metriklerden birisidir.

4. Deney Sonuçları

Bu çalışmada, yazılım hata tahmininde sınıf dengesizliği problemini çözebilmek ve yazılım hata tahminini sağlamak için mevcut algoritmalar ile bir model önerilmiştir. Üzerinde çalışan bu model Kaggle platformunda yer alan veri setlerinden yararlanılarak, Python programlama dili ile analiz edilmiştir. Tüm analizler Intel Core i7 işlemcili, 16 GB RAM ve 500 GB disk kapasitesine sahip bilgisayarda gerçekleştirilmiştir. İlk aşamada, çalışmada kullanılan NASA veri deposundan alınan CM1, KC1, KC2, JM1, PC1 veri kümeleri için bir ön işlem uygulanmıştır. Veri ön işlem aşamasında, standart normalizasyon yöntemi ile veriler normalize edilmiştir. Veri kümeleri önişlemden geçirildikten sonra içerisinde yer alan hatalı ve hatasız kayıtlar arasında bir sınıf dengesizliği problemi olduğu görülmüştür. Literatürdeki çalışmalar incelenerek yazılım hata tahmininde bu veri kümesindeki sınıf dengesizliği probleminin çözümünde sadece eğitim kümesine örnekleme uygulanarak yöntemin daha doğru sonuçlara ulaştığı sonucuna varılmıştır. Bu doğrultuda veri kümesi eğitim ve test veri kümesi olarak ayrılıp sadece eğitim veri kümesine sınıf dengesizliği probleminde kullanılan örnekleme yöntemleri uygulanmıştır. Daha sonra eğitim veri kümesine çeşitli makine öğrenmesi algoritmaları ile sınıflandırma işlemi uygulanmıştır. En son aşama olarak test aşamasında sınama veri kümesine uygulanan algoritmalar ile başarım karşılaştırması yapılmıştır. Kullanılan sınıflandırma algoritmaları ve örnekleme yöntemleri ile elde edilen başarım analizi değerleri aşağıdaki şekiller içerisinde detaylı olarak gösterilmiştir.

PC1 veri kümesinde SMOTE örnekleme yöntemi ile en iyi AUROC değeri Extra Gradyan Arttırma sınıflandırma algoritmasında %89,94, en iyi doğruluk değeri ise Extra Tree algoritmasında 94.59% elde edilmiştir. PC1 veri kümesinde Rastgele Örnekleme yöntemi ile en iyi AUROC değeri Catboost algoritmasında %89,68, en iyi doğruluk değeri ise yine Catboost algoritmasında %94,14 olarak elde edilmiştir. PC1 veri kümesinde SMOTENN yöntemi ile en iyi AUROC değeri Catboost algoritmasında %91,83, en iyi doğruluk değeri Ekstra Ağaçlar algoritmasında %94,59 olarak elde edilmiştir. PC1 veri kümesinde Rastgele Örnekleme yöntemi ile en iyi AUROC değeri Ekstra Ağaçlar algoritmasında %90,2, en iyi doğruluk değeri Gradyan Arttırma algoritmasında %73,87 olarak elde edilmiştir. PC1 veri kümesi Küme Merkezli Örnekleme yöntemi ile en iyi AUROC değeri Ekstra Ağaçlar algoritmasında %87,73, en iyi doğruluk değeri KNN algoritması ile %80,63 olarak elde edilmiştir. PC1 veri kümesinde ADASYN yöntemi ile en iyi AUROC değeri Catboost algoritmasında %89,42, en iyi doğruluk değeri Ekstra Ağaçlar algoritmasında %94,14 olarak elde edilmiştir. PC1 veri kümesinde Borderline SMOTE yöntemi ile en iyi AUROC değeri Catboost algoritmasında %89,94, en iyi doğruluk değeri Ekstra Ağaçlar algoritmasında %95,04 olarak elde edilmiştir. PC1 veri kümesinde SVM-SMOTE yöntemi ile en iyi AUROC değeri Catboost algoritmasında %91,99, en iyi doğruluk değeri Ekstra Ağaçlar algoritmasında %94,59 olarak elde edilmiştir. PC1 veri kümesinde SMOTE-Tomek yöntemi ile en iyi AUROC değeri XGB algoritmasında %90,06, en iyi doğruluk değeri Ekstra Ağaçlar algoritmasında %94,59 olarak elde edilmiştir. PC1 veri kümesinde SMOTENC yöntemi ile en iyi AUROC değeri XGB algoritmasında %88,55, en iyi doğruluk değeri Ekstra Ağaçlar algoritmasında %93,69 olarak elde edilmiştir. Elde edilen sonuçlar Şekil-3'te özetlenmiştir.

JM1 veri kümesinde SMOTE örnekleme yöntemi ile en iyi AUROC değeri Extra Tree sınıflandırma algoritmasında %74,98, en iyi doğruluk değeri ise Extra Tree algoritmasında %80,30 elde edilmiştir. JM1 veri kümesinde Random Oversampling yöntemi ile en iyi AUROC değeri LGBM algoritmasında %73,84, en iyi doğruluk değeri Extra Tree algoritmasında %80,84 olarak elde edilmiştir. JM1 veri kümesinde SMOTENN yöntemi ile en iyi AUROC değeri Catboost algoritmasında %74,68, en iyi doğruluk değeri Extra Tree algoritmasında %81,35 olarak elde edilmiştir. JM1 veri kümesinde Random Undersampling yöntemi ile en iyi AUROC değeri Extra Tree algoritmasında %72,69, en iyi doğruluk değeri Gaussian Naive Bayes algoritmasında %68,53 olarak elde edilmiştir. JM1 veri kümesi Cluster Centroid yöntemi ile en iyi AUROC değeri Quadratic Discriminant Analysis algoritmasında %60,37, en iyi doğruluk değeri Gaussian Naive Bayes algoritması ile %61,64 olarak elde edilmiştir. JM1 veri kümesinde ADASYN yöntemi ile en iyi AUROC değeri Extra Tree algoritmasında %74,69, en iyi doğruluk değeri Catboost algoritmasında %79,74 olarak elde edilmiştir. JM1 veri kümesinde Borderline SMOTE yöntemi ile en iyi AUROC değeri Extra Tree algoritmasında %74,66, en iyi doğruluk değeri XGB algoritmasında %79,65 olarak elde edilmiştir. JM1 veri kümesinde SVM-SMOTE yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %75,21, en iyi doğruluk değeri Catboost algoritmasında %80,25 olarak elde edilmiştir. JM1 veri kümesinde SMOTE-Tomek yöntemi ile en iyi AUROC değeri Extra Tree algoritmasında %74,93, en iyi doğruluk değeri LGBM algoritmasında %80,43 olarak elde edilmiştir. JM1 veri kümesinde SMOTENC yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %75,24, en iyi doğruluk değeri Catboost algoritmasında %80,75 olarak elde edilmiştir. Elde edilen sonuçlar Şekil-4'te özetlenmiştir.

KC1 veri kümesinde SMOTE örnekleme yöntemi ile en iyi AUROC değeri Random Forest sınıflandırma algoritmasında %83,48, en iyi doğruluk değeri ise Catboost algoritmasında %85,31 elde edilmiştir. KC1 veri kümesinde Random Oversampling yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %82,67, en iyi doğruluk değeri yine Random Forest algoritmasında %85,07 olarak elde edilmiştir. KC1 veri kümesinde SMOTENN yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %84,12, en iyi doğruluk değeri Adaboost algoritmasında %86,49 olarak elde edilmiştir. KC1 veri kümesinde Random Undersampling yöntemi ile en iyi AUROC değeri Catboost algoritmasında %82,34, en iyi doğruluk değeri KNN algoritmasında %72,98 olarak elde edilmiştir. KC1 veri kümesi Cluster Centroid yöntemi ile en iyi AUROC değeri Quadratic Discriminant Analysis algoritmasında %74,39, en iyi doğruluk değeri Linear Discriminant Analysis algoritması ile %73,69 olarak elde edilmiştir. KC1 veri kümesinde ADASYN yöntemi ile en iyi AUROC değeri Catboost algoritmasında %81,78, en iyi doğruluk değeri Catboost algoritmasında %79,74 olarak elde edilmiştir. KC1 veri kümesinde Borderline SMOTE yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %82,86, en iyi doğruluk değeri Catboost algoritmasında %86,73 olarak elde edilmiştir. KC1 veri kümesinde SVM-SMOTE yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %83,39, en iyi doğruluk değeri LGBM algoritmasında %85,78 olarak elde edilmiştir. KC1 veri kümesinde SMOTE-Tomek yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %83,42, en iyi doğruluk değeri LGBM algoritmasında %85,07 olarak elde edilmiştir. KC1 veri kümesinde SMOTENC yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %83,52, en iyi doğruluk değeri LGBM algoritmasında %85,54 olarak elde edilmiştir. Elde edilen sonuçlar Şekil-5'te özetlenmiştir.

KC2 veri kümesinde SMOTE örnekleme yöntemi ile en iyi AUROC değeri Quadratic Discriminant Analysis sınıflandırma algoritmasında %81,54, en iyi doğruluk değeri ise Adaboost algoritmasında %80,95 elde edilmiştir. KC2 veri kümesinde Random Oversampling yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %80,55, en iyi doğruluk değeri yine Random Forest algoritmasında %80,95 olarak elde edilmiştir. KC2 veri kümesinde SMOTENN yöntemi ile en iyi AUROC değeri Gradient Boosting algoritmasında %79,68, en iyi doğruluk değeri Catboost algoritmasında %80,95 olarak elde edilmiştir. KC2 veri kümesinde Random Undersampling yöntemi ile en iyi AUROC değeri XGB algoritmasında %82,14, en iyi doğruluk değeri Random Forest algoritmasında 77.14% olarak elde edilmiştir. KC2 veri kümesi Cluster Centroid yöntemi ile en iyi AUROC değeri Quadratic Discriminant Analysis algoritmasında %80,61, en iyi doğruluk değeri yine Quadratic Discriminant Analysis algoritması ile %80,95 olarak elde edilmiştir. KC2 veri kümesinde ADASYN yöntemi ile en iyi AUROC değeri Quadratic Discriminant Analysis algoritmasında %81,43, en iyi doğruluk değeri Gradient Boosting algoritmasında %82,85 olarak elde edilmiştir. KC2 veri kümesinde Borderline SMOTE yöntemi ile en iyi AUROC değeri Catboost algoritmasında %72,89, en iyi doğruluk değeri LGBM algoritmasında %89,00 olarak elde edilmiştir. KC2 veri kümesinde SVM-SMOTE yöntemi ile en iyi AUROC değeri Gradient Boosting algoritmasında %80,83, en iyi doğruluk değeri XGB algoritmasında %81,90 olarak elde edilmiştir. KC2 veri kümesinde SMOTE-Tomek yöntemi ile en iyi AUROC değeri XGB algoritmasında %83,04, en iyi doğruluk değeri yine XGB algoritmasında %82,86 olarak elde edilmiştir. KC2 veri kümesinde SMOTENC yöntemi ile en iyi AUROC değeri Quadratic Discriminant Analysis algoritmasında %81,70, en iyi doğruluk değeri XGB algoritmasında %81,90 olarak elde edilmiştir. Elde edilen sonuçlar Şekil-6'da özetlenmiştir.

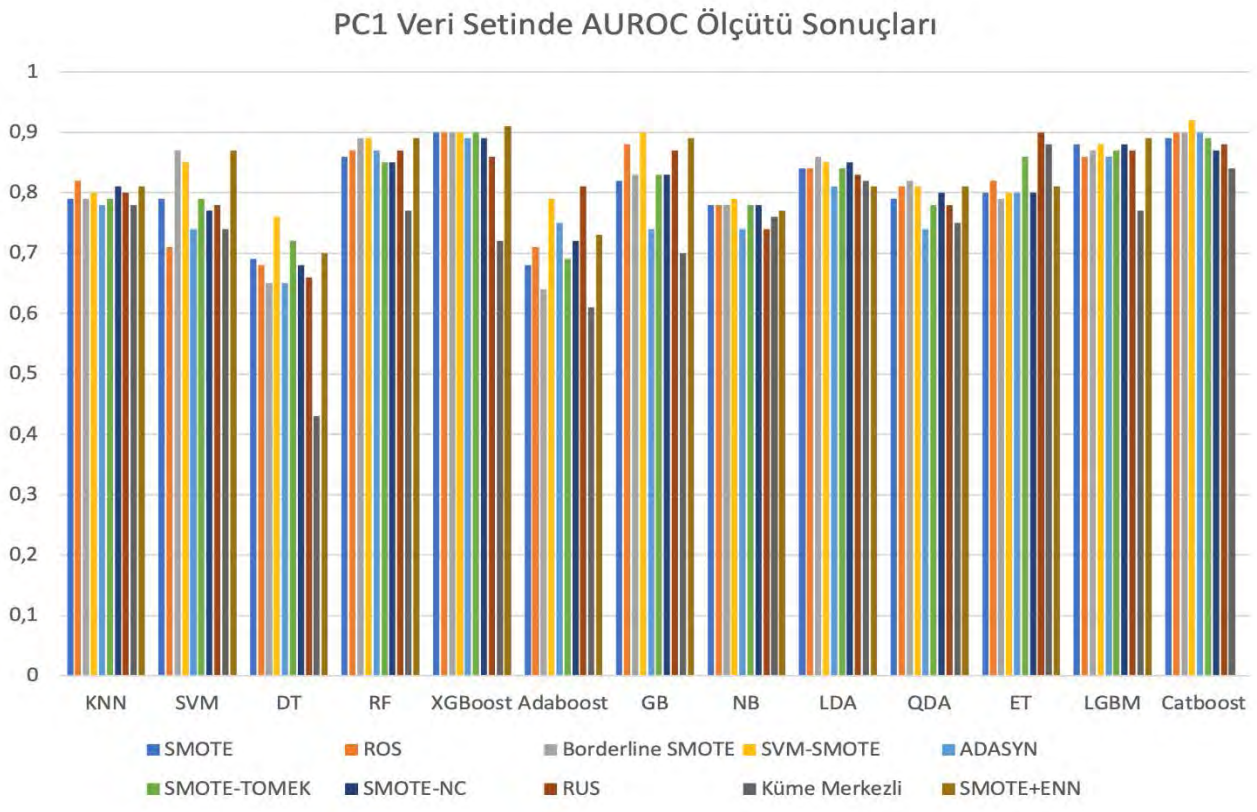
CM1 veri kümesinde SMOTE örnekleme yöntemi ile en iyi AUROC değeri Extra Tree sınıflandırma algoritmasında %74,56, en iyi doğruluk değeri ise Random Forest algoritmasında %89,00 elde edilmiştir. CM1 veri kümesinde Random Oversampling yöntemi ile en iyi AUROC değeri Extra Tree algoritmasında %72,67, en iyi doğruluk değeri yine Extra Tree algoritmasında algoritmasında %90,00 olarak elde edilmiştir. CM1 veri kümesinde SMOTENN yöntemi ile en iyi AUROC değeri Gradient Boosting algoritmasında %72,67, en iyi doğruluk değeri LGBM algoritmasında %91,00 olarak elde edilmiştir. CM1 veri kümesinde Random Undersampling yöntemi ile en iyi AUROC değeri Catboost algoritmasında %74,44, en iyi doğruluk değeri Gradient Boosting algoritmasında %71,00 olarak elde edilmiştir. CM1 veri kümesi Cluster Centroid yöntemi ile en iyi AUROC değeri

Random Forest algoritmasında %72,83, en iyi doğruluk değeri KNN algoritması ile %73,00 olarak elde edilmiştir. CM1 veri kümesinde ADASYN yöntemi ile en iyi AUROC değeri Rastgele Orman algoritmasında %77,11, en iyi doğruluk değeri LGBM algoritmasında %88,00 olarak elde edilmiştir. CM1 veri kümesinde Borderline SMOTE yöntemi ile en iyi AUROC değeri Extra Tree algoritmasında %74,39, en iyi doğruluk değeri Random Forest algoritmasında 88.00% olarak elde edilmiştir.

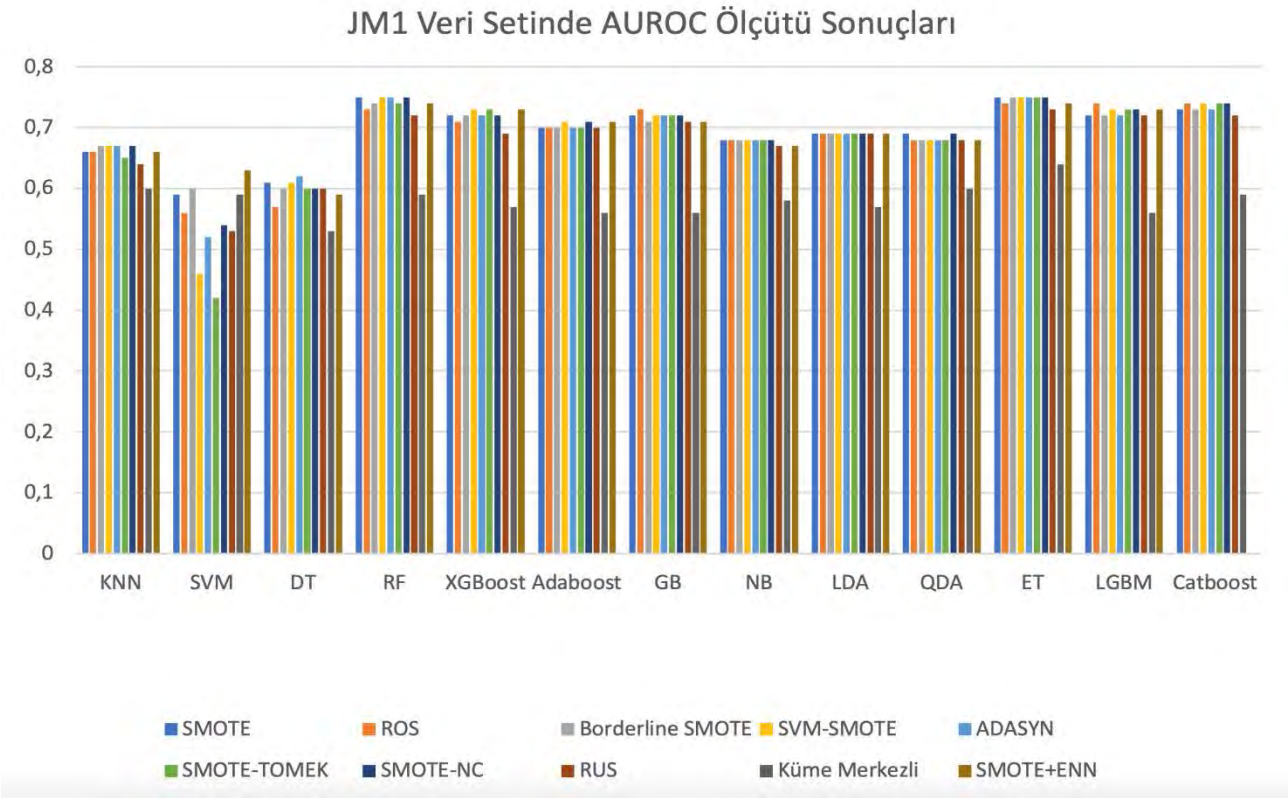
CM1 veri kümesinde SVM-SMOTE yöntemi ile en iyi AUROC değeri LGBM algoritmasında %75,44, en iyi doğruluk değeri Catboost algoritmasında %88,00 olarak elde edilmiştir. CM1 veri kümesinde SMOTE-Tomek yöntemi ile en iyi AUROC değeri Random Forest algoritmasında %83,41, en iyi doğruluk değeri XGB algoritmasında %85,07 olarak elde edilmiştir. CM1 veri kümesinde SMOTENC yöntemi ile en iyi AUROC değeri XGB algoritmasında %74,11, en iyi doğruluk değeri LGBM algoritmasında %86,00 olarak elde edilmiştir. Elde edilen sonuçlar Şekil-7'de özetlenmiştir.

Çalışma kapsamında incelenen literatür çalışmalarında eğitim ve test veri setlerini ayrı değerlendiren çalışmalarda elde edilen sonuçlar şu şekilde açıklanmıştır. Sun ve ark. tarafından yapılan çalışmada, Random Forest ve Naive Bayes algoritmaları ile SMOTE örnekleme yöntemi kullanılmıştır. Eğitim kümesinde SMOTE yöntemi ile örnekleme işlemi yapılmıştır. Elde edilen sonuçlar, bizim çalışmamız ile kıyaslandığında en iyi 0,85 AUROC değeri PC1 veri kümesinde SMOTE yöntemi ve Random Forest algoritmasının kombinasyonu ile elde edilmiştir. Bizim çalışmamızda ise; PC1 veri kümesinde SMOTE yöntemi ile XGB algoritması kullanılarak yaklaşık 0,90 değerinde AUROC değeri elde edilmiştir. Bu başarım değeri ile literatür çalışmasında yer alan başarım değerinden daha iyi bir sonuç elde edilmiştir. Eivazpour ve ark. tarafından yapılan çalışmada, Decision Tree ve Random Forest algoritmaları kullanılmış, sadece eğitim veri kümesinde SMOTE örnekleme yöntemi kullanılmıştır. Bu çalışmada KC2 veri kümesinde SMOTE yöntemi ve Karar Ağacı algoritması ile 0,74 AUROC değeri en iyi başarım değeri olmuştur. KC2 veri kümesinde, bizim çalışmamızda Quadratic Discriminant Analysis algoritması ve SMOTE yönteminin birleşmesi ile 0,82 AUROC ile daha iyi bir başarım değeri elde edilmiştir. Malhotra ve ark. tarafından yapılan çalışmada ADASYN örnekleme yöntemi ile SVM, Naive Bayes algoritmaları birleştirilip sınıflandırma işlemi yapılmıştır. Bu çalışmada KC1 veri kümesinde SVM algoritması ve ADASYN örnekleme yöntemi ile 0.70 AUROC değeri en iyi başarım değeri olmuştur. Bizim çalışmamızda ise KC1 veri kümesinde ADASYN yöntemi ve Catboost algoritması birleştirilerek en iyi 0,82 AUROC başarım değeri elde edilmiştir. Bu çalışmada SMOTE örnekleme yöntemine de yer verilmiştir. SMOTE örnekleme yönteminde KC1 veri kümesinde SVM algoritması ile 0,83 AUROC değeri elde edilen en iyi başarım değeri olmuştur. Bizim çalışmamızda ise, KC1 veri kümesinde Random Forest algoritması ve SMOTE örnekleme yöntemi ile aynı oranda yani 0,83 AUROC değeri elde edilmiştir.

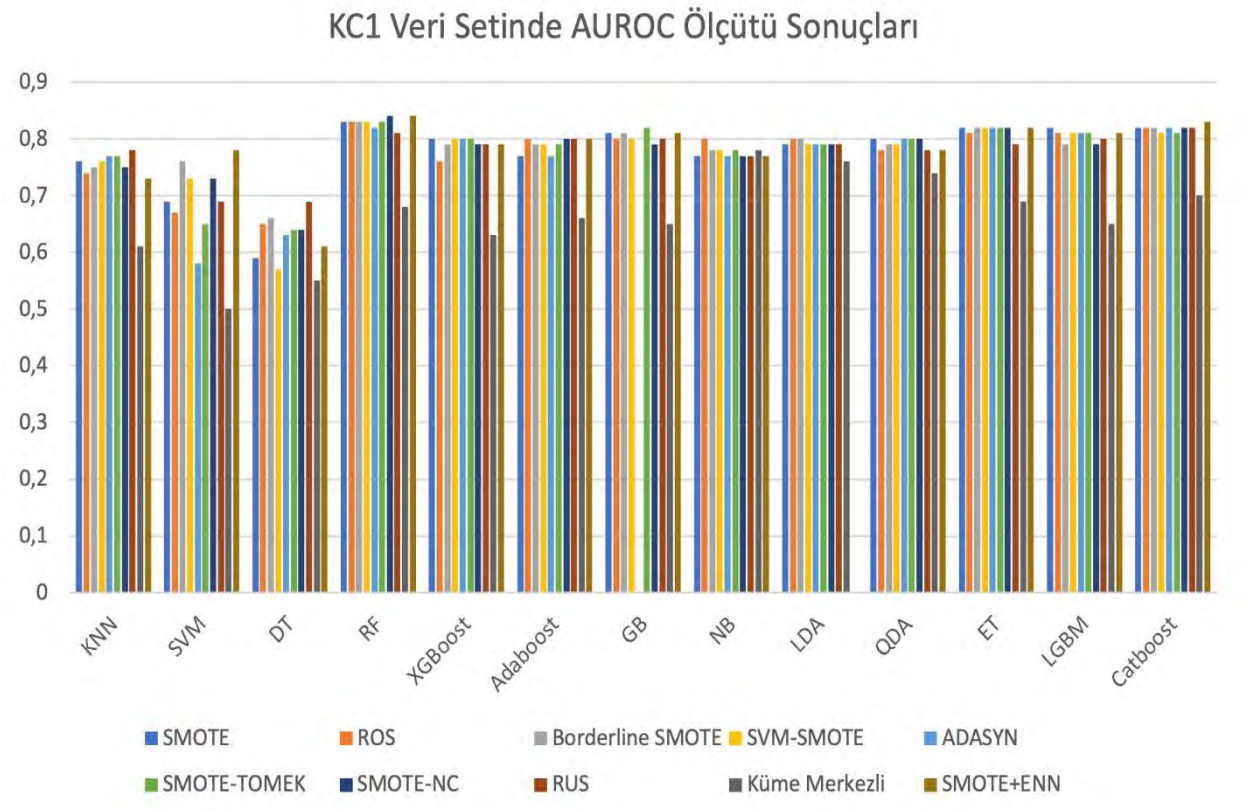
Tüm deneylerde veri kümesine bağlı olarak hangi örnekleme yönteminde hangi algoritma ile en iyi sonuçların elde edildiği Çizelge-5 içerisinde ayrıntılanmıştır.



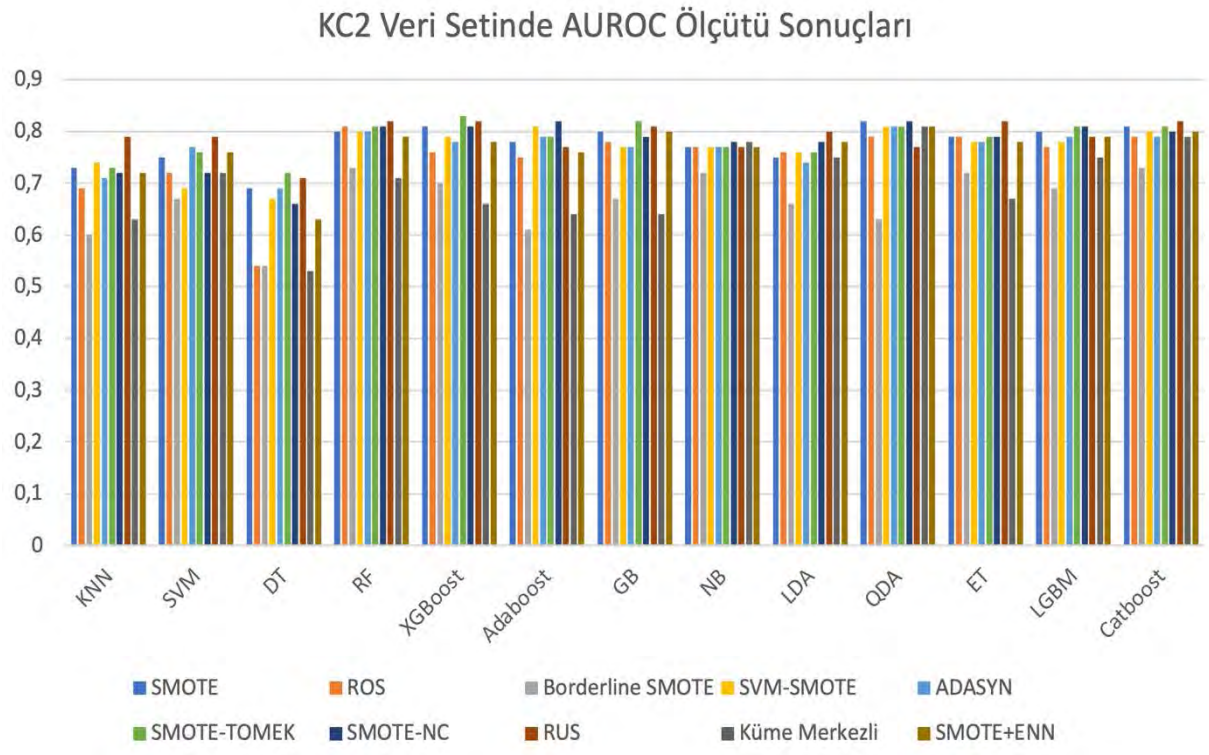
Şekil 3: PC1 veri kümesinde elde edilen AUROC başarım sonuçları grafiği



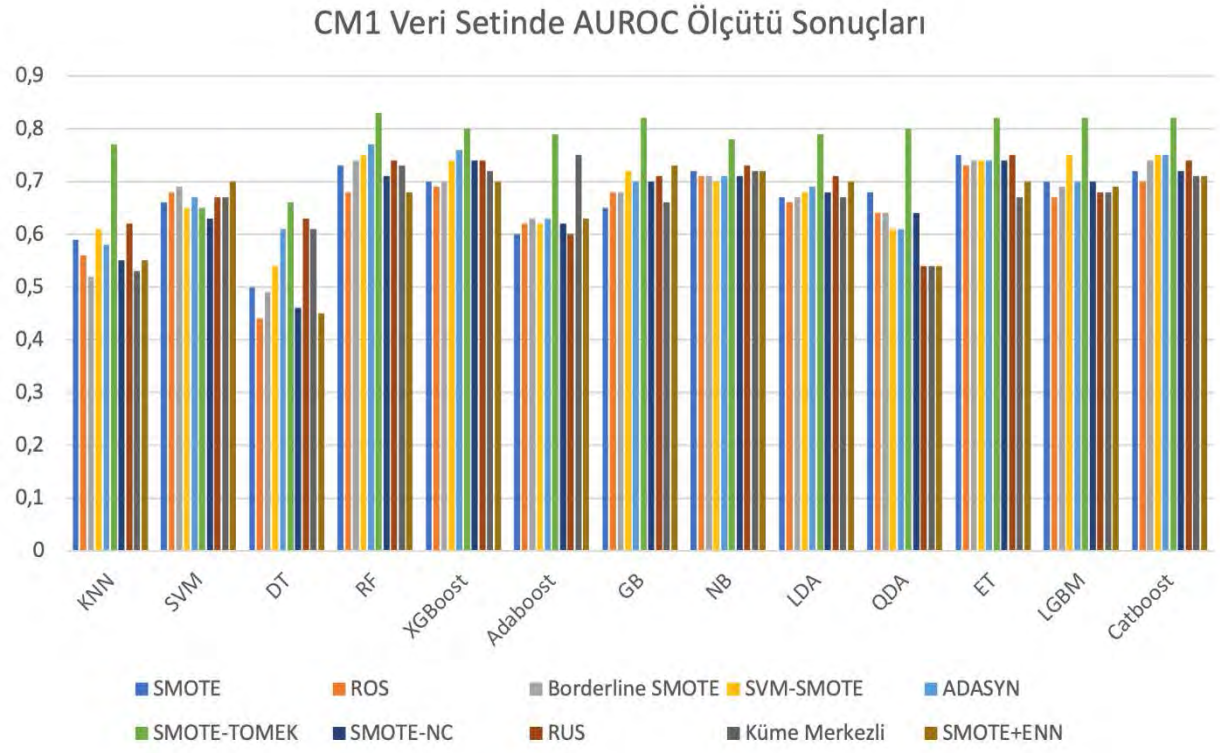
Şekil 4: JM1 veri kümesinde elde edilen AUROC başarım sonuçları grafiği



Şekil 5: KC1 veri kümesinde elde edilen AUROC başarım sonuçları grafiği



Şekil 6: KC2 veri kümesinde elde edilen AUROC başarım sonuçları grafiği



Şekil 7: CM1 veri kümesinde elde edilen AUROC başarım sonuçları grafiği

Çizelge–5: Veri Kümelerinde Örnekleme Yöntemleri ve Sınıflandırma Algoritmalarına göre en iyi Başarım Sonuçları

Veri Kümesi	Doğruluk	En İyi Doğruluk Değerinde Örnekleme Yöntemi ve Sınıflandırma Algoritması	AUROC	En İyi AUROC Değerinde Örnekleme Yöntemi ve Sınıflandırma Algoritması
PC1	0,95	Borderline-SMOTE ve Ekstra Ağaçlar Algroitması	0,92	SVM-SMOTE ve Catboost Algoritması
JM1	0,81	SMOTE-ENN ve Catboost Algoritması	0,75	SVM-SMOTE ve Random Forest Algoritması
KC1	0,87	Borderline-SMOTE ve CAtboost Algoritması	0,84	SMOTE-ENN ve Random Forest Algoritması
KC2	0,89	Borderline-SMOTE ve Light GBM Algoritması	0,83	SMOTE-Tomek ve XGBoost Algoritması
CM1	0,91	SMOTE-ENN ve Light GBM Algoritması	0,83	SMOTE-Tomek ve Random Forest Algoritması

5. Sonuçlar ve Gelecek Çalışmalar

Birçok alanda aktif olarak kullanılan yazılımların kusursuz ve istenilen şekilde olması günümüzde oldukça önem teşkil eden bir konudur. Yazılımın kalitesinde başta gelen en mühim konu yazılımın içerisinde hata bulundurmamasıdır. Yazılım hata tahmininin makine öğrenmesi algoritmaları ile tespit edildiği birçok örnek yöntem bu çalışmada incelenmiştir. Yazılım hata tahmininde kullanılan veri kümelerinde sınıf dengesizliği problemi yer almaktadır. Bu problemin çözümü için örnekleme yöntemleri denenmiştir. Sınıf dengesizliği problemi çözülen veri kümelerinde birçok sınıflandırma algoritması ile sınıflandırma işlemi yapılmış; daha sonra örnekleme yöntemlerine bağlı olarak sınıflandırma başarım sonuçları karşılaştırılmıştır. Deney sonuçlarında PC1 veri kümesinde 0,92 AUC en iyi elde edilen başarım değeri olmuştur.

Bu çalışma büyük yazılım projelerinin en çok hata alan kısımlarında referans alarak hataların tespit edilmesinde kullanılabilir. Güncel yazılım projelerinde en çok hata alınan modüllerin en çok hata alan bölümlerinde etiketler oluşturularak bu çalışmada olduğu gibi hatalar tespit edilebilir. Her bir yazılım projelerinde belirlenen metrikler ve en çok hata alınan modüller referans alınarak bu çalışma ile aynı mantıkta yazılım hata tahmini sistemleri oluşturulabilir.

Bu çalışma ile gelecekte yapılacak yazılım hata tahmininin makine öğrenmesi ve örnekleme yöntemleri yaklaşımları ile çözümleri için motivasyon oluşturabilir.

Farklı veri setlerinde yapılacak çalışmalarda, bu çalışmada kullanılan örnekleme yöntemleri ve algoritmalar bu alandaki araştırmacılar için farklı perspektifler sunacaktır.

Kaynakça

[1] Yalçın, N., & Şimşek Yağlı, B. Teknoloji mağazalarının ISO 25010 kalite modeline dayalı web sitesi kalite değerlendirmesinin çok kriterli analizi: Türkiye örneği, 2020.

[2] Bulut, S. Makine öğrenmesi, Algoritmik Habercilik ve Gazetecilikte İşlevsiz İnsan Sorunsalı, Selçuk İletişim, 2020, 13(1), 294-313.

[3] Aydilek, İ. Yazılım hata tahmininde kullanılan metriklerin karar ağaçlarındaki bilgi kazançlarının incelenmesi ve iyileştirilmesi, Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 2018, 24(5), 906-914.

[4] Çetiner, M. Makine öğrenmesi yöntemleri ile yazılım hata tahmini, Master's thesis, İstanbul Kültür Üniversitesi/Lisansüstü Eğitim Enstitüsü/Bilgisayar Mühendisliği Ana Bilim Dalı/Bilgisayar Mühendisliği Bilim Dalı, 2020.

[5] Sun, Z., Song, Q., & Zhu, X. Using coding-based ensemble learning to improve software defect prediction, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6), 1806-1817.

[6] Eivazpour, Z., & Keyvanpour, M. R. Improving performance in software defect prediction using variational autoencoder, 5th Conference on Knowledge Based Engineering and Innovation (KBEI), 2019, pp. 644-649.

[7] Malhotra, R., Agrawal, V., Pal, V., & Agarwal, T. Support Vector based Oversampling Technique for Handling Class Imblance in Software Defect Prediction, 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2021, pp. 1078-1083.

[8] Choirunnisa, S., Meidyani, B., & Rochimah, S. Software Defect Prediction using Oversamling Algorithm: A-SUWO, Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), 2018, pp. 337-341.

[9] Elahi, E., Ayub, A., & Hussain, I. Two staged data preprocessing ensemble model for software fault prediction, International Bhurban Conference on Applied Sciences and Technologies (IBCAST), 2021, pp. 506-511.

[10] Goyal, S. Handling class-imbalance with KNN (neighbourhood) under-sampling for software defect prediction, Artificial Intelligence Review, 2022, 55(3), 2023-2064.

[11] Jacob, R. J., Kamat, R. J., Sahithya, N. M., John, S. S., & Shankar, S. P. Voting based ensemble classification forsoftware defect prediction, IEEE Mysore Sub Section International Conference (MysuruCon), 2021, pp. 358-365.

[12] Cetiner, M., & Sahingoz, O. K. A comparative analysis for machine learning based software defect prediction systems, 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-7.

[13] Pelayo, L., & Dick, S. Applying novel resampling strategies to software defect prediction, NAFIPS 2007-2007 Annual meeting of the North American fuzzy information processing society, 2007, pp. 69-72.

[14] Aleem, S., Capretz, L. F., & Ahmed, F. Benchmarking machine learning Technologies for software defect detection, arXiv preprint arXiv:1506.07563, 2015.

[15] Ibrahim, D. R., Ghnemat, R., & Hudaib, A. Software defect prediction using feature selection and random forest algorithm, International Conference on New Trends in Computing Sciences (ICTCS), 2017, pp. 252-257.

[16] Akmel, F., Birihanu, E., & Siraj, B. A literatüre review study of software defect prediction using machine learning techniques,Int. J. Emerg. Res. Manag. Technol, 2017, 6(6), 300-306.

[17] Naidu, M. S., & Geethanjali, N. Classification of defects in software using decision tree algorithm, International Journal of Engineering Science and Technology, 2013, 5(6), 1332.

[18] «http://promise.site.uottawa.ca/,» google, [Çevrimiçi]. http://promise.site.uottawa.ca/SERepository/datasets-page.html [Erişildi: 7 Ocak 2023].

[19] Aydin Hakli, D. Sınıf Dengesizliği Sorununu Çözmek için Kullanılan Algoritmaların Farklı Sınıflandırma Yöntemlerinde Performanslarının Karşılaştırılması, 2018.

[20] Yavaş, M., Güran, A. & Uysal, M. Covid-19 Veri Kümesinin SMOTE Tabanlı Örnekleme Yöntemi Uygulanarak Sınıflandırılması, Avrupa Bilim ve Teknoloji Dergisi Ejosat Özel Sayı 2020 (HORA), 2020, 258-264.

[21] Çelik, Ö. & Kaplan, G. Yeniden Örnekleme Teknikleri Kullanarak SMS Verisi Üzerinde Metin Sınıflandırma Çalışması, Erciyes Üniversitesi Fen Bilimleri Enstitüsü Fen Bilimleri Dergisi, 2020, 36 (3), 433-442.

[22] Akın, P. & Terzi, Y. Dengesiz Veri Setli Sağkalım Verilerinde Cox Regresyon ve Rastgele Orman Yöntemlerin Karşılaştırılması, Veri Bilimi, 2020, 3 (1), 21-25.

[23] Öztürk, H. Dengesiz veri setlerinde farklı dengeleme algoritmalarının optimum denge oranlarının sınıflandırma ve regresyon ağaçları yöntemi ile incelenmesi: simülasyon çalışması,2022.

[24] Dal, A., Gümüş, İ. H., Güldal, S. & Yavaş, M. Dengesiz Veriler İçin Ağırlıklı Geometrik Ortalama Tabanlı Yeni Bir Yeniden Örnekleme Yaklaşımı, Adıyaman Üniversitesi Mühendislik Bilimleri Dergisi, 2021, 8 (15), 343-352.

[25] Topal, A., & Amasyalı, M. F. Yapay Örnek Üretimi Ne Zaman İşe Yarar? When does Synthetic Data Generation Work?

[26] UYANIK, F., & KASAPBAŞI, M. C. Telekomünikasyon sektörüiçin veri madenciliği ve makine öğrenmesi teknikleri ile ayrılan müşteri analizi, Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 2021, 9(3), 172-191.

[27] Mukherjee, M., & Khushi, M. SMOTE-ENC: A novel SMOTE-based method to generate synthetic data for nominal and continuous features, Applied System Innovation, 2021, 4(1), 18.

[28] Kaba, G. & Bağdatlı Kalkan, S. Kardiyovasküler Hastalık Tahmininde Makine Öğrenmesi Sınıflandırma Algoritmalarının Karşılaştırılması, İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, 2022, 21 (42), 183-193.

[29] Taşcı, E., & Onan, A. K-en yakın komşu algoritması parametrelerinin sınıflandırma performansı üzerine etkisinin incelenmesi, Akademik Bilişim, 2021, 1(1), 4-18.

[30] Şengül, Z. Makine öğrenmesi algoritmalarını kullanarak bitcoin fiyat tahmini, Master's thesis, Trakya Üniversitesi Sosyal Bilimler Enstitüsü, 2022.

[31] Gumustekin Aydın, S. & Aydoğdu, G. Makine Öğrenmesi Algoritmaları Kullanılarak Türkiye ve AB Ülkelerinin CO2 Emisyonlarının Tahmini, Avrupa Bilim ve Teknoloji Dergisi, 2022, Ejosat Special Issue (ISAS 2022), 42-46.

[32] Şenel Ahmet, F. Makine Öğrenmesi Algoritmaları Kullanılarak Kayısı İç Çekirdeklerinin Sınıflandırılması,2020.

[33] Kalaycı, T. E. Kimlik hırsızı web sitelerinin sınıflandırılması için makine öğrenmesi yöntemlerinin karşılaştırılması, Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi,2020, 24 (5), 870- 878.

[34] Yavuz, Ö. Ç., Karaman, E. & Yeşilyaprak, C. Makine öğrenmesi algoritmalarıyla astronomik gözlem kalitesi tahminine yönelik karar destek sistemi geliştirilmesi ve uygulanması, Trends in Business and Economics, 2022, 36 (3), 289-303.

[35] Yıldırım, E. & Çalhan, A. Apache Spark ile Makine Öğrenmesi Destekli Diyabet Rahatsızlığı Tahmini, Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 2022, 10 (3), 1107-1117.

[36] Sahingoz, O. K., Çebi, C. B., Bulut, F. S., Fırat, H. & Karataş, G. Saldırı Tespit Sistemlerinde Makine Öğrenmesi Modellerinin Karşılaştırılması, Erzincan University Journal of Science and Technology, 2019, 12 (3), 1513-1525.

[37] Güleş, Ş. Makine öğrenmesi yöntemleri ile zararlı yazılım tespiti,Master's thesis, Konya Teknik Üniversitesi, 2020.

[38] YILDIRIM, E. Hızlandırılmış Makine Öğrenmesi Algoritmaları İle Türkçe Sahte Haber Tespiti, 2022.

[39] Serbest, K., & Kılıç, S. A. Diz Eklemi Momentinin Tahmini İçin Makine Öğrenmesi Yöntemlerinin İncelenmesi, Academic Perspective Procedia, 4(1), 341-349, 2021.

[40] Üstüner, M., Abdikan, S., Bilgin, G. & Balık Şanlı, F. Hafif Gradyan Artırma Makineleri ile Tarımsal Ürünlerin Sınıflandırılması, Turkish Journal of Remote Sensing and GIS, Academic Perspective Procedia, 1 (2), 97-105, 2020.

[41] Şahinbaş, K. Price Prediction Model for Restaurants In Istanbul By Using Machine Learning Algorithms, Ekonomi İşletme ve Maliye Araştırmaları Dergisi, 4 (2), 159-171, 2022.

[42] Onan, A. Twıtter Mesajları Üzerinde Makine Öğrenmesi Yöntemlerine Dayalı Duygu Analizi, Yönetim Bilişim Sistemleri Dergisi, 3 (2), 1-14, 2017.

[43] Selimoglu, M. & Yılmaz, A. Kredi Kartı Dolandırıcılık Tespitinin Makine Öğrenmesi Yöntemleri ile Tahmin Edilmesi, Beykent Üniversitesi Fen ve Mühendislik Bilimleri Dergisi, 13 (2), 28-33, 2021.

[44] Cıhan, M., & Ceylan, M. Comparison of linear discriminant analysis, support vector machines and naive bayes methods in the classification of neonatal hyperspectral signatures, 29th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4), 2021.

[45] Çetin, E. Yedinci Servikal Vertebranın Antropometrik Ölçümleri İle Makine Öğrenme Algoritmaları Kullanılarak Cinsiyet Tayini Üzerine Bir Çalışma(Doctoral dissertation), 2021.

[46] Seçgin, Y. Pelvis bilgisayarlı tomografi görüntülerinden elde edilen parametreler ile makine öğrenme algoritmaları kullanılarak cinsiyet tahmini üzerine bir deneme (Master's thesis, Lisansüstü Eğitim Enstitüsü), 2020.

[47] Kartal, E., ve Özen, Z. Dengesiz veri setlerinde sınıflandırma), Mühendislikte Yapay Zekâ ve Uygulamaları, 1st ed, 2017.