T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAZILIM HATA TAHMİN MODELLERİNİN AÇIKLANABİLİR YAPAY ZEKÂ İLE İNCELENMESİ

Hazırlayan
Halime BAYRAKTAR

Danışman
Doç. Dr. Ömür ŞAHİN
İkinci Danışman
Prof. Dr. Bahriye AKAY

Yüksek Lisans Tezi

ŞUBAT 2026
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAZILIM HATA TAHMİN MODELLERİNİN AÇIKLANABİLİR YAPAY ZEKÂ İLE İNCELENMESİ

(Yüksek Lisans Tezi)

Hazırlayan
Halime BAYRAKTAR

Danışman
Doç. Dr. Ömür ŞAHİN
İkinci Danışman
Prof. Dr. Bahriye AKAY

ŞUBAT 2026
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Halime BAYRAKTAR

“Yazılım Hata Tahmin Modellerinin Açıklanabilir Yapay Zekâ ile İncelenmesi” adlı Yüksek Lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi’ne uygun olarak hazırlanmıştır.

Hazırlayan	**Danışman**
Halime BAYRAKTAR	Doç. Dr. Ömür ŞAHİN

Bilgisayar Mühendisliği ABD Başkanı

Prof. Dr. Veysel ASLANTAŞ

TEŞEKKÜR

Lisans ve yüksek lisans eğitimim süresince, akademik birikimi ile mesleki ve beşerî gelişimime katkı sunan, bilimsel bakış açısını bizlere öğreten, yüksek lisans tez çalışmam süresince bilgi, deneyim ve tecrübelerini esirgemeyerek çalışmama ışık tutan tez danışmanlarım Sayın Doç. Dr. Ömür ŞAHİN ve Sayın Prof. Dr. Bahriye AKAY hocalarıma sonsuz teşekkürlerimi sunarım.

Tüm hayatım boyunca desteğini, emeğini esirgemeden ufkumu genişleten, moral ve motivasyon kaynağı olarak her zaman yanımda olan kıymetli annem Hatice TOK'a, saygıdeğer babam Mustafa TOK'a ve tüm aileme, lisansüstü eğitimim süresince beni daima teşvik eden sevgili eşim Oğuz Fatih BAYRAKTAR'a sonsuz teşekkürlerimi sunarım. Bu tez, yazım sürecine başlarken kucağıma aldığım, varlığıyla hayatıma anlam katan biricik oğlum Alper BAYRAKTAR'a hediyemdir.

Halime BAYRAKTAR

Şubat 2026, KAYSERİ

YAZILIM HATA TAHMİN MODELLERİNİN AÇIKLANABİLİR YAPAY ZEKÂ İLE İNCELENMESİ

Halime BAYRAKTAR

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Şubat 2026
Danışman: Doç. Dr. Ömür ŞAHİN

ÖZET

Yazılımda hataların tespit edilmesi kaliteli bir yazılım ve yazılıma olan güvenin oluşturulması için önemlidir. Çalışmada; PROMISE veri tabanında yer alan CM1 ve PC1 veri setleri kullanılarak, hem geleneksel makine öğrenimi modelleri (RF, GB) hem de derin öğrenme modelleri (MLP, LSTM, RNN, CNN) üzerinde kapsamlı bir değerlendirme yapılmıştır. Veri setlerindeki belirgin sınıf dengesizliği SMOTE yöntemiyle giderilmiş ve modeller 5 katlı çapraz doğrulama ile değerlendirilmiştir. Performans ölçütleri olarak Doğruluk, Kesinlik Geri çağırma, F1-Puanı, AUC-ROC kullanılmıştır. Model performansının ötesinde, bu çalışmanın temel amacı Yazılım Hata Tahmini kapsamında yalnızca yazılımın hatalı olup olmadığını belirlemekle kalmayıp; aynı zamanda tahmine en fazla ve en az etki eden özellikleri ortaya koymak, bu özelliklerin sonuca katkılarının yönünü (pozitif/negatif) belirlemek ve bu etkilerin altında yatan nedenleri yorumlayabilmektir. Etkisi diğer özelliklere kıyasla daha düşük olan özellikler, modele gürültü katma potansiyelleri nedeniyle veri setinden çıkarılıp, tekrar performans ölçümü yapılmış ve özellik azaltımı işleminin tahmin performansını artırdığı gözlemlenmiştir. Çalışmanın özgün yönlerinden bir diğeri ise, açıklanabilirlik çıktılarının veri setlerinin istatistiksel korelasyon yapılarıyla karşılaştırılmasıdır. Tüm bu yönleriyle çalışma, SDP alanında performans–açıklanabilirlik bütünleşik analiz çerçevesi sunmakta ve literatürde artan XAI odaklı araştırmalara metodolojik katkı sağlamaktadır.

Anahtar Kelimeler: Yazılım Hata Tahmini, Açıklanabilir Yapay Zekâ, Derin Öğrenme Modelleri, SHAP, LIME

ANALYSIS OF SOFTWARE DEFECT PREDICTION MODELS USING EXPLANABLE ARTIFICIAL INTELLIGENCE

Halime BAYRAKTAR

Erciyes University, Graduate School of Natural and Applied Sciences
Master Thesis, February 2026
Supervisor: Assoc. Prof. Dr. Ömür ŞAHİN

ABSTRACT

Identifying errors in software is crucial for creating high-quality software and building trust in it. In this study, a comprehensive evaluation was conducted on both traditional machine learning models (RF, GB) and deep learning models (MLP, LSTM, RNN, CNN) using the CM1 and PC1 datasets from the PROMISE database. The significant class imbalance in the datasets was addressed using the SMOTE method, and the models were evaluated using 5-fold cross-validation. Accuracy, Precision Recall, F1-Score, and AUC-ROC were used as performance metrics. Beyond model performance, the main objective of this study is not only to determine whether the software is faulty within the scope of Software Defect Prediction, but also to reveal the features that have the most and least impact on the prediction, determine the direction (positive/negative) of these features' contribution to the result, and interpret the underlying reasons for these effects. Features with lower impact compared to other features were removed from the dataset due to their potential to introduce noise into the model, and performance measurement was repeated. It was observed that feature reduction improved prediction performance. Another original aspect of the study is the comparison of explainability outputs with the statistical correlation structures of the data sets. With all these aspects, the study presents an integrated performance-explainability analysis framework in the field of SDP and provides a methodological contribution to the growing XAI-focused research in literature.

Keywords: Software Defect Prediction, Explainable Artificial Intelligence, Deep Learning Models, SHAP, LIME

İÇİNDEKİLER

YAZILIM HATA TAHMİN MODELLERİNİN AÇIKLANABİLİR YAPAY ZEKÂ İLE İNCELENMESİ

BİLİMSEL ETİĞE UYGUNLUK ..... **ii**

YÖNERGEYE UYGUNLUK ..... **iii**

KABUL VE ONAY ..... **iv**

TEŞEKKÜR ..... **v**

ÖZET ..... **vi**

ABSTRACT ..... **vii**

İÇİNDEKİLER ..... **viii**

KISALTMA VE SİMGELER ..... **xi**

TABLOLAR LİSTESİ ..... **xiv**

ŞEKİLLER LİSTESİ ..... **xv**

GİRİŞ **1**

1. BÖLÜM

GENEL BİLGİLER VE LİTERATÜR ÇALIŞMASI

1.1. Problemin Tanımı **5**

1.2. Yazılım Hata Tahmini **6**

1.2.1. Makine Öğrenmesi ile Hata Tahmini 7

1.2.1.1. Gradyan Güçlendirme (GB) 7

1.2.1.2. Rastgele Orman (RF) 7

1.2.1.3. Derin Öğrenme ile Hata Tahmini 8

1.3. Performans Değerlendirme Ölçütleri **10**

1.3.1. Doğruluk 10

1.3.2. Kesinlik 10

1.3.3. Duyarlılık 11

1.3.4. F-1 Puanı 11

1.3.5. AUC-ROC 11
1.4. Açıklanabilir Yapay Zekâ (XAI) 11
1.4.1. Açıklanabilirlik Neden Gereklidir? 12
1.4.2. Açıklanabilir Yapay Zekâ Yöntemleri 14
1.4.2.1. Shapley Eklemeli Açıklamalar (SHAP) 14
1.4.2.2. Yerel Yorumlanabilir Model-Agnostik Açıklanabilirlik (LIME) 14
1.5. Literatür Taraması 15

2. BÖLÜM

YÖNTEM VE MATERYAL

2.1. Veri Setleri 24
2.1.1. NASA PROMISE CM1 25
2.1.2. NASA PROMISE PC1 25
2.2. Veri Ön İşleme ve Hazırlık 28
2.3. Model Geliştirme ve Sınıflandırma 29
2.4. Yapay Zekâ Açıklama Yöntemleri 33
2.4.1. SHAP Yöntemi 33
2.4.1. LIME Yöntemi 35

3. BÖLÜM

DENEYSEL ÇALIŞMALAR

3.1. Deneysel Kurulum 39
3.2. Modellerin Performanslarının Karşılaştırılması 41
3.3. Özelliklerin Tahmin Sürecine Katkısının İncelenmesi (Küresel Açıklanabilirlik) 42
3.3.1. CM1 Veri Seti ile SHAP Yöntemi Kullanımı 43
3.3.2. PC1 Veri Seti ile SHAP Yöntemi Kullanımı 47
3.4. Yerel Açıklamalar Arasındaki Tutarlılığın Değerlendirilmesi 52
3.4.1. CM1 Veri Seti için Yerel Açıklamalar 53
3.4.2. PC1 Veri Seti için Yerel Açıklamalar 56
3.5. Özellik Azaltımı Uygulamasının Model Performansına Etkisi 60
3.6. Korelasyon Tabanlı İlişki Analizi (Pearson Korelasyonu) 62

4. BÖLÜM

TARTIŞMA, SONUÇ VE ÖNERİLER

4.1. Tartışma, Sonuç ve Öneriler 68

KAYNAKÇA 72

ÖZGEÇMİŞ 81

KISALTMA VE SİMGELER

AdaBoost : Adaptive Boosting (Uyarlanabilir Güçlendirme)

AUC-ROC : Area Under the Receiver Operating Characteristic Curve (Alıcı İşletim Karakteristik Eğrisi Altındaki Alan)

AutoSperman : Automatically mitigating correlated software metrics for interpreting defect models (Kusur Modellerini Yorumlamak İçin İlişkili Metrikleri Otomatik Olarak Azaltma)

avNNet : Neural Networks Using Model Averaging (Ortalama Sinir Ağı)

BERTopic : Dönüştürücü tabanlı bir konu modelleme tekniği

BreakDown : Break Down Explanation of Model Prediction (Tahminin Bileşenlerine Ayrıştırılması Yöntemi)

CatBoost : Category Boosting (Kategorik Güçlendirme)

CfExplainer : Karşıt olgulara dayalı açıklanabilir tam zamanında kusur tahmini.

CNN : Convolutional Neural Network (Evrişimli Sinir Ağı)

Deeper : Hataya eğilimli değişiklikleri tahmin etmek için derin öğrenme yöntemlerinden faydalanan bir yaklaşımdır.

DP-CNN : Deep Pyramid Convolutional Neural Networks (Derin Piramit Evrişimli Sinir Ağları)

DPNN : Derin öğrenme sinir ağı tabanlı kusur tahmini.

DT : Decision Tree (Karar Ağacı)

ELI5 : Explain Like I’m 5 (5 Yaşındaymışım Gibi Açıkla)

ET : Extraa Trees (Ekstra Ağaçlar)

GB : Gradient Boosting (Gradyan Güçlendirme)

GRU : Gated Repeating Units (Kapılı Tekrarlayan Birimler)

JIT : Just-In-Time (Tam Zamanında Hata Tahmini)

KNN : K-Nearest Neighbors (K-En Yakın Komşu)

LIME : Local Interpretable Model-Agnostic Explanations (Yerel Yorumlanabilir Modelden Bağımsız Açıklamalar)

LIME-HPO : LIME with Hyper Parameter Optimisation (Model Agnostik Açıklama tekniği ile Hiper Parametre Optimizasyonu)

LR : Logistic Regression (Lojistik Regresyon)

LSTM : Long Short-Term Memory (Uzun Kısa Süreli Bellek)

MAE : Mean Absolute Error (Ortalama Mutlak Hata)

MLP : Multi-Layer Perceptron (Çok Katmanlı Algılayıcı)

MSE : Mean Squared Error (Ortalama Kare Hata)

NBC : Naive Bayes Classifier (Naive Bayes Sınıflandırıcısı)

NN : Neural Network (Sinir Ağı)

PDP : Partial Dependence Plot (Kısmi Bağımlılık Grafiği)

ProtoDash : Fast Interpretable Prototype Selection (Hızlı Yorumlanabilir Prototip Seçimi)

RENN : Recursively Regularized Nearest Neighbors (Tekrarlanan Düzenlenmiş En Yakın Komşular)

RF : Random Forest (Rastgele Orman)

RMSE : Root Mean Square Error (Karekök Ortalama Kare Hata)

RNN : Recurrent Neural Network (Yinelemeli sinir ağı)

SDP : Software Defect Prediction (Yazılım Hata Tahmini)

SHAP : SHapley Additive ExPlanations (SHapley Katkı Açıklamaları)

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

SMOTE : Synthetic Minority Oversampling Technique (Sentetik Azınlık Aşırı Örnekleme Tekniği)

SP-LIME : Alt modüler optimizasyon yoluyla "modele güvenme" sorununu ele almak için açıklamalarla birlikte bir dizi temsili örnek seçen bir yöntemdir.

SQAPlanner : Yazılım hata tahmini modelleri tarafından tahmin edilen kusurların olasılığını azaltmak için yazılım modüllerinde değişiklikler öneren, ilişki kuralı madenciliği algoritmasına dayalı olan bir araçtır.

SVM : Support Vector Machines (Destek Vektör Makineleri)

TimeLIME : Yazılım mühendisliğinden alan bilgisini dahil ederek açıklamaları iyileştiren yaklaşımlara örnektir.

XAI : Explainable Artificial Intelligence (Açıklanabilir Yapay Zekâ)

XGBoost : Extreme Gradient Boosting (Aşırı Gradyan Güçlendirme)

XGBR : Extreme Gradient Boosting Regressor (Aşırı Gradyan Güçlendirici Regresör)

TABLOLAR LİSTESİ

Tablo 1.1. Literatür Özeti 20
Tablo 2.1. CM1 ve PC1 Veri setlerinin Özelliklerinin Ayrıntıları 27
Tablo 3.1. Kullanılan Modellerin Yapı ve Hiperparametre Özeti 40
Tablo 3.2. CM1 ve PC1 Metriklerinin Sonuçları 41
Tablo 3.3. CM1 Metriklerinin Sonuçları 60
Tablo 3.4. PC1 Metriklerinin Sonuçları 61
Tablo 3.5. CM1 veri seti için korelasyon sonuçları 64
Tablo 3.6. PC1 veri seti için korelasyon sonuçları 66

ŞEKİLLER LİSTESİ

Şekil 2.1. CM1 Veri seti Dağılımı 25

Şekil 2.2. PC1 Veri Seti Dağılımı 26

Şekil 3.1. CM1 Veri Seti için SHAP yöntemi kullanılarak hesaplanan özellik önem grafikleri 44

Şekil 3.2. CM1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP özet grafikleri. 46

Şekil 3.3. PC1 Veri Seti için SHAP yöntemi kullanılarak hesaplanan özellik önem grafikleri 48

Şekil 3.4. PC1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP özet grafikleri. 50

Şekil 3.5. CM1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP şelale grafikleri. 54

Şekil 3.6.a CM1 veri setinin 1. Örneği için LIME Sonuçları 55

Şekil 3.6.b CM1 veri setinin 5. Örneği için LIME Sonuçları 55

Şekil 3.6.c CM1 veri setinin 15. Örneği için LIME Sonuçları 55

Şekil 3.7. PC1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP şelale grafikleri. 57

Şekil 3.8.a PC1 veri setinin 1. Örneği için LIME Sonuçları 58

Şekil 3.8.b PC1 veri setinin 15. Örneği için LIME Sonuçları 58

Şekil 3.8.c PC1 veri setinin 30. Örneği için LIME Sonuçları 59

Şekil 3.9. CM1 veri setinin ısı haritası 63

Şekil 3.10. PC1 veri setinin ısı haritası 65

GİRİŞ

Yazılım, bilgisayar sistemlerinin çalışmasına imkân sağlayan programlar, prosedürler ve bunlarla ilişkili verilerin bütününü oluşturan [1], sosyal yaşamı etkileyen önemli faktörler arasındadır. Güvenilirlik açısından kritik öneme sahip sistemler, işlevselliklerini sürdürebilmek için kullandıkları yazılımın doğruluk ve kararlılığına yüksek düzeyde bağımlıdır. Yazılımların hem kullanımının artması hem kullanıcı gereksinimi hem de yazılım kapsamının genişlemesiyle yazılımlardaki karmaşıklık ve yazılım hataları gün geçtikçe artmış ve bu durum yazılım geliştirme süreçlerinde güvenilirlik, bakım ve maliyet açısından çeşitli zorluklar ortaya çıkarmıştır [2]. Yazılıma olan güvenilirlik ve yazılım kalitesi, yazılıma ait hata ve eksikliklerin giderilmesi ile doğrudan ilişkilidir [3].

Yazılım sistemlerindeki hata, ilerleyen aşamalarda sistemin başarısızlıkla sonuçlanmasına neden olabilecek yapısal bir hata örüntüsü olarak tanımlanmaktadır [4]. Yazılım hataları, yazılımların arıza veya çökme durumlarına yol açan [5], [6] ve yazılım geliştirme sürecinde geliştiricilerin yaygın olarak karşılaştığı bir problemdir [7]. Hataların, yazılım kalitesi ve yazılım maliyeti üzerinde etkisi fazladır [6]. İstatistiklere göre, yazılım geliştirme sürecindeki toplam maliyetin %50 ile %75'i hataların giderilmesine harcanmaktadır [6], [8]. Yazılım hata tahmini ve bu hataların yazılım kullanıma sunulduktan sonra düzeltilmesi süreci, yazılımın güvenilirliğini ve bakım etkinliğini doğrudan etkileyen, en maliyetli aşamalardan biridir [9]. Yazılım hata tahmininin otomatikleştirilmesi ise, yazılım maliyetlerinin azaltılmasına katkı sağlamaktadır [8], [10]. Araştırmacılar yazılımda hata problemini ortadan kaldırmak için çok sayıda yöntem denemişlerdir. Literatürde öne çıkan çalışma alanlarından biri, geliştiricilerin yazılım hatalarını üretim ortamına ulaşmadan önce proaktif bir şekilde tespit edip ele alınmasına olanak tanıyan, yazılımın güvenilirliğini artırmakta ve bakım maliyetlerini azaltmaya katkıda bulunan makine öğrenimi tabanlı hata tahmin yöntemleridir [7].

Yazılım hata tahmini (Software Defect Prediction, SDP), yazılım geliştirme sürecinin erken dönemlerinde hataların fark edilmesi, yazılım kalitesinin artırılması ve bakım çabasını azaltmak açısından önemli bir aşamadır. Bu aşama, hataların zamanında giderilmesine olanak sunduğu gibi güvenilir ve sürdürülebilir yazılımların geliştirilmesine imkân tanımaktadır [4].

Günümüzde yapay zekâ ve makine öğrenmesinin gelişim göstermesiyle kullanımı birçok kritik alanda yaygınlaşmıştır. Örneğin, tıp alanı, sürücüsüz araç teknolojileri, yatırım, portföy yönetimi ve adalet sistemleri gibi hata toleransı düşük olan alanlarda, yalnızca doğru tahminler üretilmesi yeterli değildir, bunun yanında şeffaf, açıklanabilir ve güvenilir karar alma süreçleriyle desteklenmesi de aynı oranda önemlidir [11]. Bazı makine öğrenimi modellerinin "kara kutu" yapısı, karar verme sürecini ve bu kararlarda etkili olan faktörleri anlamayı güçleştirmekte, dolayısıyla modelin davranışlarına ilişkin içgörü elde etmeyi zorlaştırmaktadır. Yüksek performansın yanı sıra, modellerin geliştiriciler ve kullanıcılar tarafından anlaşılabilir olması kritik bir gereklilik olarak ortaya çıkmaktadır. Kara kutu modellerinin güvenlik gerektiren bazı sorulara cevap verebilmesi için de açıklanabilirliğe ihtiyaç duyulmaktadır. Son kullanıcı ve geliştiricilerin tahmini anlaması, sisteme olan güvenini artırmakta olup modelin kullanımına yönelik motivasyonlarını güçlendirmektedir. Model ya da tahminlere ilişkin içgörü edinilmesi, sistemin şeffaflığını artırmakta olup özelliklerin katkı düzeyine ve insani karar süreçlerine dair bilgi sunmakta ve nihayetinde güven oluşturmaktadır [12].

Yapay zekânın gelişmesi ile makine öğrenmesi ve alt dalı olan derin öğrenme modellerinin karar alma aşamasındaki şeffaflık eksikliği, açıklanabilir yapay zekâya ihtiyaç olduğunu göstermektedir [13]. Karar alma sürecindeki şeffaflık eksikliği ve veri dengesizliği, geleneksel SDP modellerinin karşılaştığı problemler arasındadır. Bu sınırlılıklar, uygulama ortamlarında SDP modellerine olan güveni azaltmaktadır [14]. Şeffaflık, yüksek güvenlik gerektiren kritik sistemlerde SDP'nin kalite düzeyini iyileştirme ve hataların daha hızlı ve doğru biçimde giderilmesi yoluyla yazılım geliştirme maliyetinin düşürülmesi için önemlidir. Ayrıca modellerin açıklanabilir olması, bu modellere duyulan güvenin artmasına katkı sağlamaktadır [15].

Modellerin açıklanabilir olması gerekliliği göz önünde bulundurulduğunda Açıklanabilir Yapay Zekâ (Explainable Artificial Intelligence, XAI), bu probleme çözüm getiren etkili bir yaklaşım olarak literatürde yerini almıştır. XAI, yüksek doğruluk ve model

anlaşılabilirliğini sağlamak amacıyla, makine öğrenimi gibi yüksek düzeyde karmaşık modellerin belirli çıktılara hangi nedenlerle ve nasıl ulaştığını ortaya koyarak "kara kutu" yapısını azaltmaktadır [16], [17]. XAI, Yapay zekâ modellerinin karar mekanizmalarını ortaya koyarak, bu süreçlerin son kullanıcı ve geliştiriciler için anlaşılmasını kolaylaştıran yöntemler sunmaktadır [18]. Çalışmamızda CM1 ve PC1 veri setlerini farklı modellerle kullanarak yazılım hata tahmini performansını, SHapley Katkı Açıklamaları (SHapley Additive ExPlanations, SHAP) ve Yerel Yorumlanabilir Modelden Bağımsız Açıklamalar (Local Interpretable Model-Agnostic Explanations, LIME) yöntemleri ile yerel ve küresel açıklanabilirlik sağlanarak, modelin karar verme süreci şeffaflaştırılarak yorumlanabilirlik ve güvenilirlik düzeyi artırılmaktadır.

Bu çalışmanın temel amacı, makine öğrenmesi ve derin öğrenme modellerinin kullanıldığı yazılım hata tahmini tespitinde yazılımın sadece hatalı olduğunu veya olmadığını belirlemek değil, bunun yanında hatanın neden oluştuğunu anlamlandırabilmek, şeffaflığı artırarak geliştiriciler tarafından yorumlanabilir kılabilmektir. Kullanılan yöntemlerde gerek modellerin farklılığı gerekse veri setinin eğitim ve test için rastgele ayrılmış olması bulguların birbirleriyle birebir örtüşmesini engellemektedir. Amacımız elde edilen sonuçlara göre yazılım hata tahminine en çok ve en az etki eden özellikleri belirlemek, bu özelliklerin etkilerinin sonuca katkısının pozitif mi negatif mi olduğunu belirlemek ve nedenlerini anlamlandırabilmektir. Etkisi diğer özelliklere kıyasla daha düşük olan özellikler, modele gürültü katma potansiyelleri nedeniyle veri setinden çıkarılıp, tahmin performansını tekrar ölçerek tüm veri setinin performansıyla karşılaştırmayı hedeflemekteyiz. Bu tez çalışmasında aşağıda belirtilen araştırma sorularına cevap aranmaktadır:

- **AS1**: Aynı veri ön işleme koşulları altında hangi model yazılım hata tahmini için daha iyi performans gösterir?
- **AS2**: Veri setlerinin sunduğu özelliklerden hangilerinin tahmin sonucuna katkısı daha fazladır?
- **AS3**: Küresel (tüm veri sette) ve yerel (aynı örnek ile) açıklanabilirlikte elde edilen bulgulara göre yöntemler benzer sonuçlar üretebilir mi, kullanılan modeller arasında en çok benzer sonuçlar elde eden model hangisidir?
- **AS4**: Veri seti üzerinde, özellik azaltımı uygulandığında modellerin performans metriklerinde nasıl değişimler gözlemlenmektedir?

Bu tez dört ana bölümden oluşmaktadır. Birinci bölümde yazılım hata tahmini ve açıklanabilir yapay zekânın gerekliliğinden, araştırma probleminin tanımı, çalışmanın amacı ve kapsamı sunulmuş, ayrıca konuyla ilgili literatür incelemesine yer verilmiştir. İkinci bölümde, araştırmada kullanılan yöntemler, veri setleri, makine öğrenimi ve derin öğrenme modelleri ile XAI yöntemleri ayrıntılı biçimde açıklanmıştır. Üçüncü bölümde deneysel çalışmalardan elde edilen bulgular sunulmuş, performans metrikleri ile modellerin doğruluk ve yorumlanabilirlik açısından değerlendirilmesi yapılmıştır. Dördüncü bölümde ise bulgular tartışılmış, sonuçlar özetlenmiş ve gelecek çalışmalar için öneriler geliştirilmiştir. Tezin sonunda kaynakça ve ekler bölümleri bulunmaktadır.

1. BÖLÜM

GENEL BİLGİLER VE LİTERATÜR ÇALIŞMASI

1.1. Problemin Tanımı

Son dönemlerde bilgi ve iletişim teknolojilerinin yaygınlaşması ile yazılım kullanımının artması, yazılım kalitesi teriminin önemini artırmıştır. Yazılım kalitesi, yazılımın isteklere cevap verebilme düzeyi, doğru ve optimum seviyede çalışması vb. ölçütlerle belirlenmektedir. Bu aşamada yazılım testi, kaliteli bir yazılım için kaçınılmaz bir adım haline gelmiştir. Yazılım testi ise yazılımın kendisi kadar iş gücü ve zaman kullanımına neden olmaktadır. Benzer şekilde SDP kavramı da işgücü, zamandan tasarruf sağlamak ve yazılım maliyetlerini azaltabilmek amacıyla ortaya çıkmıştır.

SDP ve hatanın erken tespiti ile önlenmesi, yüksek kaliteli ve güvenilir yazılım sistemlerinin geliştirilmesinde temel bir gerekliliktir. Yazılım geliştirme sürecinde belirli ilkelere dikkat edilmesi hem hata oranlarının azaltılmasına hem de genel yazılım kalitesinin artırılmasına önemli katkılar sunmaktadır. Öncelikle, gereksinim analizinin dikkatli ve ayrıntılı bir biçimde gerçekleştirilmesi, hataların daha sürecin başında ortaya çıkmasını engelleyebilir [19]. Kodun modüler, okunabilir ve yeniden kullanılabilir biçimde yazılması, geliştirici ekipler arasındaki iş birliğini kolaylaştırırken aynı zamanda hata ayıklama süreçlerinin daha etkin yürütülmesini sağlar. Ayrıca, kod incelemeleri (code reviews), otomatik testler, sürekli entegrasyon ve versiyon kontrolü gibi yazılım mühendisliği uygulamalarının sistemli biçimde kullanılması, potansiyel hataların erken aşamalarda tespit edilmesini sağlar [20]. Bununla birlikte, açıklanabilir yapay zekâ yöntemlerinin entegre edilmesi, sadece hataları tahmin etmekle kalmayıp, bu hataların neden oluştuğunu da açıklayarak geliştiricilerin daha bilinçli kararlar almasına yardımcı olabilir [16], [21]. SDP alanında yüksek performans elde eden yapay zekâ modelleri

taşıdıkları kara-kutu yapısı nedeniyle karar alma süreçlerini anlaşılır biçimde ortaya koymakta yetersiz kalmakta ve bu durum XAI yöntemlerinin kullanımını gerekçelendirmektedir [16]. Tüm bu unsurlara dikkat edilmesi, hem proaktif hata önleme stratejilerinin uygulanmasını destekler hem de yazılım yaşam döngüsü boyunca kaliteyi ve güvenilirliği sürdürülebilir kılar.

Çalışmamızda, karmaşık derin öğrenme ve makine öğrenmesi modellerini anlamlandırabilmek amacıyla SHAP ve LIME yöntemleri kullanılarak yerel ve küresel açıklanabilirlik analizleri gerçekleştirilmiş, böylece modelin karar süreçlerinde hangi özelliklerin neden ve ne ölçüde etkili olduğuna dair daha sistematik ve yorumlanabilir bir çerçeve sunulmuştur. SHAP ve LIME literatürde sıkça kullanılan ve açık kaynak olarak sunulan yöntemlerdir [4]. Özelliklerin önem sıralaması ve etkisinin açıklanması için SHAP küresel bir görünüm sunarken, LIME bireysel tahminler ve yerel açıklamalar sunar [18].

1.2. Yazılım Hata Tahmini

SDP, 1970'lerden beri yazılım mühendisliğinin üzerine en çok çalışılan araştırma alanlarından biridir [6]. Temel amaç, belirli bir kod parçasının veya dosyanın hata içerip içermediğini öngörmektir [22]. Halka açık popüler veri setlerinden NASA [23] ve PROMISE [24] veri setleri SDP çalışmalarında sıklıkla kullanılmıştır. SDP'de yazılım hatalarının bazıları derleyici çıktıları veya bayt kod gösterimleri vb. yazılımsal olmayan etkenlerden kaynaklanır fakat temel sebebi yazılım kodudur [13].

SDP, kod bloklarını hatalı ve hatasız olarak ayırmayı hedefler [25]. SDP için 2000-2013 yılları baz alınarak yapılan bir literatür taramasında, sınıflandırma yöntemlerinin daha fazla kullanıldığını belirtmişlerdir. Sınıflandırma yöntemlerinden de en yaygın kullanılanları: RF, KNN, NN, SVM, LR, NBC ve DT şeklinde listelemişlerdir [26].

SDP yöntemleri sınıf dengesizliği problemiyle karşı karşıya kalmaktadır. Kullanılan veri setindeki sınıf dengesizliği, SDP performansının değerlendirilmesinde kullanılan metriklerin güvenilirliğini azaltmakta ve yanıltıcı sonuçlar ortaya çıkmasına neden olabilmektedir [27]. SDP performansını artırmaya yönelik, özellikle sınıf dengesizliğinin neden olduğu sorunları azaltmak amacıyla aşırı örnekleme gibi çeşitli veri ön işleme tekniklerinden yararlanılmaktadır [10]. Bu teknikler, sınıflar arasındaki veri dağılımını

dengeleyerek modellerin hem doğruluk hem de genelleme kabiliyetini geliştirmeye katkı sağlamaktadır.

SDP modelleri, yazılım projelerinden elde edilen geçmiş verilere dayalı olarak inşa edilmektedir [28]. Mevcut makine öğrenmesi tabanlı SDP modelleri, özellik çıkarım sürecinde yazılım metriklerine bağımlı olup bu süreci manuel olarak yürütmektedir. Manuel özellik çıkarımı, fazla zaman alan maliyetli bir işlem olup, hata raporlarında bulunan anlamsal bilgileri yakalama konusunda yetersiz kalmaktadır. Derin öğrenme modelleri ise özellik mühendisliğini otomatikleştirerek karmaşık ve çok boyutlu verilerden üst düzey özellikleri kendiliğinden çıkarma kapasitesine sahiptir [29]. Bu doğrultuda, son yıllarda birçok araştırmacı, derin öğrenme modellerini kullanarak yazılım hata tahmini yöntemlerini geliştirmeye yönelmiştir [10].

XAI destekli SDP modelleri, hem geliştiricilere hata tahminlerinin nedenlerini anlayarak kritik alanlara odaklanmalarına imkân sağlar hem de proje yöneticilerinin kaynak, zaman ve risk yönetimi konusunda daha bilinçli kararlar almalarına olanak tanır [9].

1.2.1. Makine Öğrenmesi ile Hata Tahmini

1.2.1.1. Gradyan Güçlendirme (GB)

Esnek yapısı ve yazılım hata tahmininden elde edilen bulgulara göre başarılı performans sergilemesi sınıflandırma problemlerinde sıkça kullanılan, zayıf öğrenicilerin aşamalı biçimde bütünleştirilmesi yoluyla güçlü bir tahmin modeli elde eden toplu öğrenme yöntemidir [30].

GB modeli temel olarak, ilk aşamada bir tahmin fonksiyonu tanımlar. Tahminler ile gerçek gözlem değerleri arasındaki farklar dikkate alınarak kayıp fonksiyonu tanımlanır. İkinci aşamada, tahmin ve kayıp fonksiyonları birlikte değerlendirilerek tahmin değerleri ile gözlem değerleri arasındaki fark yeniden hesaplanmaktadır. Bu süreçte, her adımda modele yeni bileşenler eklenerek tahmin fonksiyonunun doğruluk düzeyinin artırılması hedeflenmektedir. Amaç, model tarafından üretilen tahmin değerleri ile gerçek gözlem değerleri arasındaki farkın olabildiğince sıfıra yaklaşmasını sağlamaktır [31].

1.2.1.2. Rastgele Orman (RF)

Birden fazla karar ağacından oluşan öğrenme modeli [5], ağaçlar arasındaki ilişkiyi en aza indirerek yeni ve daha önce karşılaşılmamış tahmin doğruluk performansını artırır,

ağaçların arasındaki ilişkiyi en aza indirgerken rastgelelikten faydalanılır [5], [32]. Literatürdeki çalışmalarda, RF makine öğrenmesi modelinin SDP problemlerinde yüksek başarı elde ettiği gözlemlenmiştir [32], [33].

1.2.1.3. Derin Öğrenme ile Hata Tahmini

Yazılım hata tahmini çoğunlukla, makine öğrenmesi modelleri ile çalışılmıştır. Fakat mevcut makine öğrenimi tabanlı yaklaşımlar hem zahmetli ve zaman alıcı hem de hata raporlama araçlarında yer alan anlamlı bağlamları büyük ölçüde göz ardı eden manuel özellik çıkarımına ihtiyaç duymaktadır [10]. Bu problem üzerine yoğunlaşıldığında SDP performansını artırmaya yönelik olarak farklı derin öğrenme modelleri incelenmiştir [25]. Son yıllarda derin öğrenme, makine öğrenmesi literatüründe öncelikli araştırma alanlarından biri haline gelmiştir [34]. Yang vd. tarafından [34], çalışmalarında derin inanç ağlarından faydalanarak bir dizi özellik oluşturup derin öğrenme tabanlı ilk hata tahmin modeli olan Deeper önerilmiştir. Yang'ın öncü araştırması, daha sonra geliştirilen birçok derin öğrenme tabanlı yaklaşımın ilham kaynağı olmuştur. Bunlara örnek vermek gerekirse, Hoang vd. [35] Deeper'dan daha iyi performans gösteren konvolüsyonel sinir ağı tabanlı bir hata tahmin modeli önermiştir. Qiao vd. [36] tarafından DPNN adlı, Li vd. [37] tarafından ise DP-CNN adlı derin öğrenme tabanlı hata tahmin modelleri tanıtılmıştır. Bu çalışmaların bulgularına göre derin öğrenme modelleri, hata tahmininin performansını kayda değer ölçüde iyileştirmiştir ve literatürde giderek daha fazla araştırmacının ilgisini çeken bir alan haline gelmiştir [25].

Derin öğrenme modelleri, doğru tahmin için şeffaflık ve açıklanabilirlikten ödün veren makine öğrenimi modelleridir. Katmanlı ve karmaşık yapısı, girdiler ile çıktılar arasında doğrusal olmayan ilişkiler oluşturduğundan, modelin açıklanabilirliği geliştiriciler ve kullanıcılar açısından kısıtlıdır. Bu modeller, sonuca nasıl ulaşıldığına dair herhangi bir gerekçe sunmadıkları nedeniyle şeffaf olmayan modeller olarak değerlendirilir [15]. Öte yandan geliştiricilere yüksek boyutlu ve karmaşık verilerden otomatik olarak özellik çıkarma ve öğrenme olanağı sağlayan [10] bu modeller, kapsamlı ve güncel yöntemler içerir. Bu nedenle, literatürdeki birçok çalışma söz konusu yöntemleri SDP alanına uygulamakta ve elde edilen bulguları mevcut çalışmalarla karşılaştırmaktadır [11]. Son zamanlarda derin öğrenme modellerinde, yazılım hata tahmininde performans ile açıklanabilirlik arasındaki dengeyi ele alan post-hoc açıklanabilirlik yöntemleri üzerine önemli ilerlemeler katedilmiştir. Bu yöntemler, derin öğrenme gibi kara kutu modellerini,

basit ve açıklanabilir modellerin sağladığı yaklaşık temsiller yoluyla modelin kararlarını açıklamayı hedefler. Böylece, derin öğrenme modellerinin tahmin süreçlerini şeffaflaştırarak onları kara kutudan, beyaz kutuya dönüştürür. Bu nedenle XAI olarak adlandırılan bu yöntemler, SDP'de geliştiricilere ve kullanıcılara hem yüksek tahmin doğruluğu hem de açıklanabilirlik sayesinde yorumlanabilirlik sunduğu için konu üzerine olan ilgi giderek artmaktadır [38].

Tekrarlayan Sinir Ağları (RNN), mimarileri sahip olduğu bilgiyi kısa süreli olarak ezberlenilmesine olanak tanıyan geri besleme mekanizmasına sahiptir. RNN'ler, hesaplama hatası olabildiği gibi patlayan veya kaybolan gradyanlar sorunu ile karşı karşıyadır. Bu tür problemleri çözebilmek amacıyla ağda bulunan bilgi akışını düzenleyen kapılar adıyla ek sinir ağları önerilmiştir. Chung vd. [39] tarafından, tanıtılan Kapılı Tekrarlayan Birimler (Gated Repeating Units, GRU) bir RNN türüdür. GRU'ların sahip olduğu kapılı yapı, RNN'lerin uzun dizilerdeki bilgi aktarım sorunu, güncelleme ve sıfırlama kapıları sayesinde basitleştirilmiş bir mimari aracılığıyla çözülmektedir [10].

Evrişimli Sinir Ağı (CNN), girdi verilerinden anlamlı özellikleri otomatik olarak çıkarabilmek için evrişim işlemini kullanan CNN, insan görsel sisteminin işleyişinden esinlenir. Evrişim işlemi, girdinin farklı bölgelerinde yer alan kenar, köşe, doku ve desen gibi yerel özellikleri bilgilerini sunar. Böylece CNN modelleri, düşük seviyeli özelliklerden başlayarak daha karmaşık ve soyut temsillere ulaşan çok katmanlı bir öğrenme süreciyle, insan görsel korteksinin bilgi işleme biçimini taklit eder [10].

Uzun Kısa Süreli Bellek (LSTM), RNN'in kaybolan gradyan ve patlayan gradyan problemlerine çözüm bulabilmek amacıyla Hochreiter ve Schmidhuber tarafından RNN'in bir türü olan LSTM modeli önerilmiştir [10]. LSTM modeli, işlenen verilerdeki örüntüleri analiz etmek amacıyla tasarlanmıştır. Kaybolan Gradyan (Vanishing Gradient) sorununu aşmak amacıyla geliştirilen LSTM birimi; giriş kapısı, çıkış kapısı ve unutma kapısı olmak üzere üç temel kapıdan oluşmaktadır. Bu kapılar aracılığıyla model, önemli bilgileri uzun süre belleğinde tutabilmesine ve önemsiz olanları silmesine imkân tanır [3], [7].

Çok Katmanlı Algılayıcı (MLP), birden çok sayıda katmandan meydana gelir. Modele ait ilk bileşen giriş katmanıdır, giriş katmanı vektör biçiminde temsil edilen giriş verilerini ifade etmektedir. Diğer bileşenler ise birbirleriyle bağlantılı düğümlerden oluşan gizli

katmanlardır. Söz konusu katmanlar sayesinde model, girdi verileri üzerinde çok katmanlı işlem adımları gerçekleştirerek verileri farklı biçimlerde dönüştürebilmektedir. Doğrusal olmayan aktivasyon fonksiyonlarının kullanımı ve geri yayılım optimizasyon süreci aracılığıyla ağın ağırlık ve önyargı parametrelerini düzenli olarak güncellenmekte ve modelin öğrenme performansı iyileştirilmektedir. Katmanlar aracılığıyla elde edilen soyut ve yüksek düzeyli özellikler, modelin temsiliyet gücünü artırmaktadır. Nihai aşamada çıkış katmanı, girdilerin ait oldukları sınıfları veya hedef değerleri temsil eden çıktı vektörlerini üreterek sınıflandırma ya da regresyon sürecini tamamlamaktadır [10].

1.3. Performans Değerlendirme Ölçütleri

Tahmin yapıldıktan sonra varlık için dört farklı olası sonuç vardır:

Gerçek pozitif, TP; Hatalı verinin doğru olarak sınıflandırılması ile hatalı veri sayısını gösterir.

Yanlış Negatif, FN; Hatalı verilerin hatalı olarak sınıflandırılması ile hatalı verilerin sayısını gösterir.

Gerçek Negatif, TN; Hatasız verinin doğru olarak sınıflandırılması ile hatasız veri sayısını gösterir.

Yanlış Pozitif, FP; Hatalı olduğu tespit edilen fakat aslen hatasız olan verilerin sayısını gösterir [32].

1.3.1. Doğruluk

Doğru sınıflandırılan varlıkların sayısının toplamının, tüm varlıkların sayısının toplamına bölümü ile bulunur [40].

$$\text{Doğruluk} = (TP+TN) / (TP+TN+FP+FN) \quad (1.1)$$

1.3.2. Kesinlik

Sonuçların ilişkisini ölçen metriktir, hatalı verinin doğru olarak sınıflandırıldığı verilerin (TP), hatalı bir şekilde sınıflandırılan tüm verilere oranını hesaplar [40].

$$\text{Kesinlik} = TP/(TP+FP) \quad (1.2)$$

1.3.3. Duyarlılık

Tahmin verilerinden hatalı sonuç vermesi gerekenlerden kaç tanesinin hatalı olduğunu gösteren metriktir [28]. Hatalı verinin doğru olarak sınıflandırıldığı verilerin (TP), tüm hatalı verilere (TP + FN) oranını hesaplar [40].

$$\text{Duyarlılık} = TP/(TP+FN) \tag{1.3}$$

1.3.4. F-1 Puanı

Duyarlılık ve kesinlik metriklerinin harmonik ortalaması ile bulunur [40].

$$F1-\text{puanı} = 2 * \text{Hassasiyet} * \text{Duyarlılık} / (\text{Hassasiyet} + \text{duyarlılık}) \tag{1.4}$$

1.3.5. AUC-ROC

İkili sınıflandırma modellerinin başarımını değerlendirmek için kullanılan AUC (Eğri Altındaki Alan) metriği, ROC (Alıcı İşletim Karakteristiği) eğrisinin altındaki alanı hesaplayarak, modelin farklı eşik değerleri altında doğru pozitif oranı ile yanlış pozitif oranı arasındaki dengeyi ortaya koyar [25]. Modellerin örnekleri, pozitif ve negatif olarak ayırma becerisini değerlendirir. Hatalı ve hatalı olmayan varlıkların güvenli bir şekilde ayırt edilmesini sağlar. SDP bulgularında yüksek doğruluk, güvenilirlik ve yorumlanabilirlik değerlerini incelemeye sunar [12].

Batool ve Khan [17], SDP modellerinde en sık kullanılan performans ölçütlerinin doğruluk ve kesinlik olduğunu ifade ettiği görülmektedir [15].

1.4. Açıklanabilir Yapay Zekâ (XAI)

XAI, makine öğrenmesi ve yapay zekâ modellerinin karar alma süreçlerinin iyileştirebilmeyi ve süreçlerin anlaşılabilirliğini artırmayı hedefleyen bir yaklaşımdır. XAI'nin temel amacı ise, özellikle yüksek doğruluk sağlayan ancak yorumlanabilirliği düşük olan gelişmiş tahmin modellerinin "kara kutu" doğasını ortadan kaldırarak, bu sistemleri daha şeffaf, izlenebilir ve açıklanabilir hale getirmektir [9], [41]. Kara kutu niteliği taşıyan yapay zekâ tabanlı modellerin, karar alma süreçlerinin şeffaf olmaması ve yorumlanabilirliğin yetersiz olması, önyargı ve taraflılık riskini artırarak, bu sistemlerin etik uygunluğu ve güvenilirliği konusunda ciddi endişelere yol açmaktadır [9].

Buna karşılık beyaz kutu yapay zekâ tabanlı sistemlerin karar alma süreçleri, geliştirici ve kullanıcıların inceleyebileceği ve yorumlayabileceği şekilde şeffaf kurallara ve

mantığa dayanır. Beyaz kutu yapay zekâsı olarak bilinen XAI, geliştirici ve kullanıcıların tahmin veya kararı sonucunu hangi gerekçelerle ürettiğini anlamalarını sağlayarak, sistemin güvenilirliğini artırır ve adil, etik karar verme yetisini destekler. Beyaz kutu AI/ML modellerinin, kara kutu AI/ML modellerine kıyasla daha genel açıklamalar ürettiği ve bunların daha az doğru olduğu bilinmektedir [9]. Doğruluk ve güvenilirlik SDP sürecinde, önemli parametrelerdendir.

Yapay zekâ yöntemlerinin kararlarının açıklanması kullanılan modellere bağlıdır. Bazı modeller doğası gereği açıklanabilir çıktılar sunarken, derin öğrenme tabanlı modeller karmaşık mimarileri nedeniyle karar verme süreçleri açısından şeffaf değildir ve yorumlanabilmeleri için açıklanabilir yapay zekâ yaklaşımlarına gereksinim duyar. [38], [42]. Derin öğrenme modelleri yazılım hata tahmininde diğer modellere nazaran daha iyi performans gösterseler de genellikle kara kutu modelleri olmaları tahminleri açıklayamamalarına sebep olmaktadır. Bu durum, güvenlik açısından kritik yazılımlarda derin öğrenme modellerinin kullanımının yaygınlaşmasının önünde engel teşkil etmektedir. XAI, hata tahmininde ulaşılan sonuçların insanlar için anlaşılabilir gerekçelerle açıklanmasını sağlayarak bu sorunun giderilmesini amaçlamaktadır [43].

Modellere açıklanabilirlik kazandırmak için yapılan kısıtlamalar, tahmin doğruluğunda azalmaya yol açabilir. Daha yorumlanabilir modeller genellikle daha karmaşık modellere göre daha düşük doğruluk sağlar [42]. Doğruluğu daha yüksek olan modellerin daha karmaşık olduğunun ve bu nedenle açıklanabilirliği ve yorumlanması zorlaşmaktadır [9].

1.4.1. Açıklanabilirlik Neden Gereklidir?

Son araştırmalara göre, şeffaf olmayan karar mekanizmaları ve pratik içgörü yetersizliği geliştirici ve kullanıcıların SDP modellerinin çıktılarını onaylamakta tereddüt etmelerine neden olmaktadır. Bu durum, açıklanabilirlik ve uygulanabilirlik gereksinimlerini karşılamak amacıyla yeni bir araştırma alanı doğurmuştur [44]. Kamuoyu ve akademik araştırmacılar özellikle iç işleyişi şeffaf olmayan kara kutu yapay zekâ ve makine öğrenimi modellerini, anlaşılır kılabilmek amacıyla açıklanabilirlik konusuna ilgileri yoğun bir şekilde artmaktadır [9].

Geliştiriciler, uygulama alanları ile ilgili ön bilgilere sahiptir. Bu bilgilerin, arkasında yatan gerekçeyi anlayıp yorumladıkları halde tahmin bulgularını, katılmak veya karşı

çıkmak için kullanabilirler [16]. Yorumlanabilirlik, son kullanıcı ve geliştiricilerin modelin karar verme sürecine ilişkin açık ve anlaşılabilir içgörüler elde etmesine olanak tanır. Bir bulgunun yorumlanabilir olmaması, geliştiricilerin SDP tahminlerinin altında yatan mantığı kavramasını zorlaştırıp, modellerin güvenilirliğine yönelik kuşkuların artmasına ve ilgili yöntemlerin benimsenmesinde isteksizlik yaşanmasına neden olur [25].

Açıklanabilir yapay zekâ yöntemleri, test ve hata tahmini süreçlerinde uzun vakitler harcayan uygulayıcıların bu tahminleri anlamalarına ve yorumlamalarına yardımcı olur. Yöntemlerin açıklamalarının doğru ve güvenilir olmasının yanında uygulayıcıların anlayacağı şekilde uygulayıcıya sunması gerekir. Bu şartlarda uygulayıcılar makine öğrenmesini ve açıklanabilir yapay zekâ yöntemlerini güvenerek ve aktif bir şekilde kullanabilirler [16].

XAI yöntemleri, SDP modellerine sadece yorumlanabilirlik kazandırmaz, bunun yanında yazılım geliştirme sürecinde geliştiricilere, kararlarını yönlendirebilecek anlamlı ve uygulanabilir bilgiler sunar [9].

Açıklanabilir yapay zekânın başlıca problemlerinden biri yapılan açıklamanın güvenilirliğine duyulan eksikliktir [45]. Bu problemin çözümünde, açıklamaları karşılaştırabilmek amacıyla birden fazla açıklama yöntemi kullanılabilir [46]. Ancak, farklı yöntemlerin açıklamaları ve bulguları aynı olmak zorunda değildir [45]. Hatta, aynı modelin birden fazla yürütülmesiyle elde edilen sonuçlar ve açıklamaları özdeşleşmeyebilir [25].

Makine öğrenimi modeli yüz tanıma, görüntü sınıflandırma, doğal dil işleme vb. gibi birçok alanda olağanüstü düzeyde bir performansa sahip olsa da bu performans doğrusal olmayan bir modele ve parametre ayarlama teknolojisine daha çok bağımlıdır. Makine öğrenimi modellerinin verilerden ne öğrendiğini ve nihai kararlarını nasıl aldığını anlamanın bir yolu yoktur. Bu "uçtan uca" karar alma modeli, son derece açıklayıcı olmayan bir makine öğrenimi modeliyle sonuçlanır. İnsan bakış açısından, modelin karar alma süreci anlaşılmazdır. Makine öğrenmesi modelinin açıklanamazlığı birçok potansiyel tehlikeye sahiptir ve insanlar ile makineler arasında güven oluşturmak kolay değildir. Açıklanamayan bir model daha güvenilir bilgi sağlayamayacağı için, birçok alanda gerçek dağıtımı minimum düzeyde olacaktır [6].

SDP'de XAI yöntemlerini kullanmak doğru tahminlere ulaşmamızın yanında tahmini etkileyen özellikler hakkında fikir sahibi olmamızı sağlar [12]. XAI'nin faydalarını farklı kullanıcıların perspektifinden bütüncül bir şekilde ele almak, XAI'nin nasıl etkin şekilde kullanılabileceğine ilişkin kapsamlı bir bakış açısı kazandıracaktır [38].

1.4.2. Açıklanabilir Yapay Zekâ Yöntemleri

Yazılım hata tahmininin açıklanabilirliği genelde küresel açıklanabilirlik ve yerel açıklanabilirlik olmak üzere iki grupta incelenir [25]. Küresel düzeyde açıklama sağlayan yöntemler, hata tahmin modellerinin tüm veri set üzerindeki karar süreçlerini analiz eder ve modelin genel işleyişine dair kapsamlı çıkarımlar sağlayarak bütüncül içgörüler sunar [47]. Yerel düzeyde açıklama sağlayan yöntemler ise belirli bir örnek üzerine odaklanarak tahminin neden hatalı veya hatasız olduğuna dair bulgular sunar. Yerel düzeyde yapılan açıklamalar, belirli örnekler için hataların altında yatan ana sebepleri ortaya çıkarmaya odaklanır ve geliştiricilerin bu sebepleri anlamalarına, buna bağlı riskleri azaltmalarına olanak sağlar [48]. Küresel boyutta etki değeri yüksek olan özellikler yerelde etki değeri yüksek olmayabilir, aynı şekilde yerelde katkı değeri yüksek özellikler küresel de katkısı düşük çıkabilir [16].

1.4.2.1. Shapley Eklemeli Açıklamalar (SHAP)

Oyun teorisine dayalı açıklamalar yapan bir açıklanabilir yapay zekâ yöntemidir [43]. Bu yöntem, her bir özelliğin model tahminine katkısını, tüm olası özellik kombinasyonları küresel ve yerel açıklamalar sağlar [41]. Gerek ağaç tabanlı modellerde (örneğin rastgele ormanlar, gradyan güclendirme) gerekse sinir ağlarında etkili biçimde uygulanabilir. Gerçek zamanlı uygulamalarda kullanımı sınırlı olup, karmaşık ve büyük veri setleri yoğun hesaplama gerektirdiğinden SHAP yöntemi zaman ve kaynak açısından yüksek maliyetli olabilir [9].

1.4.2.2. Yerel Yorumlanabilir Model-Agnostik Açıklanabilirlik (LIME)

Tahmin çerçevesinde herhangi bir örneğin yorumlanmasını sağlayan XAI yöntemidir [16]. Bir hatanın tahminine etki eden en belirleyici özellikleri ortaya çıkarmak için en yaygın yöntemdir [28]. Ancak LIME yöntemi, özelliklerin birbiri ile olan ilişkilerinden bağımsız olarak tek özelliğin tahmine olan etkisini araştırır, elde edilen bulgular ve açıklamalar tahmin sonuçlarının genel açıklamasını yapamaz ve genel açıklama için

işlevsel değildir [49] ve tek örneğe yönelik açıklama üretmesi sebebiyle genelleştirilme imkânı bulunmamaktadır [25]. Bu yöntem, açıklaması yapılacak olan örneği belirler, orijinal veriye uygulanan kontrollü rastgele bozulmalarla sentetik örnekler oluşturulur, bu sentetik örnekleri tahmin eder ve bu örneklere dair yerel düzeyde oluşturulan açıklanabilir modellerle, kara kutu modellerinin çıktılarının nedenlerini açıklamayı amaçlar. Ancak, modelden bağımsız yöntemler SDP model tahminlerinin ardındaki gerçek mantığı yakalayamaz, temel açıklamaları sunmak yerine çıkarıma dayalı sonuçlar üretebilirler [22], [44].

LIME yöntemi, hata tahmini konusunda umut verici sonuçlara sahiptir [28]. Bu yöntemin temel amacı, modelin tahmin ettiği sonuçların yerel düzeyde anlaşılabilirliğini sağlamak için bu öngörülen bulgulara yakın yorumlanabilir bir model eğitmektir. Açıklanabilir SDP çalışmalarında kullanım kolaylığı nedeniyle [6] sıklıkla kullanılmaktadır. Buna rağmen bazı sınırlamaları bulunur. Örneğin, raslantısal bozulmalarla oluşturulan sentetik örnekler, açıklaması yapılacak örneklere benzer olmayabilir. LIME'ın sunduğu yerel açıklamalar, sentetik olarak oluşturulan örneklere yüksek derecede bağımlılık göstermektedir [50], ek olarak yapa zekâ, ihtiyaçları yanlış veya eksik yorumlayabilir ya da önemli kısımları göz ardı edebilir, ayrıca belirsiz veya eksik ifadeleri yanlış yorumlayabilir [41], [51]. Bu durumlar, açıklamaların güvenilirliğini ve açıklama gücünü sınırlayabilir [49]. Buna karşın RF, GB ve MLP gibi modeller üzerinden belirlenen bir örneğin hata tahmini için elverişlidir [9].

1.5. Literatür Taraması

Yazılım hata tahmini üzerine yürütülen çalışmaların kökeni 1970'li yıllara uzanmakta olup, bu alandaki araştırmalar günümüzde de sürdürülmektedir [51]. Mevcut yazılım hata tahmini çalışmalarındaki temel sınırlılık, kullanılan modellerin şeffaflık ve açıklanabilirlikten yoksun olmasıdır [50]. Son dönemlerde yapılan çalışmalar, geliştirilen tahminlerin kullanıcılar tarafından anlaşılabilir olmasını sağlamak amacıyla yazılım hata tahmininin açıklanabilirliğine odaklanmıştır. Gezici ve Tarhan Gradyan Güçlendirme (Gradient Boosting, GB) makine öğrenmesi algoritması üzerinde üç farklı modelden bağımsız açıklanabilir yapay zekâ yöntemini (Explain Like I'm 5 (ELI5), SHAP ve LIME) yazılım hata tahmininin açıklanabilirliğini sağlamak amacıyla kullanmışlardır. Çalışmalarında, küresel düzeyde ELI5 yönteminin, yerel düzeyde ise LIME yönteminin daha iyi performans sergilediği rapor edilmiştir [27]. Yazılım hata tahmininde

anlaşılabilirlik sağlamak amacıyla SHAP yönteminin kullanıldığı bir diğer çalışmada, özellik çıkarımı ve ön işlem adımlarından geçirilmiş ISBSG veri seti üzerinde Rastgele Orman (Random Forest, RF), Uyarlanabilir Güçlendirme (Adaptive Boosting, AdaBoost), GB, Ekstra Ağaçlar (Extraa Trees, ET), Aşırı Gradyan Güçlendirme (Extreme Gradient Boosting, XGBoost) ve Kategorik Güçlendirme (Category Boosting, CatBoost) algoritmalarının sonuçları açıklanmıştır. Çalışmada GB ve XGBoost modelleri Ortalama Mutlak Hata (Mean Absolute Error, MAE), Ortalama Kare Hata (Mean Squared Error, MSE) ve Karekök Ortalama Kare Hata (Root Mean Square Error, RMSE) ölçütlerinde yüksek performans sergilemiş olsada, çapraz doğrulama ve test verilerinde aynı başarı elde edilememiştir. Bu uyumsuzluk, eğitim sürecinde aşırı öğrenme olasılığını gündeme getirmiştir. Sonuçlar, ET, RF ve CatBoost algoritmalarının SDP' de daha iyi performans sunduğunu ortaya koymuştur [50].

Jiarpakdee vd., çalışmalarında LIME dışındaki modelden bağımsız yöntemlerle elde edilen örnek açıklamaların, modellerin genel açıklamalarıyla benzer sonuçlar verdiğini rapor etmiştir. Ayrıca, sınıflandırma yöntemlerinden bağımsız olarak Tahminin Bileşenlerine Ayrıştırılması Yöntemi (Break Down Explanation of Model Prediction, BreakDown) ve LIME ile Hiper Parametre Optimizasyonu (LIME with Hyper Parameter Optimisation, LIME-HPO) yöntemlerinin belirli bir örnek için aynı açıklamaları ürettiğini ifade etmişlerdir [4]. Benzer şekilde, Shin vd. yazılım hata tahmini modelleri üzerinde LIME-HPO ve BreakDown yöntemlerini uygulamış; makine öğrenmesi modellerinden Lojistik Regresyon (Logistic Regression, LR), Karar Ağacı (Decision Tree, DT) ve RF algoritmalarını kullanarak veri dengesizliği sorununu gidermek amacıyla örnekleme tekniklerinden yararlanmışlardır. Çalışmalarının sonucunda, LIME-HPO ve BreakDown yöntemleri ile elde edilen açıklamaların ayar farklılıklarına bağlı olarak önemli ölçüde değişiklik gösterdiğini ve tutarsız sonuçlar ortaya çıkardığını, dolayısıyla bu yöntemlerin kullanımının güvenilir olmadığını ifade etmişlerdir [52].

Begum vd., çalışmalarında, NASA veri setlerine Sentetik Azınlık Aşırı Örnekleme Tekniği (Synthetic Minority Over-sampling Technique, SMOTE) ve etiket kodlama tekniklerini uygulamış ve on üç farklı makine öğrenmesi modeliyle yazılım hata tahminine yönelik bir değerlendirme gerçekleştirmişlerdir. Makine öğrenmesi modelleri doğruluk, MSE ve R^2 değerleri açısından karşılaştırıldığında, Aşırı Gradyan Güçlendirici Regresör (Extreme Gradient Boosting Regressor, XGBR) algoritmasının daha yüksek

performans sergilediği belirlenmiştir. Ayrıca yazılım hata tahmininde açıklanabilirliği elde edebilmek amacıyla LIME ve SHAP yöntemleri kullanılmış; her iki yöntemin de farklı kullanım bağlamlarına ve önceliklendirdikleri durumlara göre özelliklerin tahmin üzerindeki etkisini ve bu etkinin büyüklüğünü yorumlayarak sonuçları daha anlaşılır hale getirdiği ortaya konulmuştur. Araştırmanın temel sınırlılığı yazılım tahmininde kullanılan özelliklerin sınırlı sayıda olmasıdır [53]. LIME ve SHAP kullanılan başka bir çalışmada, LIME ve SHAP'in kullanıldığı açıklamaları elle analiz edip hata problemlerini öngörmek ve gelecek çalışmalara kaynak oluşturmak hedeflenmiştir [54]. LIME ve SHAP yöntemlerinin KC2 veri seti üzerinden uygulandığı bir diğer çalışmada, tahmin süreci şeffaflaştırılmış, böylelikle geliştiricilerin tahmin analizlerini yorumlayabilmeleri sağlanmıştır [18], [53]. Gezici Geçer ve Tarhan çalışmalarında KC2 veri seti ile SDP ve bu tahminin hem yerel hem küresel açıklanabilirliğini artırmak amacıyla RF, GB, Naive Bayes Sınıflandırıcısı (Naive Bayes Classifier, NBC), MLP ve Sinir Ağı (Neural Network, NN) makine öğrenimi modellerini ve SHAP, LIME, ELI5, Kısmi Bağımlılık Grafiği (Partial Dependence Plot, PDP) ve Hızlı Yorumlanabilir Prototip Seçimi (Fast Interpretable Prototype Selection, ProtoDash) açıklanabilir yapay zekâ yöntemlerini kullanmışlardır [9]. Roy vd. çalışmalarında, LR, RF, GB, MLP modellerini ve XAI yöntemlerinden SHAP ve LIME kullanarak çalışmalarına açıklanabilirlik eklemişlerdir [45]. GB ve DT'yi içeren dört farklı modelin %90'ın üstünde doğruluk elde eden Al-Smadi vd., veri setlerindeki özelliklerin en etkili olanını, nasıl etkilediğini ve etki değerini anlamlandırabilmek amacıyla SHAP yöntemini kullanmışlardır [55]. Bommi ve Negi farklı makine öğrenmesi modellerini kullanarak sınıflandırma tekniklerinin başarısını ve dengesiz veri setlerinin sınıflandırma üzerindeki etkisi üzerine çalışmışlardır. Destek Vektör Makineleri (Support Vector Machines, SVM) algoritmasının sınıflandırma modelinin uygulanabilirliğini artırabileceği ifade etmişlerdir. XAI yöntemlerinin sınıflandırma modellerinde şeffaflık ve yorumlanabilirliği geliştirdiğini, dolayısıyla kullanıcıların model çıktısına duyduğu güveni artırabileceğini belirtmişlerdir [2].

Lee ve Lee çalışmalarında, RF, GB, XGBT, MLP modellerinin performanslarını karşılaştırmış daha sonra LIME, SHAP ve BreakDown XAI yöntemlerine göre her bir özelliğin etkisini modelleri kullanarak hesaplamışlardır. Devamında, tekrarlanan deneyler aracılığıyla yerel açıklamalardan elde edilen tüm bulgulardaki özelliklerin katkı değerlerini; dağılım, sıralama ve etki yönü açısından kapsamlı şekilde analiz

yapmışlardır. Sonuç olarak, farklı modelden bağımsız yöntemlerin, geliştiricilere hem katkı düzeyi, sıralaması hem de yönü bakımından tutarsızlık gösteren yerel açıklamalar sunduğunu belirtmişlerdir [56]. Bir diğer çalışmada Lee vd. SDP'nin açıklanabilirliğinin öneminden bahsetmiş olup, SDP için SVM, RF ve XGBoost modelleri ve LIME, LIME-HPO, TimeLIME ve SQAPlanner açıklanabilir yapay zekâ yöntemleri ile çalışmışlardır. LIME yöntemi tüm özellikleri içeren bulgular verdiği için, bulgulardaki analizlere göre en iyi performans gösteren 5 özelliği belirlemişler ve eyleme geçirilebilir planlar simüle etmiş ve sonuçlarını değerlendirmişlerdir. Özellik sayısının kısıtlanması deneysel bulgular üzerinde etkili olsa da yapılan analizler özellik sayısının değiştirilmesinin bulgular üzerinde kayda değer bir fark oluşturmadığı ortaya koymuşlardır [44].

Tantithamthavorn ve diğerleri, beş ayrı veri kümesinden oluşturulan açık kaynaklı verilerle beş farklı sınıf yeniden dengeleme yöntemi ve yedi sınıflandırma yöntemi kullanarak sınıf yeniden dengeleme yöntemlerinin ve hata tahmin modellerinin performansı ve yorumlanması üzerine çalışmışlardır. Elde edilen bulgular, hata tahmin modellerinin yorumları üzerinde kullanılan sınıflandırma yöntemlerinin etkisinin fazla olduğunu ve bu nedenle geliştirici ve kullanıcıların hata tahmin modellerinden açıklama elde ederken sınıf yeniden dengeleme yöntemlerini kullanmamaları gerektiğini göstermektedir [57].

Shin vd., toplam dokuz açık kaynak yazılım projesinden elde edilen 32 adet SDP veri setinin LIME ve BreakDown yöntemleri ile açıklanabilirlik kazandırdıkları çalışmalarında, çeşitli veri örnekleme yöntemlerin kullanılması ile çıkan sonuçlara göre özelliklerin sıralarının büyük oranda değiştiğini gözlemlemişlerdir [58]. Kod bloğu, çok sayıda çalıştırıldığında elde edilen yorumlar aynı olmayabilir. Xu vd. ise çalışmalarında, tahmin metriklerini yükseltmek amacıyla derin öğrenme modellerinden Derin Piramit Evrişimli Sinir Ağlarına (Deep Pyramid Convolutional Neural Network, DP-CNN) bir dikkat tabakası ekleyerek kullanmış ve yeni bir çerçeve önermişlerdir. Önerilen çerçevenin, karar alma işlemi sırasında yapılan Bilgi Damıtma yöntemi ile tahmini doğru bir şekilde açıklayıcıya aktarması, LIME ve BreakDown yöntemlerinden daha güvenilir ve tutarlı hale getirdiğini ve küresel açıklanabilirlik sunduğunu söylemişlerdir. DP-CNN; SVM, RF, DBN, AutoSpearman (Automatically mitigating correlated software metrics for interpreting defect models) modellerinden daha yüksek performans göstermiştir [25].

Ribeiro vd., birkaç örnek ile yerel tahmin bulgularının, küresel tahmin elde etmenin bir yolu olduğunu ve LIME yönteminin güvenilir olduğunu söylemişlerdir. Bireysel hata tahmin açıklamalarının güven değerlendirmesinde etkili olduğu savunulmuştur. Çalışmada, kullanıcıya modelin bütüncül bir görünümünü sağlayan ve yalnızca temsili, gereksiz olmayan tahminleri seçmeye imkân sağlayan SP-LIME yöntemi tanıtılmıştır [16].

Gottumukkala vd., dönüştürücü tabanlı modelleme yöntemi olan BERTopic ile çok çıkışlı sınıflandırıcıyı bütünleştirip, hata ve hataya sebep olan kök nedenleri tahmin eden bir yöntem ortaya koymuşlardır. Çok çıktılı sınıflandırıcıyı eğitirken LR, RF, K-En Yakın Komşu (K-Nearest Neighbors, KNN), DT sınıflandırıcılarını kullanmışlardır [59]. Bir diğer açıklanabilir SDP yaklaşımı sunan çalışmada, yerel açıklamalar sunan modelden bağımsız CfExplainer'ı önermişlerdir. CfExplainer'ın, hızlı ve yinelemeli yazılım geliştirme süreçlerine uyumlu olan tam zamanında hata tahmininde (Just-In-Time, JIT) uygulandığını, oluşturulan açıklamaların hata tahmin modellerinin şeffaflığını artırabileceğini, böylelikle geliştiricilerin hataların altında yatan sebepleri anlamalarına yardımcı olabileceğini ve bu doğrultuda, hata riskini azaltacak düzeltici önlemleri belirlemeleri için yol gösterici olabileceğini söylemişlerdir [49]. JIT teknolojisi konusunda yapılan bir başka çalışmada, tahmin performansını artırabilmek için 6 açık kaynak kodlu proje kullanılarak RF sınıflandırıcısı yardımıyla bir tam zamanlı hata tahmin modeli geliştirmiş ve eğitmişlerdir. Açıklanabilirlik için LIME yöntemini kullanarak SP-LIME, açıklanabilirlik aracını oluşturmuşlardır [6]. Pornprasit ve diğerleri, JIT açıklayabilmek için PyExplainer yöntemini önermişlerdir. PyExplainer, sınıflandırma probleminin önsezilerini açıklayabilmek amacıyla hazırlanmıştır, özellikler arasındaki etkileşimleri dikkate alan bir açıklama sağlar [60]. Literatür taramasına dair özet bilgiler Tablo 1.1'de verilmiştir.

Yazılım hata tahmini için derin öğrenme modelleri ile çalışmaların, makine öğrenmesi modelleri ile olan çalışmalara nazaran çok az sayıda olduğu gözlemlenmiştir. Bu çalışmada derin öğrenme modellerine de yer verilerek SHAP ve LIME modellerinin her ikisi de incelenmiştir. Yazılım hata tahmini sırasında sınıf dengesizliği problemi, veri dengesizliği problemi ve maliyete duyarlılık problemleriyle karşılaşılmaktadır. Sınıf dengesizliğini ortadan kaldırmak amacıyla aşırı örnekleme ve veri ölçekleme işlemleri de uygulanmıştır.

Tablo 1.1. Literatür Özeti

Yazarlar / Yıl	**Amaç / Problem**	**Yöntem**	**Veri Seti**	**Sonuçlar**
Gezici ve Tarhan (2022) [27]	XAI yöntemlerinin hata tahminine katkılarını belirlemektir.	GB + SHAP, LIME, ELI5	NASA – KC2	En iyi doğruluğu %84 ile GB sınıflandırıcısı vermiştir.
Mustaqeem vd. (2024) [14]	SDP'de şeffaflık ve güvenilirlik için XAI yöntemlerine entegre, hibrit bir model olan SPAM-XAI önerilmektedir.	SMOTE + PCA + MLP SHAP, LIME	NASA-CM1, PC1, PC2	CM1; Doğruluk %98,13 PC1; Doğruluk %96 PC2; Doğruluk %98,65
Asal ve Özdeş Demir (2024) [18]	XAI yöntemlerini entegre ederek yazılım hata tahmininde yapay zekâ uygulamalarını şeffaf kılabilmektir.	LR – RO – NBC SHAP, LIME	NASA KC2	SHAP ve LIME entegrasyonu açıklanabilirliği artırmış; model, temel yöntemlerle benzer doğruluk düzeyine sahiptir.
Haldar ve Capretz (2023) [50]	Şirketler arası veriler kullanılarak, SDP modelleri geliştirmeye odaklanmaktadır.	RF, Adaboost, GB, ET, XGBoost, Catboost - SHAP	ISBSG Developments & Enhancements 2021 Release 2 veri kümesi	ET modeli bu çalışmada kullanılan diğer modeller arasında en verimli model olarak kabul edilmektedir. R^2: %89
Bommi ve Negi (2023) [2]	SDP performansına, modellerin güvenilirliğini artırmak için ise XAI yöntemlerinin uygulanmasına odaklanmaktadır.	LR, NBC, KNN, SVM ve RF	NASA PROMISE JM1	%82'lik zirve doğruluk ve azaltılmış eğitim süresi

Yazarlar / Yıl	Amaç / Problem	Yöntem	Veri Seti	Sonuçlar
Begum vd. (2023) [53]	Hata tahmin işleminde en verimli model seçilip, çalışmaya açıklanabilirlik eklenmektedir.	RF, LR, KNN, AdaBoost, GB, XGBR, XGBoost, ET ve SVM SHAP, LIME	NASA PROMISE PC1, CM1, JM1	XGBR modelinin değerlendirilen tüm kriterlerde üstün performans gösterdiği ve yazılım hata tahmininde, LIME yönteminin açıklama doğruluğu açısından daha üstün performans sergilediği belirtilmiştir.
Roy vd. (2022) [45]	XAI yöntemleri arasındaki tutarsızlıkları ölçmektir.	LR, RF, GBT ve MLP modellerinin LIME ve SHAP	Java projelerinin çeşitli sürümlerini içeren on hata tahmin veri seti	RF algoritması AUC performans ölçümlerinde %94' e kadar çıkmıştır.
Shin vd (2023) [52]	SDP modelleri üzerinde LIME-HPO ve BreakDown yöntemlerinin güvenilirliğini ve kararlılığını araştırmışlardır.	Veri Örnekleme Teknikleri, RENN (Tekrarlanan Düzenlenmiş En Yakın Komşular), LR, DT, RF, Ortalama Sinir Ağı (Neural Networks Using Model Averaging, avNNet), GBM, xGBTree LIME-HPO, BreakDown	9 açık kaynaklı yazılım sisteminden seçilen ve düzenlenen 32 veri seti.	LIME-HPO tekniğinin, bu projelerdeki ortalama isabet oranı 0,515 (DT), 0,613 (AVNNet) aralığındadır. BreakDown tekniğinin, bu projelerdeki ortalama isabet oranı 0,609 (DT) – 0,681 (AVNNet) aralığındadır. -PyExplainer tekniğinin, RF sınıflandırıcısı için isabet oranı: 0,776'dır.

Yazarlar / Yıl	Amaç / Problem	Yöntem	Veri Seti	Sonuçlar
Jiarpakdee vd. (2020) [4]	Mevcutta bulunan hata tahmin modellerinin, yorumlanabilirlik ve açıklanabilirlik konusunda hala eksik olduğu belirtilmiştir.	LIME BreakDown LIME-HPO	9 açık kaynaklı yazılım sistemini kapsayan 32 sürümden oluşmaktadır	LIME-HPO'nun kullanılan modellerde tahmin performansını LR için %6,6, RF için %7,4, C5.0 için %6,7, AVNNet için %6,1, GBM için %8 ve xGBTree için %7,6 oranında modeli temsil etme doğruluğunu artırmıştır.
Lee ve Lee (2023) [56]	Modelden bağımsız yöntemlerle SDP için yerel açıklamalardan elde edilen özelliklerin etki değerleri, etki sıralamaları ve etki yönleri bulguları karşılaştırılmıştır.	RF, GBM, MLP ve XGBT LIME, SHAP ve BreakDown	9 yazılım sisteminin 32 sürümü	Modelden bağımsız yöntemleri, özelliklerin katkı değerlerinin ve yönlerinin birbirinden farklı olduğu yerel açıklamalar sunmuştur.
Tantithamthavorn vd. (2020) [57]	Sınıf yeniden dengeleme yöntemlerinin SDP modellerinin açıklanabilirliğinde ve performans üzerindeki etkisi incelenmiştir.	Alt Örnekleme, Aşırı Örnekleme, SMOTE, Boostrap Rastgele Aşırı Örnekleme Yöntemleri	101 açık kaynak yazılım projesine ait veriler	Hata tahmin modelleri, optimize edilmiş SMOTE tekniği uygulandığında en büyük AUC iyileştirmesini, alt örnekleme tekniği uygulandığında ise en büyük Duyarlılık iyileştirmesini sağlar.
Schulte vd. (2024) [54]	Açıklamaların kalitesini, insan beklentisiyle uyumunu ve makine öğrenmesi modellerinin anlaşılmasını sağlayıp sağlamadığı araştırılmıştır.	LIME ve SHAP	38 açık kaynaklı projeden 11.154 elle doğrulanmış hata içeren veri set	Her iki XAI yönteminde yüksek kalitede olduğu, fakat SHAP yönteminin LIME'dan daha performanslı çalıştığı ortaya koyulmuştur.

Yazarlar / Yıl	Amaç / Problem	Yöntem	Veri Seti	Sonuçlar
Xu vd. (2025) [25]	DP-CNN yöntemi önerilmiştir. DP-CNN'in; LIME ve BreakDown yöntemlerinden daha güvenilir ve tutarlı açıklanabilirlik sunduğu söylenmiştir.	SVM, RF, DBN, AutoSpearman,LIME, BreakDown,DP-CNN	NASA PROMISE PC2	DP-CNN F-Skor için: SVM'yi %42,86, RF'yi %30,63, DBN'i %61,75, AutoSpearman'ı %2,13 iyileştirmiştir. AUC için: SVM'yi %10,93, RF'yi %15, DBN'i %20,75, AutoSpearman 'ı %0,97 iyileştirmiştir.
Ribeiro vd. (2016) [16]	LIME önerilmiştir. Tekrarsız tahminleri seçmek ve modeli özetlemek için SP-LIME tanıtılmıştır.	SVM, LR, DT, NN LIME, SP-LIME	Sınıflandırmak için iki duygu (olumlu, olumsuz) analizi veri seti (her biri 2000 örnek içeren kitaplar ve DVD'ler) kullanmışlardır.	Güvenilirlik ölçütü olarak sadece doğruluğun kullanılması durumunda performansı kötü olan sınıflandırıcının seçilebileceği belirtilmiştir.

2. BÖLÜM

YÖNTEM VE MATERYAL

2.1. Veri Setleri

Çalışmamızda yazılım hata tahmini çalışmalarında sıklıkla kullanılan NASA PROMISE veri havuzunda yer alan CM1 ve PC1 [25] veri setleri kullanılmıştır. Bu veri setler, SDP çalışmalarında standart bir karşılaştırma zemini sunup, literatür geçmişine sahip olmaları sebebiyle önceki raporlanan sonuçlarla doğrudan performans karşılaştırmasına olanak tanımaktadır [61]. CM1 veri setinin, dengesiz sınıf dağılımı ve nispeten küçük örneklem hacmi, modellerin dengesiz sınıf yapıları altındaki dayanıklılığını değerlendirmeye elverişli bir deneysel çerçeve sunmaktadır. PC1 veri seti ise, nispeten daha dengeli sınıf dağılımı ve daha büyük veri hacmi ile modellerin, genelleme yeteneğinin ölçülmesine yönelik uygun bir metodolojik çerçeve sağlamaktadır [14].

Her iki veri setinin de Halstead karmaşıklığı, McCabe döngüsellik ölçütü, kod hacmi ve yapısal karmaşıklık gibi temel yazılım mühendisliği özelliklerini sistemli biçimde içermesi, modellerin hem performans hem de açıklanabilirlik yönünden bütüncül olarak değerlendirilebilmesine imkân sağlamaktadır [14], [62].

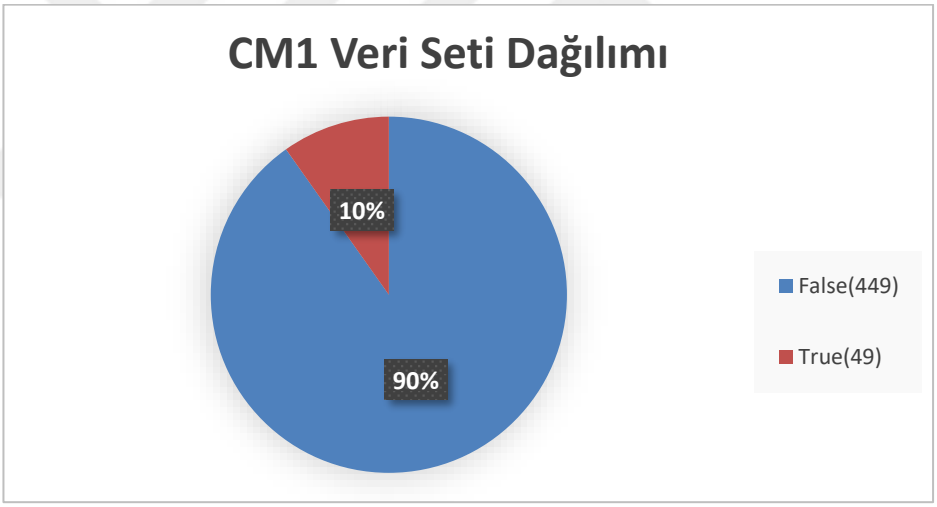
CM1 ve PC1 veri setlerinin temizlenmiş, kapsamlı biçimde belgelenmiş ve araştırma topluluğu tarafından uzun süredir doğrulanmış olmaları, veri hazırlama sürecindeki ek iş yükünü önemli ölçüde azaltmakta; bu nedenle araştırmacıların dikkatini çekerek literatürde yaygın biçimde tercih edilmelerine katkı sağlamaktadır [61]. CM1 veri seti, geniş bir öznitelik çeşitliliği sunması, hata durumlarının açık biçimde etiketlenmiş olması ve model çıktılarının hem yerel (örnek bazlı) hem de küresel (veri seti düzeyi) ölçekte yorumlanmasına elverişli yapısı nedeniyle açıklanabilirlik odaklı analizler için güçlü bir deneysel çerçeve sunmaktadır. PC1 veri seti ise özellik seçimi süreçlerine uygunluğu, görece daha büyük örneklem hacmi, etiketlenmiş hata verilerinin bulunması ve yazılım

niteliklerinin çeşitliliği gibi faktörler nedeniyle literatürde sıkça kullanılan bir diğer referans veri kaynağıdır [12].

Bu iki veri setinin sunduğu yapısal farklılıklar, açıklanabilirlik yöntemlerinin veri bağımlı sonuçlar üretip üretmediğinin ya da daha geniş bağlamlarda genellenebilir bir davranış sergileyip sergilemediğinin test edilmesine olanak sağlamaktadır. Bu nedenle, çalışmamızda CM1 ve PC1 veri setlerinin birlikte kullanılması metodolojik açıdan uygun ve bilimsel olarak gerekçelendirilmiş bir tercih olarak değerlendirilmiştir.

2.1.1. NASA PROMISE CM1

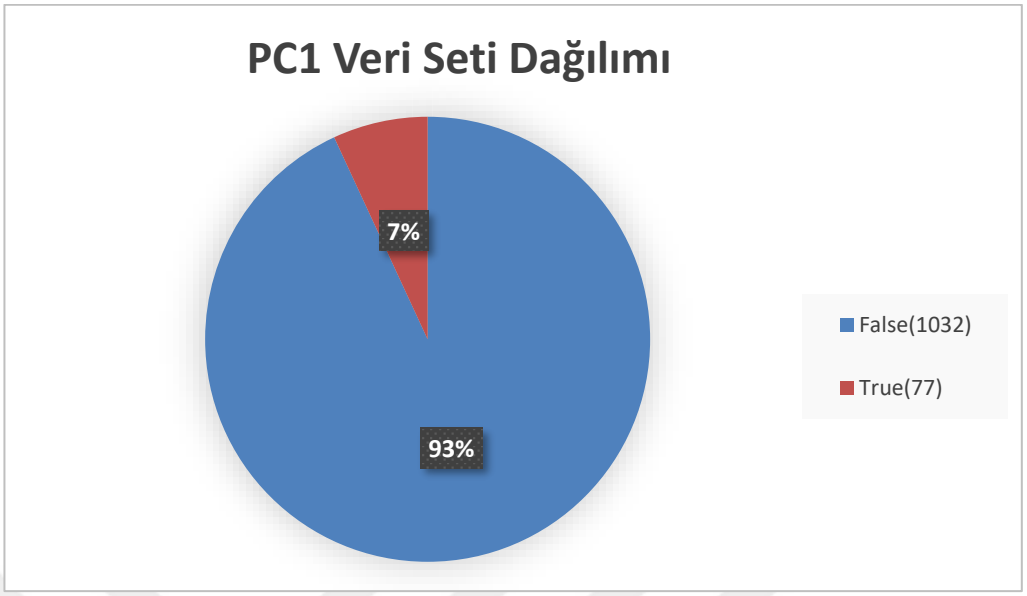
CM1 veri seti 498 satır ve 23 özellikten (Tablo 2.1.) oluşmakta, Şekil 1'de gösterildiği üzere verilerde dengesiz dağılım bulunmaktadır. CM1 veri seti C programlama dilinde yazılmış bir NASA uzay aracı enstrüman verisidir [14], [63].



Şekil 2.1. CM1 Veri seti Dağılımı

2.1.2. NASA PROMISE PC1

PC1 veri seti, C programlama dilinde yazılmış olup Dünya yörüngesindeki uzay aracı uçuş yazılımlarından derlenmiştir [52]. Veri seti 1109 satır ve 23 özellikten (Tablo 1) oluşmakta ve Şekil 2'de gösterildiği üzere verilerde dengesiz dağılım bulunmaktadır [12], [14], [63].



Şekil 2.2. PC1 Veri Seti Dağılımı

Güven Aydın ve Samli, dört farklı veri seti üzerinden yazılım hata tahmini üzerine yoğunlaştıkları çalışmada en yüksek başarı oranlarını PC1 veri setinin verdiğini rapor etmişlerdir [32].

Tablo 2.1. CM1 ve PC1 Veri setlerinin Özelliklerinin Ayrıntıları

Özellik	**Açıklama**	**Veri Tipi**
Loc (McCabe's Line Count Of Code)	Toplam kaynak kod sayısıdır.	Numerik
v (g) (McCabe "Cyclomatic Complexity")	Kodu etkileyen karar noktalarının (if, while vs.) sayısına dayalıdır.	Numerik
Ev (g) (McCabe "Essential Complexity")	Kodu daha doğrusal hale getirebilmek için gerekli minimum karmaşıklık.	Numerik
iv (g) (McCabe "Desing Complexity")	Karar yapılarının kontrol edilme düzeyini ölçer.	Numerik
n (Halstead Operator + Operand)	Operatör ve Operandların toplamı	Numerik
v (Halstead Volume)	Halstead Yoğunluk -> Programın bilgi içeriğini ölçer.	Numerik
l (Halstead Length)	Programın uzunluğunu ölçer.	Numerik
d (Halstead "difficulty")	Algoritmanın karmaşıklığını ölçer.	Numerik
i (Halstead "intelligence" Content)	Bilgi içeriği	Numerik
e (Halstead "effort")	Geliştiricinin kodu yazmak veya anlamak için harcadığı çabadır.	Numerik
b (Halstead "estimated Bugs")	Yazılımın hata olasılığı.	Numerik
T (Halstead time estimator)	Tahmini programlama süresi (sn) t= e / 18	Numerik
IOCode (Halstead's Lines of Code)	Sadece kod satırlarını sayısı.	Numerik
IOComment (Halstead's Lines of Comment)	Sadece yorum satırlarını sayısı.	Numerik
IOBlank (Halstead's count of Blank Lines)	Boş satır sayısı.	Numerik
IOCodeAndComment	Kod ve Yorumların toplamı	Numerik
uniq_Op (Unique Operator)	Farklı operatörlerin sayısı(+, -, if, return vb.)	Numerik
uniq_Opnd (Unique Operand)	Farklı operandların sayısı	Numerik
total_Op (Total Operator)	Operatörlerin toplam kullanım sayısı	Numerik
total_Opnd (Total Operand)	Operandların toplam kullanım sayısı	Numerik
branchCount	Karar noktası sayısı (if, switch, while vb.)	Numerik
defects (Hedef Değişken)	Veri hatalı mı, değil mi?	Kategorik

2.2. Veri Ön İşleme ve Hazırlık

Daha verimli bulgulara erişebilmek amacıyla CM1 ve PC1 veri setleri üzerinde bazı değişiklikler yapılmıştır. Bu yapılan değişiklikler aşağıda sunulmuştur.

- Gerçek ölçümleri içeren temiz ve tutarlı bir analiz için veri setleri üzerinde yorum satırları ve gereksiz başlık satırları kapsam dışı bırakılmıştır.
- Kullanılan veri setlerde özellik isimleri belirsiz ifadeler içerip (ham veri çalıştırıldığında feature 0, feature 1 şeklinde listelenmektedir) hangi sonucun hangi özelliğe ait olduğu anlaşılamamaktadır. Bu nedenle veri seti sütunları isimlendirilmiş olup, veri analizi daha anlaşılır hale getirilmiştir.
- Özellik (bağımsız) ve hedef (bağımlı) değişkenlerinin birbirine karışması veri sızıntısına ve bunun sonucunda sahte yüksek performans elde edilmesine neden olmaktadır. Bu iki değişken grubu birbirinden ayrılmış olup, veri seti makine öğrenmesi modellerinin beklediği yapıya uygun, güvenilir ve genelleme yapılabilir hale getirilmiştir.
- Özellik Ölçekleme işlemi (StandartScaler) yapılarak, her özellik [0, 1] aralığına dönüştürülmüştür. Bu işlem CNN, RNN ve LSTM gibi gradyan tabanlı sinir ağı mimarilerinde gradyan patlaması ya da sönmesi problemlerini azaltır. Bu işlemde temel amacımız, tüm özelliklerin modele eşdeğer ölçekte katkıda bulunmasını sağlayabilmektir.
- Veri seti %80'i eğitimde, %20'si testte kullanılmak üzere rastgele bir şekilde ayrılmıştır. Modelin öğrendiği veri örnekleri ile hiç görmediği veri örneklerini birbirinden ayırarak aşırı uyum riskini ortadan kaldırıp, gerçekçi bir performans değerlendirmesi yapabilmek amaçlanmıştır. Sabit bir rastgelelik değeri kullanılmasıyla, kod her yeniden çalıştırıldığında veri karışımının aynı biçimde oluşturulmasını sağlayarak deneyin tekrarlanabilirliğini güvence altına almıştır.
- CM1 ve PC1 veri setlerinde hatasız modüller çok, hatalı modüller azdır, bu dengesiz veri dağılımı, modelin azınlık sınıfını tanıma kabiliyetini azaltır. SMOTE kullanılarak azınlık sınıfına benzer veriler üretilerek veri dengelenmiştir. Böylelikle model, hatalı ve hatasız sınıfları dengeli öğrenebilmiştir.
- Kullanılan modelin istemine göre girdiler şekillendirilmiş, model oluşturulmuş ve eğitilmiştir.

- Her modelin çalıştırılmasıyla elde edilen tahminler sonucunda Doğruluk, Kesinlik, Duyarlılık, F1-Puanı, AUC-ROC metrikleri hesaplanmış ve karşılaştırılmıştır.
- Hangi modelin, hangi metrik için daha iyi performans gösterdiği belirlenmiştir.
- Hata tahmininde şeffaf olmayan makine öğrenmesi modellerinin şeffaflaştırılabilmesi amacıyla SHAP ve LIME yöntemleri ile açıklanabilirlik kazandırılmıştır.
- CM1 ve PC1 veri setlerindeki tahmini en az etkileyen veya tahmin performansını negatif yönde etkileyen özellikler belirlenmiş ve veri sette özellik çıkartma (feature elimination) işlemi yapılmıştır. Doğruluk, Kesinlik, Duyarlılık, F1-Puanı, AUC-ROC metrikleri tekrar hesaplanmış olup veri setinin eksiltilmemiş haliyle karşılaştırılmıştır. Böylece gereksiz veya negatif etkili özelliklerin model performansına olan etkisi değerlendirilmiş, daha yalın ve genellenebilir tahmin modelleri elde edilmesi amaçlanmıştır.

SDP problemlerinde özellik çeşitliliği, etiketlenmiş veri yapısı ve yerel, küresel düzeyde yorumlanabilirlik sağlaması gibi nedenlerle literatürde yaygın biçimde kullanılan CM1 ve PC1 veri setlerinde gözlenen sınıf dengesizliği problemi, SMOTE yöntemi uygulanarak giderilmiştir. Verilerde standartlaştırma işlemi yapılarak her özelliğin başlangıç ağırlığı eşit hale getirilmiş, model hangi özelliğin gerçekten daha önemli olduğunu kendisi öğrenmiştir. Bu ön işleme adımı RF, GB gibi klasik makine öğrenmesi ve MLP, RNN, LSTM, CNN gibi derin öğrenme algoritmalarının daha kararlı ve güvenilir şekilde optimize edilebilmesi açısından önemlidir. SDP probleminde makine öğrenmesi ve derin öğrenme modellerinden faydalanılmış olup, bu modellerin karar alma süreçlerini şeffaflaştırabilmek için ise XAI yöntemleri uygulanmıştır.

2.3. Model Geliştirme ve Sınıflandırma

Bu çalışmada tercih edilen algoritmalar, SDP probleminin yapısal özellikleri, PROMISE veri setlerindeki yüksek çoklu bağlantı, özellik çeşitliliği ve hata oluşum süreçlerinin doğrusal olmayan doğası dikkate alınarak seçilmiştir. RF ve GB gibi ağaç tabanlı yöntemler, hem karmaşık bağımlılıkları doğrusal olmayan karar bölümleriyle modelleyebilme kapasitesi hem de küçük ve orta ölçekli yazılım metrik setlerinde güçlü genellenebilirlik göstermektedir [64], [65]. Bu algoritmaların hata tahmininde verimli sonuçlar elde etmesi [27], [33], [56], [66] ve SHAP gibi XAI yöntemleriyle yüksek

uyumlu açıklanabilirlik yapıları sunması [21] nedeniyle tercih edilmiştir. MLP, CNN, RNN ve LSTM derin öğrenme algoritmaları ise, yazılım metriklerinin karmaşık, çok boyutlu ve doğrusal olmayan etkileşimleri öğrenebilme kapasiteleri, literatürde ortaya konan ampirik kanıtların; bu algoritmaların karmaşık yapısal örüntüleri yakalamada klasik yöntemlere göre üstünlük göstermesi [67], [68], [69] nedeniyle seçilmiştir. RNN ve LSTM algoritmaları özellikle yazılım geliştirme süreçlerindeki ardışık bağımlılıkları ve etkileşimsel davranışları modelleme yetenekleri [67], [69], CNN mimarisinin ise, girdilerdeki yerel yapısal örüntüleri, konumsal ilişkileri ve bu örüntülerin taşıdığı anlamsal bağlamı etkili bir biçimde yakalayabilme kabiliyeti, modellerin tercih edilmesinde belirleyici olmuştur [68]. MLP tabanlı bileşenlerin, doğrusal olmayan karar sınırlarını temsil etme kapasitesinin hem geleneksel metriklere dayalı hem de semantik özelliklerle zenginleştirilmiş senaryolarda mevcut yöntemlerin üzerinde bir başarı sağlaması tercih sebebi olmuştur [69]. Bahaa vd.'nin, sistematik incelemesine göre, SDP literatüründe birincil çalışmaların çoğu CNN ve RNN gibi tekil derin öğrenme modellerini kullanmış ve bu modellerin karmaşık örüntüleri yakalamadaki avantajından yararlanılmıştır [63].

Model çeşitliliğinin kullanılması, topluluk öğrenmesi, derin öğrenme ve ardışık öğrenme gibi farklı öğrenme paradigmalarının aynı problem üzerinde bütüncül biçimde değerlendirilmesine olanak tanımıştır. Böylece hem modeller arası performans farklarının sistematik olarak karşılaştırılması hem de açıklanabilirlik yöntemlerinin model davranışı üzerindeki etkilerinin incelenmesi için geniş bir metodolojik çerçeve oluşturulmuştur. Bu çalışmada hem genellenebilirliğin hem de açıklanabilirliğin daha bütüncül bir yaklaşımla değerlendirilebilmesini sağlamak amacıyla, yapısal olarak farklı özelliklere sahip model gruplarının birlikte incelenmesi bilinçli bir metodolojik tercih olarak benimsenmiştir.

Çalışmada model eğitim süreci, SDP literatüründe tercih edilen ve önerilen veri hazırlama, veri ön işleme ve öğrenme adımları temel alınarak düzenlenmiştir. İlk olarak CM1 ve PC1 veri setlerinde bulunan belirgin sınıf dengesizliği, tahmin modellerinin çoğunluk sınıfa aşırı uyum sağlamasını engellemek amacıyla SMOTE yöntemi kullanılarak giderilmiştir [70]. Ağaç tabanlı modeller (RF, GB) doğrudan ölçeklenmiş özellikler üzerinde eğitilmiştir; bu modellerin hiperparametreleri literatürde önerilen varsayılan yapılandırmalar üzerinden belirlenmiş, öğrenme süreci tüm örüntüleri kapsayacak şekilde tam veri üzerinde yürütülmüştür [64], [65].

Her model, mimarisinin gerektirdiği doğrultuda veri tensörleri yeniden şekillendirilerek eğitilmiş; böylece aynı veri setinin farklı öğrenme paradigmaları altında nasıl davrandığı bütüncül bir biçimde değerlendirilmiştir. Aynı deneysel koşullar altında yürütülen bu süreç, modellerin hem ayırt edici tahmin gücünü hem de özellik setine duyarlılık düzeylerini nesnel bir şekilde karşılaştırmayı mümkün kılmıştır. Sonuç olarak önerilen metodolojik çerçeve, SDP çalışmalarında model güvenilirliğini artıran ve performans farklılıklarının kaynağını daha açık biçimde ortaya koyan sağlam bir değerlendirme stratejisi sunmuştur.

Modellerde kullanılan katman boyutları, aktivasyon fonksiyonları, optimizasyon teknikleri ve eğitim tekrar sayıları, PROMISE veri setlerinin görece küçük örneklem boyutu ve düşük boyutluluğu dikkate alınarak aşırı öğrenmeyi engelleyen, ancak doğrusal olmayan ilişkileri yakalamaya yeterli bir kapasite sunacak şekilde belirlenmiştir. MLP mimarisinde tercih edilen 64–32 nöronlu yapı ile ReLU–Sigmoid kombinasyonu, tabular veri üzerinde yaygın olarak kullanılan ve gradyan kaybolmasını sınırlayan bir yaklaşım olarak literatürde önerilmektedir [71]. CNN mimarisinde küçük kernel boyutları ve düşük filtre sayıları, tabular veri üzerinde yerel örüntü yakalamak için kullanılan hafif konfigurasyonlarla uyumludur. Tüm ağlarda Adam optimizasyon algoritması ve 100 epoch'luk eğitim şeması, küçük veri setlerinde kararlı yakınsama sağladığı için sıkça tercih edilmektedir. Ampirik değerlendirmeler, Adam optimizasyon algoritmasının pratik uygulamalarda tutarlı biçimde üstün performans sergilediğini göstermekte; algoritmanın, karşılaştırmalı analizlerde diğer stokastik optimizasyon yöntemlere kıyasla daha hızlı, daha istikrarlı ve daha güvenilir bir yakınsama davranışı sunduğu ortaya konulmaktadır [72]. Açıklanabilirlik aşamasında KernelExplainer ve TreeExplainer'ın tercih edilmesi, derin ve ağaç tabanlı modellerin doğrusal olmayan karar yapısını en doğru biçimde temsil eden SHAP fonksiyonlarının bu varyantlarıyla uyumluluk gereğinden kaynaklanmaktadır [21].

Çapraz Doğrulama (CV), makine öğrenimi modellerinin başarılarını değerlendirmek için kullanılan bir yöntemdir, sonuçların güvenilirliğini artırır ve eğitim setindeki öğelerin sırasından kaynaklanan sorunları azaltır [72]. Verilerin daha iyi kullanılmasına yardımcı olur ve algoritma performansı hakkında daha fazla bilgi verir. Bu yöntem kullanılmadan önce bir k değeri seçilir. Bu çalışmada, k değeri 5 seçilmiştir. Bunun anlamı veri kümesi 5 eşit parçaya bölünmüştür. Her işlemden ayrı ayrı sonuçlar üretilmiştir. Bu sonuçların

aritmetik ortalaması alınarak genel bir hata veya başarı metriği sonucu elde edilmiştir. Literatürdeki çalışmalara göre, değeri 5 olarak seçilen çapraz doğrulamanın önyargısı %95 güven aralığındadır. On katlı çapraz doğrulama tahminlerin çoğunda oldukça iyidir [73] fakat PROMISE veri setleri binler seviyesinde örneğe sahip olduğundan, yüksek fold sayıları her iterasyonda daha küçük eğitim parçaları üretmekte; bu da modelin öğrenme kararlılığını azaltabilmektedir. Bu nedenle 5-fold CV, hem düşük varyans–yüksek verimlilik dengesini sağlaması hem de aşırı bölünmüş eğitim alt kümelerinin önüne geçmesi bakımından daha doğru bir tercihtir.

Modellerin performansı, SDP literatüründe sıkça kullanılan sınıflandırma metrikleri olan Accuracy (Doğruluk), Precision (Kesinlik), Recall (Duyarlılık), F1-puanı ve AUC-ROC ölçütlerine [68] göre değerlendirilmiştir. PROMISE CM1 ve PC1 veri setleri dengesiz sınıf dağılımı içerdiğinden, yalnızca doğruluk metriği model performansını sağlıklı biçimde yansıtamayabilir, örneğin yalnızca çoğunluk sınıfını tahmin eden bir model yüksek doğruluk elde edebilir. Bu nedenle duyarlılık ve kesinlik gibi sınıflar arasındaki dengeyi dikkate alan metrikler daha anlamlı sonuçlar sunmaktadır [15]. Duyarlılık, modelin hata tahminini yakalama olasılığını ölçtüğünden SDP problemlerinde kritik öneme sahiptir, çünkü hatalı modüllerin gözden kaçması bakım, maliyet ve güvenilirlik açısından ciddi risklere yol açmaktadır. Kesinlik, modellerin hatalı olarak belirttiği modüllerin, hatalı olup olmadığını gösterdiği için önemlidir. F1-puanı, duyarlılık ve kesinlik metriklerinin harmonik ortalaması ile bulunur. AUC-ROC metriği ise, modelin tüm olası eşik değerleri boyunca pozitif ve negatif sınıfları ayırt etme kapasitesini değerlendirerek özellikle farklı modellerin genel ayrıştırma güçlerinin karşılaştırılmasında güvenilir bir ölçüt sağlamıştır [12], [25].

Model çıktıları, doğruluk, kesinlik, duyarlılık, F1 ve AUC-ROC gibi sınıflandırma metrikleri üzerinden değerlendirilmiş; her bir modelin performansı SHAP tabanlı açıklanabilirlik analizleriyle birlikte yorumlanarak hem öğrenme başarımı hem de karar mekanizmasının şeffaflığı birlikte ele alınmıştır. Daha sonra SHAP sonuçlarına göre özellik azaltma (feature elimination) işlemi yapılıp tekrar performans ölçümü yapılmıştır.

Metrik sonuçlarının yorumlanması, modellerin hangi koşullarda daha başarılı olduğunu nicel olarak göstermektedir; ancak bu bulgular, tek başına model davranışının altında yatan karar mekanizmalarını açıklamaya yetmemektedir. SDP bağlamında geliştirilen

modellerin büyük bölümü “kara kutu” niteliği taşıdığı için, modelin belirli bir örneği neden hatalı ya da hatasız sınıflandırdığı çoğu zaman doğrudan anlaşılamamaktadır. Bu nedenle, performans değerlendirmesini tamamlayan ikinci bir aşama olarak açıklanabilirlik analizi gereklidir. SHAP ve LIME gibi XAI yöntemleri, hem küresel düzeyde hangi özelliklerin model kararlarına ağırlık verdiğini ortaya koymakta hem de tekil örnekler üzerinde modelin nasıl davrandığını şeffaflaştırarak performans bulgularının arka planını anlamlandırmayı mümkün kılmaktadır. Böylece, yalnızca metriklerle sınırlı bir değerlendirme yerine, modelin neyi, nasıl ve hangi gerekçelerle öğrendiğini bütüncül biçimde inceleyen daha sağlam bir metodolojik çerçeve elde edilmiştir.

Böylece hem global hem de lokal açıklama yöntemlerinin birlikte kullanılması, modellerin yalnızca tahmin performansı açısından değil, karar mekanizmalarının anlaşılabilirliği ve güvenilirlik açısından da SDP problemine uygun şekilde uyarlanmasını sağlamıştır. Bu yaklaşım, XAI literatürünün “model–veri hizalanması” ve “insan sezgisine yakın açıklama gerekliliği” vurgularıyla da örtüşmektedir [74].

2.4. Yapay Zekâ Açıklama Yöntemleri

2.4.1. SHAP Yöntemi

SHAP, Shapley değerlerinin etkin ve hızlı biçimde hesaplanmasına olanak tanıyan ve her bir özelliğin model tahminine katkısını (Shapley değeri) belirleyerek belirli bir örnek için üretilen tahmini açıklamada kullanılan bir yöntemdir [75], [76]. Tahminler, modelin ortalama tahmini ile karşılaştırılarak, katkı temelli açıklamalar elde edilmektedir [75].

SHAP, yalnızca en önemli özellikleri belirlemek ve vurgulamak amacıyla değil [77], [78] aynı zamanda açıklanabilirlik açısından önemli olan doğruluk, tutarlılık ve eksiklik özellikleri arasında denge kurabilen bir yöntem olarak da benimsenmiştir [79]. SHAP ile her bir modelin tahmin üretirken hangi özellikleri ne ölçüde kullandığı küresel ve yerel düzeyde değerlendirilmiştir. SHAP Özellik Önem Grafikleri (SHAP Feature Plot) ve SHAP Özet Grafikleri (SHAP Summary Plot) yardımıyla SHAP’in katkı temelli açıklama yapısı, modele pozitif veya negatif yönde etki eden özellikleri ve onların hassas biçimde belirlemeyi mümkün kılmıştır [21]. SHAP Özellik Önem Grafikleri ve SHAP Özet Grafikleri çıktıları, hangi özelliklerin modeller üzerinde daha fazla katkı sağladığını ve bu katkının yönünü ortaya koyarak küresel düzeyde açıklanabilirlik sağlamıştır. Ayrıca

SHAP değerleri kullanılarak açıklanabilirlik güdümlü özellik azaltımı (feature elimination) gerçekleştirilmiş; düşük katkılı öznitelikler çıkarılarak modeller yeniden eğitilmiş ve hata tahmini için verinin eksiltilmemiş hali ile performans ilişkisi araştırılmıştır.

SHAP özellik önem grafiğinde x koordinatında numerik olarak etki değeri, y koordinatında ise yukarıdan aşağıya doğru etki değeri azalan özellikler listelenmektedir. Özelliklerin ortalama mutlak SHAP değerlerini kullanarak modellerin hangi özellikleri daha baskın biçimde kullandığını sıralı ve karşılaştırılabilir bir formatta sunmuştur. Bu grafik, özellikle modeller arası global önem sıralaması karşılaştırmaları için kullanılmış olup araştırma kapsamında "en etkili" veya "en az etkili" özelliklerin belirlenmesinde temel araç olmuştur.

SHAP özet grafikleri, kullanılan veri sette bulunan her bir özelliğin SHAP değerine katkısını ve bu katkının yönünü göstermektedir. Mavi renkteki noktalar düşük etki değerlerini, kırmızı renkteki noktalar yüksek etki değerlerini ifade etmektedir. Pozitif yön tahmin performansını artırırken, negatif yön tahmin performansını azaltmaktadır. Grafikte özelliklerin tahmine olan katkıları, yukarıdan aşağıya azalmaktadır [53].

SHAP özellik önem grafikleri ve SHAP özet grafikleri için tüm modellerin aynı ölçekleme yöntemi (scaler), aynı rastgelelik ayarları (seed) ve aynı eğitim-test ayrımı altında çalıştırılmış olmasına rağmen sonuçların birbirleriyle tam olarak örtüşmemesi kullanılan öğrenme algoritmalarının doğasından ve açıklayıcı yöntemlerin hesaplama mantığından kaynaklanmaktadır. Her modelin istemi ve öğrenme şekli farklılık gösterebilir. Ağaç tabanlı modeller (RF, GB), yalnızca bölünmelerde kullanılan birkaç baskın özelliğe yüksek önem atfederken, derin öğrenme tabanlı modeller (LSTM, MLP vb.) girdilerden daha dağıtık bir şekilde öğrenir.

SHAP şelale grafiği yöntemi tek bir örnek üzerindeki tahmini ayrıntılı biçimde çözümleyerek her bir özelliğin modele yaptığı katkının büyüklüğünü ve katkının pozitif mi negatif mi olduğunu açık biçimde ortaya koymaktadır. Bu yöntem, modelin belirlenen bir veri modülü üzerinden yaptığı tahmini, hangi özelliklerin bu tahmini güçlendirdiği ya da zayıflattığını açıklanabilir hale getirmiştir [21]. Özellikle derin öğrenme modellerinde karar süreçlerinin basit doğrusal ilişkilere dayanmadığı durumlarda, SHAP şelale grafiği

yöntemi, modelin içsel karar dinamiklerine ilişkin daha ayrıntılı ve yorumlanabilir bir şeffaflık sağlamıştır.

SHAP yaklaşımı, bir özelliğin model tahminine yaptığı katkıyı, tüm olası özellik kombinasyonları üzerindeki ortalama marjinal katkıyı hesaplayarak belirler. Bu sayede yöntem, matematiksel olarak temellendirilmiş ve tutarlı açıklamalar sunar. Böylece her bir özelliğin model çıktısını olumlu veya olumsuz yönde nasıl etkilediği açık biçimde ortaya konulabilir [80]. Ayrıca SHAP'in sağladığı hızlı hesaplama mekanizması, küresel model yorumlaması için gerekli olan çok sayıda Shapley değerinin pratik bir şekilde hesaplanmasını mümkün kılmaktadır [75].

$$f(x) = E[f(x)] + \sum_{i=1}^{M} \varphi_i(x) \text{ [81]} \quad (2.1)$$

Formülde geçen f(x) değeri örneğin tahmin değerini vermektedir. E(f(x)) değeri ise özelliklerin etkisi dikkate alınmadan beklenen model çıktı değeridir. Beklenen model çıktı değeri ile özelliklerin hepsinin katkı değerleri toplandığında, örneğin tahmin değeri elde edilir [81].

SHAP değeri φ_i, herhangi bir özellik için şu şekilde ifade edilebilir:

$$\varphi i = \sum_\{S \subseteq N \setminus \{i\}\} \frac{|S|!\,(|N|-|S|-1)!}{|N|!} [\, f(S \cup \{i\}) - f(S) \,] \quad (2.2)$$

Formül 2.2'de; N, özellik kümesini, S ise seçilen özelliğin dışındaki özelliklerin alt kümesini ifade eder. f(S) yalnızca S kümesindeki özellikler mevcut olduğunda modelin çıktısını gösterir ve $f(S \cup \{i\})$, i özelliği S kümesine eklendiğinde elde edilen çıktıdır [82].

2.4.1. LIME Yöntemi

LIME, yerel tahminlerin açıklanabilirliğini sağlamak amacıyla yaygın olarak kullanılan ve yorumlanması kolay bir yöntemdir [76], [80]. Bu yöntem, açıklanmak istenen veri örneğinin çevresinde oluşturulan örneklerden elde edilen çıktı olasılıklarını kullanarak yerel bir doğrusal vekil model kurar. Oluşturulan bu vekil modelde hesaplanan ağırlıklar, girdi özelliklerinin model tahminine ne ölçüde katkı sağladığını değerlendirmek için kullanılır. Ayrıca LIME, modelden bağımsız bir yaklaşım olduğundan farklı makine öğrenmesi modellerine kolaylıkla uygulanabilmektedir [76].

LIME yöntemi, modelin bireysel örnekler için verdiği kararları yerel düzeyde açıklayarak, en etkili özellikleri belirlemek için kullanılan en yaygın yöntemdir [16], [28]. SDP çalışmalarında kullanım kolaylığı [6] ve hızlı çalışması nedeniyle sıklıkla kullanılmaktadır. Bu sayede geliştiriciler, özellikle hata tahmini çalışmalarında elde edilen sonuçları kod parçacığı düzeyinde daha anlaşılır şekilde yorumlayabilmektedir [16]. Bu yöntemde veri set üzerinden belirlenen bir örneğin hata tahmininde RF, GB ve MLP gibi modellerin kullanımı oldukça elverişlidir. Özellikle son kullanıcıların belirli bulgulara ilişkin kaygılarının giderilmesine yönelik anlamlı katkılar sunar. Yerel tahminlerde şeffaflığın kritik bir gereklilik olduğu, özellikle düzenleyici ortamlar gibi kullanıcıya dönük senaryolar açısından tercih edilen bir yöntemdir [9]. Bu gerekçeler doğrultusunda, SDP literatüründe yaygın biçimde benimsenmiş ve etkinlikleri önceki çalışmalarla ortaya konmuş olmaları nedeniyle söz konusu yöntemler tercih edilmiştir. SHAP yöntemi ise, gerek ağaç tabanlı modellerde (örneğin rastgele ormanlar, gradyan güclendirme) gerekse sinir ağlarında etkili biçimde uygulanabilir [9]. Veri setlerin sunduğu özelliklerin, tahmin üzerindeki katkısını sıralayan ve katkı yönünü gösteren bir çerçeve sağlar. Hata tahmin sürecini şeffaflaştırabilmek amacıyla anlaşılır bir yapı sağlar, bu durum hata tahmin bulgusuna güveni ve pratikte kullanılabilirliği artırır [12]. Bunun yanında SHAP temelli yöntemlerin, literatürdeki benzer yöntemlerle karşılaştırıldığında daha yüksek performans sergilemesi ve insan sezgisiyle diğerlerine göre daha yüksek düzeyde uyum göstermektedir [21].

$$\text{X i(x)} = \arg\min_\{g \in G\} \; [\mathcal{L}(f, g, \pi_x) + \Omega(g)] \text{ [81]} \tag{2.3}$$

Bu formülde G, yorumlanabilir modeller kümesini temsil ederken, L fonksiyonu $\pi_x$tarafından tanımlanan yerel bölgede f ve g modelleri arasındaki uyum düzeyini ölçmektedir. $\Omega(g)$ terimi ise model karmaşıklığını sınırlayarak daha sade ve yorumlanabilir bir açıklama modeli elde edilmesini sağlamaktadır [81]. Burada amaç, formül 2.4'teki kayıp fonksiyonunu en aza indirmektir [82].

$$\mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.4}$$

Burada $\mathcal{L}(f, g, \pi_x)$: π_x tarafından tanımlanan bölgede g'nin f'yi ne kadar doğru şekilde yaklaşık olarak temsil ettiğini ölçer. Ω(g) ise, yorumlanabilirliği sağlamak amacıyla kullanılan bir karmaşıklık sınırlamasıdır. Doğrusal bir modelde sıfır olmayan katsayıların sayısı buna örnek verilebilir.

$\pi x(z)$: örneği ile $x$örneği arasındaki yakınlığı ölçen bir fonksiyondur ve genellikle üstel çekirdek (exponential kernel) olarak tanımlanır [82].

$$\pi_x(z) = \exp\left(-\frac{D(x,z^2)}{\sigma^2}\right) \tag{2.5}$$

Burada $D(x,z)$, bir mesafe ölçütünü ifade eder, σ ise komşuluğun genişliğini kontrol eder.

Kayıp fonksiyonu L çoğunlukla, karesel kayıptır:

$$\mathcal{L}(f,g,\pi_x) = \sum_{z\in Z}^{\infty} \pi_x(z)(f(z)-g(z))^2 \tag{2.6}$$

Burada Z, x örneği etrafında oluşturulmuş bozulmuş örneklerden oluşan bir kümedir. Genel metodoloji, sentetik örnekler üretmek için girdi örneğini bozmayı ve bu örnekleri orijinal kara kutu model üzerinden değerlendirmeyi içerir. Bunun için algoritma şu adımları uygular: Önce x etrafında bozulmuş örneklerden oluşan veri kümesi Zoluşturulur; ardından kara kutu modelin tahminleri $f(z)$her $z \in Z$için elde edilir. Yerel yaklaşımda her örneğin önemi, $z$ile $x$arasındaki yakınlığı ölçen $\pi_x(z)$ile ağırlıklandırılır. Daha sonra, yorumlanabilir model g, ağırlıklar $\pi_x(z)$kullanılarak Zveri kümesine uygulanır ve karmaşıklık cezası $\Omega(g)$dikkate alınarak kayıp $\mathcal{L}(f,g,\pi_x)$minimize edilir. Sonuç olarak, yorumlanabilir model g, $f(x)$tahminini açıklamak için kullanılır; doğrusal modellerde ise katsayılar her bir özelliğin katkısını açıkça gösterir [82].

Bu yaklaşımda LIME, yorumlanabilir vekil değişkenler aracılığıyla model davranışını yerel olarak yaklaşık biçimde temsil eder. Belirli bir örnek x için yöntem, söz konusu örneğin çevresinde komşu örnekler üretir, bu örnekler için model tahminlerini elde eder ve ardından bu tahminleri açıklamak amacıyla ağırlıklı bir doğrusal model uydurur. Böylece her bir özelliğin ilgili tahmine olan etkisi yorumlanabilir bir şekilde ortaya konulur [81].

3. BÖLÜM

DENEYSEL ÇALIŞMALAR

Tez çalışmasında yazılım hata tahmin modellerinin açıklanabilirliğini değerlendirmek amacıyla NASA PROMISE CM1 ve PC1 veri setleri üzerinde gerçekleştirilen modelleme sürecine ait bulgular sistematik biçimde ortaya konmuş ve analiz edilmiştir. AS1 sorusunu cevaplayabilmek için, RF, GB, RNN, CNN, LSTM ve MLP modelleri eğitilmiş; 5-katlı çapraz doğrulama kullanılarak, doğruluk, kesinlik, duyarlılık, F1-Puanı ve AUC-ROC metriklerine göre tahmin performansları değerlendirilmiştir. Daha sonra modellerin karar süreçlerinin daha şeffaf biçimde ortaya konulabilmesi amacıyla, küresel açıklanabilirlik SHAP Özellik Önem Grafiği ve SHAP Özet Grafiği ile; yerel açıklanabilirlik ise SHAP Şelale Grafiği ve LIME yöntemi ile elde edilmiştir. Bu analizler sayesinde özelliklerin modele katkı düzeyleri, katkı sıralamaları ve katkı yönleri ayrıntılı biçimde belirlenmiş; böylece AS2 ve AS3 araştırma sorularının ampirik olarak değerlendirilmesi sağlanmıştır. Ek olarak, küresel açıklanabilirlik bulgularının istatistiksel tutarlılığını değerlendirmek amacıyla SHAP ile elde edilen özellik önem sıralamaları, Pearson korelasyon katsayılarına dayalı önem ölçümleriyle karşılaştırılmıştır. SHAP yöntemi sonuçları karşılaştırıldığında her iki veri setinde SHAP – Pearson uyumsuzluğu, modellerin insan sezgisiyle örtüşmeyen, karmaşık ve doğrusal olmayan örüntüler yakaladığını göstermekte; bu durum açıklanabilirliğin yalnızca korelasyon temelli yöntemlere dayandırılmasının yetersizliğine işaret etmektedir. Özellikle derin öğrenme tabanlı modellerin doğrusal ilişkilere sıkı biçimde bağlı kalmadığı, bunun yerine zaman içinde oluşan çok boyutlu ve doğrusal olmayan yapısal örüntüleri öğrenme eğiliminde olduğu görülmüştür. Son aşamada, CM1 ve PC1 veri setlerinin kullanılan modellere göre hata tahminine en az katkı sağlayan özellikler belirlenmiş ve bu doğrultuda özellik azaltımı yapılmıştır Özellik azaltımı sonrası modeller yeniden değerlendirilmiş; elde edilen sonuçlar AS4 kapsamında incelenmiş ve

performans metriklerinde anlamlı iyileşmeler gözlemlenmiştir.

3.1. Deneysel Kurulum

Bu çalışma, Python 3.10 programlama dili kullanılarak Google Colaboratory ortamında yürütülmüştür. Deneylerde kullanılan temel kütüphaneler ve sürümleri sırasıyla NumPy (1.24.x), Pandas (2.0.x), Scikit-learn (1.3.x), TensorFlow/Keras (2.13.x), ve SHAP (0.43.x)'tir. Tüm deneyler, sonuçların tekrarlanabilirliğini sağlamak amacıyla sabit bir rastgelelik değeri (random_state = 42) kullanılarak gerçekleştirilmiştir.

Çalışmada, NASA PROMISE veri havuzunda yer alan CM1 ve PC1 yazılım hata veri setleri kullanılmıştır. Veri setlerindeki tüm sayısal özellikler, modeller arası karşılaştırılabilirliği sağlamak ve özellikle ağ tabanlı modellerde daha kararlı bir öğrenme süreci elde etmek amacıyla standartlaştırma yöntemiyle ölçeklendirilmiştir. Eğitim ve test ayrımı, her iki veri seti için de %80 eğitim – %20 test oranında ve aynı rastgelelik koşulları altında yapılmıştır.

Çalışmada RF, GB, MLP, RNN, LSTM ve CNN modelleri kullanılmıştır. Derin öğrenme tabanlı modellerde, giriş verileri modele özgü biçimde yeniden düzenlenmiştir. MLP, doğrudan ölçeklenmiş özellik vektörleri üzerinden eğitilirken; RNN ve LSTM modellerinde her örnek, zaman boyutunu temsil edecek şekilde (1 × özellik sayısı) formatında ele alınmıştır. CNN modelinde ise her özellik tek boyutlu bir kanal olarak değerlendirilmiş ve giriş verisi buna uygun şekilde yeniden şekillendirilmiştir.

Tüm sinir ağı modelleri, ikili çapraz entropi (binary cross-entropy) kayıp fonksiyonu ve Adam optimizasyon algoritması kullanılarak eğitilmiştir. Öğrenme oranı 0.001 olarak sabitlenmiş, gizli katmanlarda ReLU, çıktı katmanında ise sigmoid aktivasyon fonksiyonu tercih edilmiştir. Modeller, herhangi bir erken durdurma veya hiperparametre optimizasyonu uygulanmaksızın, karşılaştırılabilirliği korumak amacıyla 100 epoch boyunca ve sabit hiperparametreler altında eğitilmiştir.

Model performansları, Doğruluk, Kesinlik, Duyarlılık, F1-Puanı ve AUC-ROC metrikleri kullanılarak değerlendirilmiştir. Ayrıca, modellerin karar mekanizmalarının yorumlanabilirliğini sağlamak amacıyla küresel ve yerel düzeyde SHAP ve LIME tabanlı açıklanabilir yapay zekâ yöntemleri kullanılmıştır.

Tablo 3.1. Kullanılan Modellerin Yapı ve Hiperparametre Özeti

Model	**Katman/Yapı Özeti**	**Aktivasyon**	**Optimizasyon**	**Epoch**	**Çapraz Doğrulama**	**Ek Parametreler**
MLP	Dense(64) → Dense(32) → Dense(1)	ReLU, ReLU Sigmoid	Adam	100	5-fold, shuffle=True, seed=42	Binary Crossentropy, Seed=42, KernelExplainer
RNN	SimpleRNN (32) → Dense(1)	tanh / Sigmoid	Adam	100	5-fold, shuffle=True, seed=42	Binary Crossentropy, 1×Feature reshape, KernelExplainer
LSTM	LSTM(32) → Dense(1)	tanh + gates, Sigmoid	Adam	100	5-fold, shuffle=True, seed=42	Binary Crossentropy, 1×Feature reshape, KernelExplainer
CNN	Conv1D(32) → MaxPool(1) → Flatten → Dense(32) → Dense(1)	ReLU, ReLU, Sigmoid	Adam	100	5-fold, shuffle=True, seed=42	Binary Crossentropy, Conv1D kernel=2/3, KernelExplainer
GB	GradientBoostingClassifier	n_estimators=100 ,learning_rate=0.1 , depth=3	-	-	5-fold, shuffle=True, seed=42	TreeExplainer, Seed=42

Tablo 3.1'de karşılaştırılabilirliği sağlamak amacıyla sabit hiperparametreler altında eğitilen modellere ait yapı ve eğitim ayarları sunulmaktadır.

3.2. Modellerin Performanslarının Karşılaştırılması

Tablo 3.2. CM1 ve PC1 Metriklerinin Sonuçları

Veri Set	Model	Doğruluk	Kesinlik	Duyarlılık	F1-Puanı	AUC-ROC
CM1	**RF**	0.930821	0.890636	**0.981769**	0.933785	**0.985782**
	GB	0.911247	0.874227	0.960589	0.914880	0.977458
	MLP	**0.933427**	**0.908707**	0.967226	**0.936876**	0.963951
	RNN	0.885086	0.85207	0.932962	0.890292	0.94481
	LSTM	0.894254	0.866144	0.934889	0.898594	0.943953
	CNN	0.886419	0.861351	0.924197	0.890599	0.937547
PC1	**RF**	**0.971123**	**0.956079**	0.987307	**0.971344**	**0.995033**
	GB	0.966845	0.947355	**0.988214**	0.967329	0.991533
	MLP	0.964706	0.95074	0.979684	0.964882	0.987825
	RNN	0.950267	0.930333	0.973376	0.951204	0.97637
	LSTM	0.945989	0.925383	0.970318	0.94706	0.980882
	CNN	0.951337	0.943982	0.959601	0.951483	0.983453

Tablo 3.2'de CM1 ve PC1 veri setleri için altı farklı model, beş farklı metriğe göre sonuçları bulunmaktadır. CM1 veri setinde en iyi doğruluk, kesinlik, F1-puanına sahip model MLP olurken, duyarlılık ve AUC-ROC metriklerinde en iyisi RF modeli olmuştur. CM1 veri setinde en kötü doğruluk, kesinlik, F1-Puanına sahip model RNN olurken, duyarlılık ve AUC-ROC metriklerinde en kötü performans gösteren CNN modeli olmuştur. PC1 veri setinde ise en iyi performansı doğruluk, kesinlik, F1-puanı ve AUC-ROC metriklerinde RF modeli gösterirken, duyarlılık metriğinde GB modeli göstermiştir. Yine PC1 veri setinde en kötü doğruluk, kesinlik, F1-Puanına sahip model LSTM modeli

olurken, duyarlılık metriğinde CNN modeli ve AUC-ROC metriğinde RNN modeli olmuştur.

***AS1**:* CM1 veri setinde doğruluk, kesinlik ve F1-Puanı açısından MLP modelinin; duyarlılık ve AUC-ROC açısından ise RF modelinin daha başarılı olduğunu, PC1 veri setinde ise genel performansta RF'nin, duyarlılık metriğinde ise GB modelinin öne çıktığını göstermektedir.

3.3. Özelliklerin Tahmin Sürecine Katkısının İncelenmesi (Küresel Açıklanabilirlik)

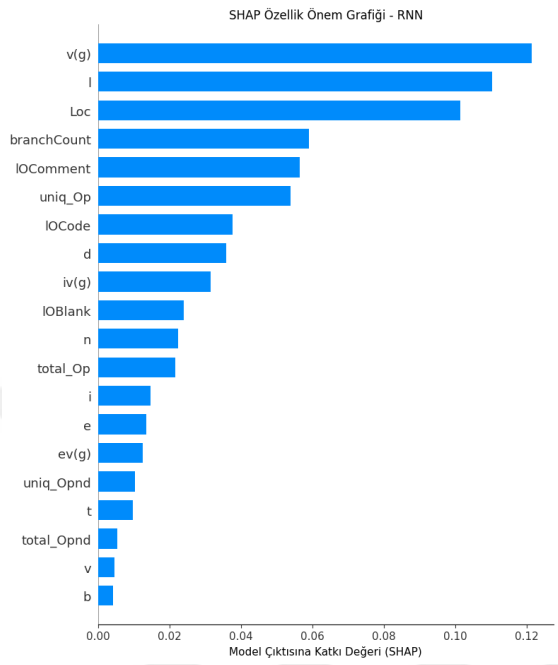
Küresel açıklanabilirliğin temel amacı, modelin karar sürecini tüm özellik kümesi, model girdileri ve kullanılan veriler ışığında açıklığa kavuşturmaktır. Küresel açıklamalar üreten yöntemler, modelin yalnızca hangi tahminleri ürettiğini değil, aynı zamanda bu tahminleri oluştururken özelliklerden yararlanma ölçüsünü, özelliklerin tahmine olan etkisini ve aralarındaki etkileşimleri nasıl ve neye göre değerlendirdiğini sunarak bütüncül şekilde şeffaflık sunmaktadır.

Küresel açıklanabilirlik için kullanılan SHAP Özellik Önem Grafiği ve SHAP Özet Grafikleri ile SDP'ye en çok etkisi olan özellikler ve bu özelliklerin etki yönünün pozitif mi, negatif mi olduğu bulguları elde edilmiştir.

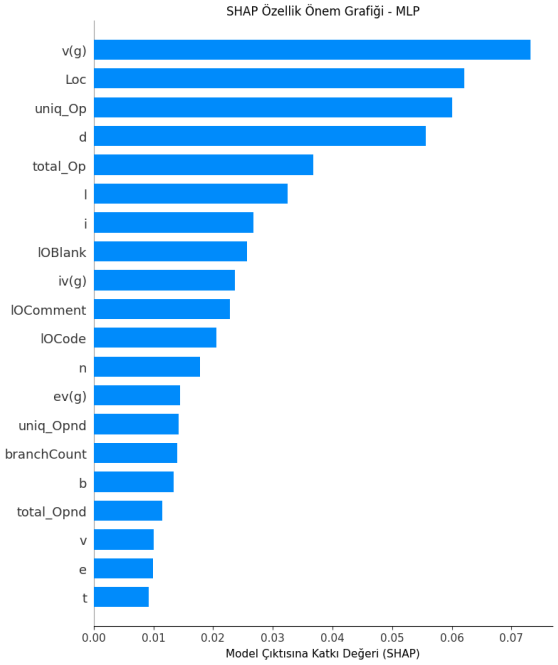
Kullanılan modellerden elde edilen bulgulara göre, CM1 veri setinin SDP'ye en çok katkı sunan özellikleri; Loc (kaynak kod satır sayısı), v(g) (McCabe karmaşıklık metriği) ve l (programın uzunluğu)'dir. PC1 veri setinin SDP'ye en çok katkı sunan özellikleri ise; i (algoritmanın zorluğu ve içeriği) ve uniq_Op (farklı operatör türlerinin sayısı)' dir.

3.3.1. CM1 Veri Seti ile SHAP Yöntemi Kullanımı

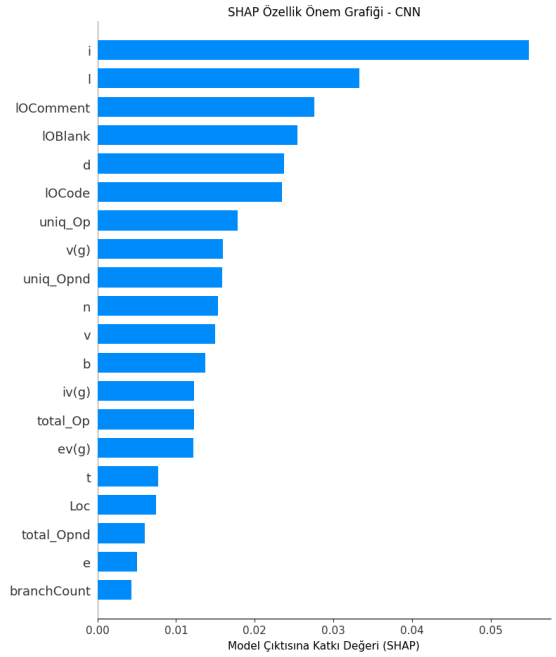
- SHAP Özellik Önem Grafikleri



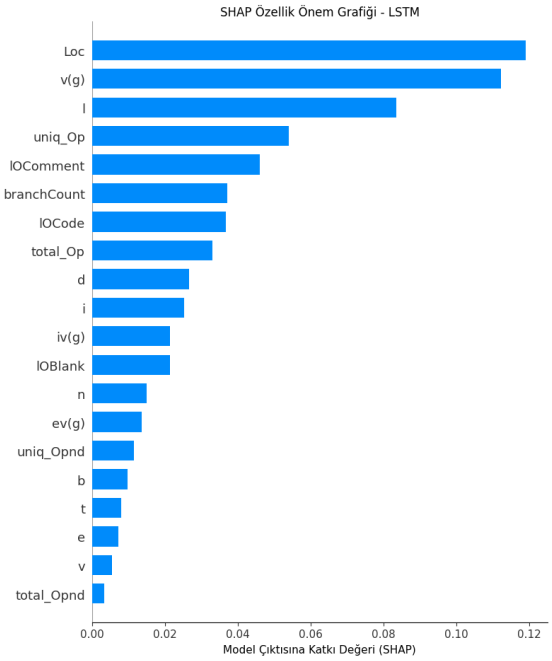
a) RNN tabanlı özellik önem grafiği



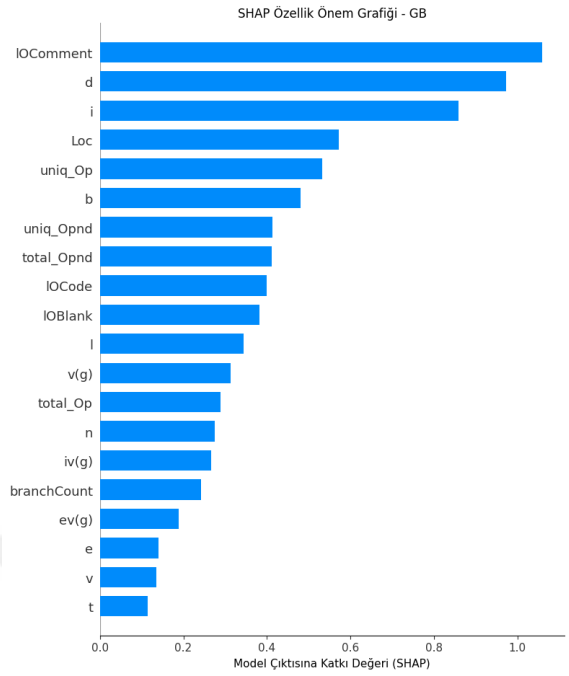
b) MLP tabanlı özellik önem grafiği



c) CNN tabanlı özellik önem grafiği



d) LSTM tabanlı özellik önem grafiği



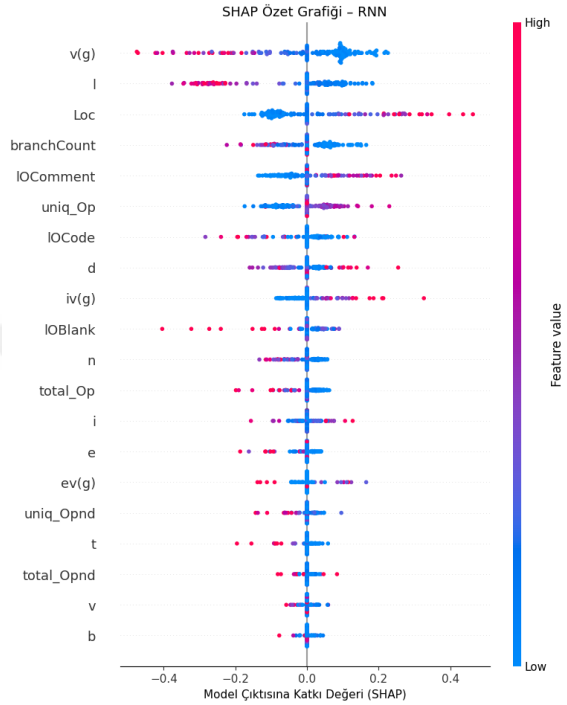
e) GB tabanlı özellik önem grafiği

Şekil 3.1. CM1 Veri Seti için SHAP yöntemi kullanılarak hesaplanan özellik önem grafikleri

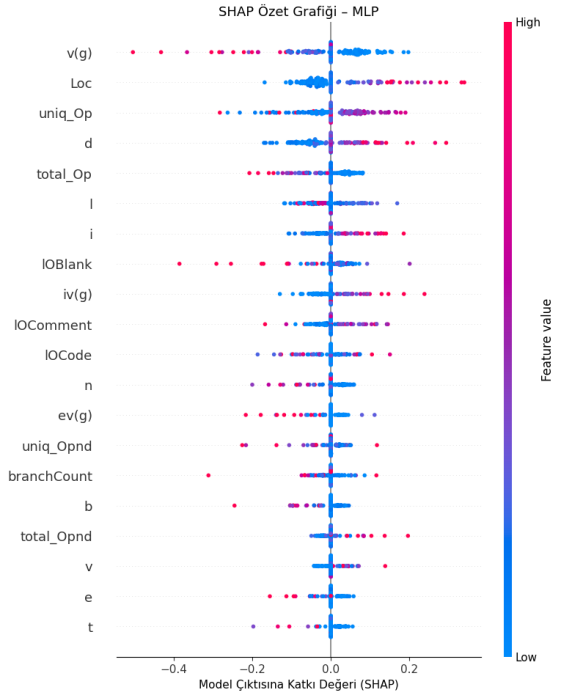
Şekil 3.1'de, küresel açıklama yapabilmek amacıyla CM1 veri setine beş farklı modelin uygulanması ile elde edilen Özellik önem grafiklerine ait bulgular bulunmaktadır. Bu veri sette IOCodeAndComment özelliğinin hiçbir etkisi olmadığı için grafiklerde bulunmamaktadır. Şekil 3.1.a'da RNN modeline göre en yüksek etki değerine sahip özellik v(g) olarak gözlemlenmektedir. v(g) özelliğini takip eden özellikler l, Loc, branchCount, IOComment, şeklinde sıralanmaktadır. RNN modeline göre yazılım hata tahmininde CM1 veri setinde bulunan b ve v özelliklerinin etki değerlerinin diğer özelliklere göre daha az olduğu gözlemlenmektedir. Şekil 3.1.b'ye göre v(g), Loc, uniq_Op, d, total_Op özellikleri en etkili özellikler olarak listelenmektedir. MLP modeline göre yazılım hata tahmininde CM1 veri setinde bulunan v, e ve t özelliklerinin etki değerlerinin diğer özelliklere nazaran daha az olduğu gözlemlenmektedir. Şekil 3.1.c'de CNN modeline göre i, l, IOComment, IOBlank, d özellikleri en etkili özellikler olarak listelenir, branchCount, e ve total_Opnd özelliklerinin etki değerlerinin diğer özelliklere göre daha az olduğu gözlemlenmektedir. Şekil 3.1.d 'de LSTM modeline göre en etkili özellikler Loc, v(g), l, uniq_Op, IOComment şeklinde sıralanmaktadır. total_Opnd, v e ve t özelliklerinin ise etki değerlerinin daha az olduğu gözlemlenmektedir. Şekil 3.1.e'de GB modeline göre hata tahmininde en etkili özellikler

IOComment, d, i, Loc, uniq_Op vb. şeklinde sıralanmaktadır. t, v ve e özelliklerinin katkı sıralamasında sonlarda olduğu gözlemlenmektedir.

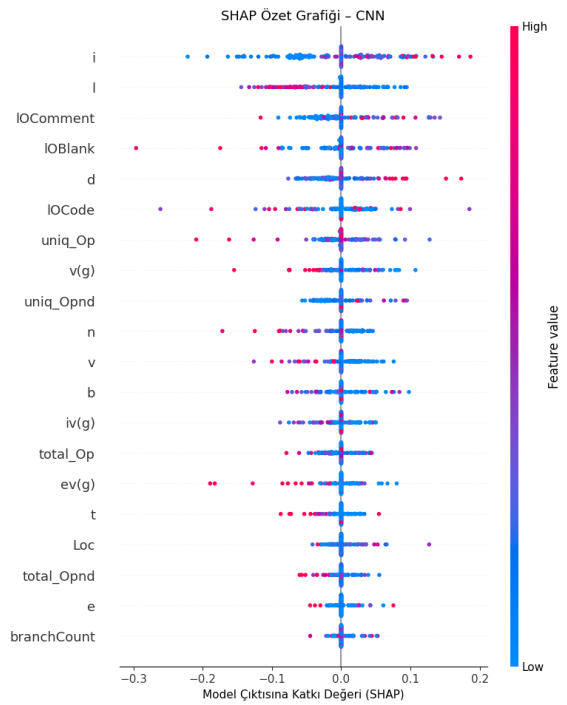
- SHAP Özet Grafikleri



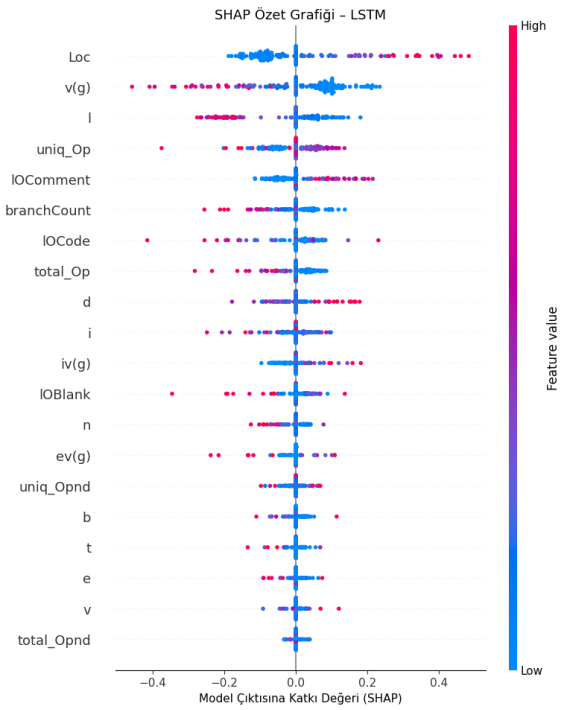
a) RNN tabanlı SHAP Özet Grafiği



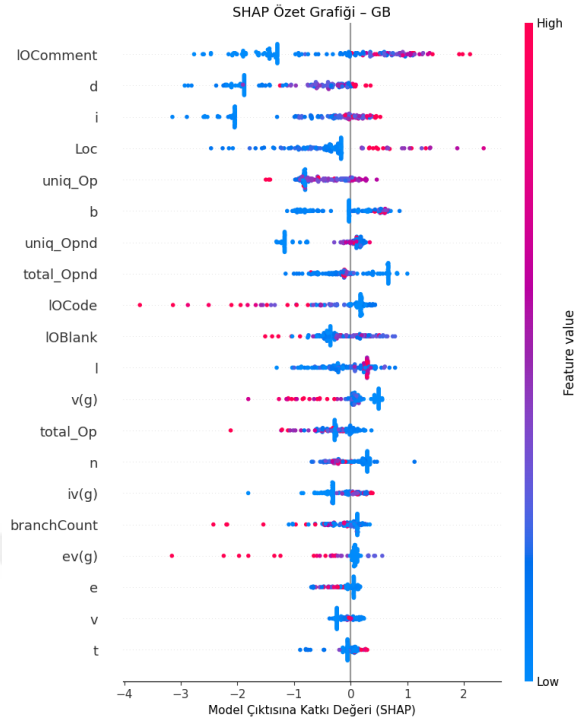
b) MLP tabanlı SHAP Özet Grafiği



c) CNN tabanlı SHAP Özet Grafiği



d) LSTM tabanlı SHAP Özet Grafiği



e) GB tabanlı SHAP Özet Grafiği

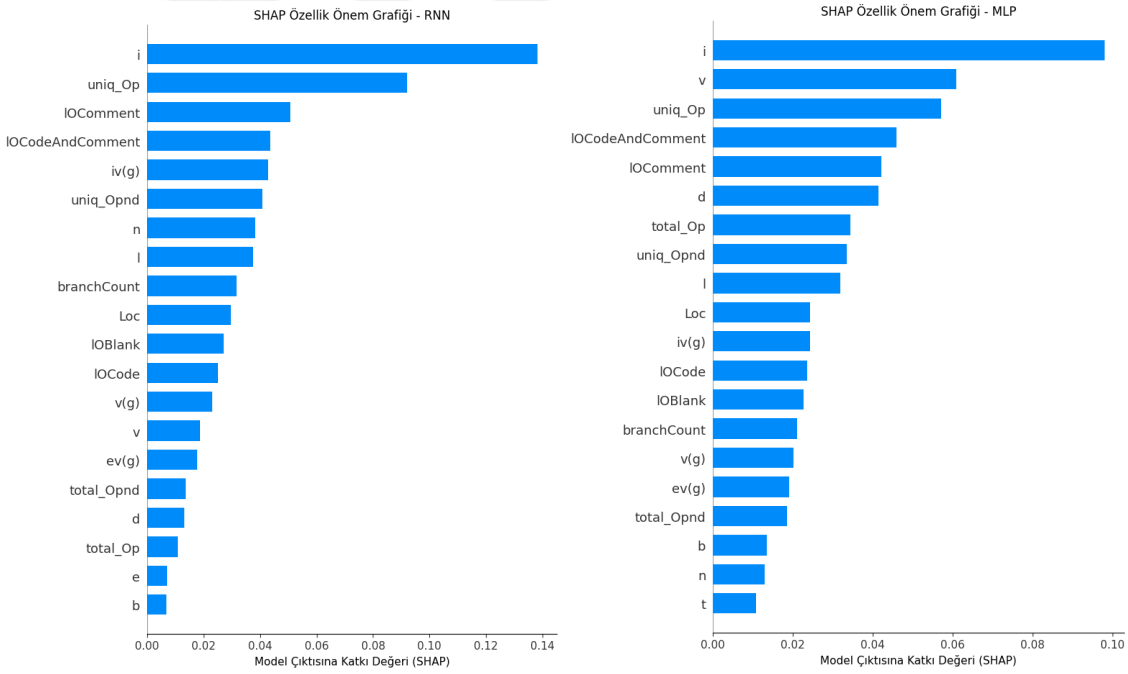
Şekil 3.2. CM1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP özet grafikleri.

CM1 veri setinde IOCodeAndComment özelliğinin hata tahminine hiçbir etkisi olmadığı için grafiklerde yer almamaktadır. Şekil 3.2.a'da tahmine en çok etki eden özellikler v(g), l, Loc, branchCount, IOComment şeklinde sıralanmaktadır. Grafikte bu özelliklerin olumlu ve olumsuz katkılarının olduğu görülmektedir. Fakat Loc, IOComment özelliklerinin daha yüksek özellik değerleri (kırmızı noktalar) pozitif yöndedir bu da tahmine pozitif yönde katkı sağladığını v(g), l ve branchCount özelliklerinin ise negatif yönde katkı sağlayıp tahmin performansını düşürdüğü, b ve v özelliklerinin etki değerinin diğerlerine nazaran daha düşük olduğu gözlemlenmektedir. Şekil 3.2.b'de özellikler etki sırasına göre v(g), Loc, uniq_Op, d, total_Op şeklinde listelenmektedir. Grafiğe göre bu özelliklerin olumlu ve olumsuz katkılarının olduğu görülmektedir. Ancak Loc, uniq_Op ve d özelliklerinin yüksek özellik değerlerinin pozitif yönde olması pozitif yönde, v(g), total_Op özelliklerinin ise negatif yönde katkı sağladığını göstermektedir. Grafiğe göre v, e ve t özelliklerinin etki değerinin daha az olduğu gözlemlenmektedir. Şekil 3.2.c'de özellikler etki değerlerine göre i, l, IOComment, IOBlank ve d şeklinde listelenmektedir. Fakat i, IOComment ve d özelliklerinin daha yüksek özellik değerleri (kırmızı noktalar) pozitif yöndedir bu da tahmine pozitif yönde katkı sağladığını l özelliğinin ise negatif

yönde katkı sağlayıp tahmin performansını düşürdüğü branchCount, e ve total_Opnd özelliklerinin ise etki değerinin son sıralarda olduğu görülmektedir. Şekil 3.2.d'ye göre en etkili özelliklerden Loc, uniq_Op, IOComment'in yüksek özellik değerlerinin pozitif yönde olması nedeniyle tahmine pozitif yönde, v(g), l özelliklerinin ise negatif yönde katkı sağladığı gözlemlenmektedir. total_Opnd, v, e ve t özelliklerinin katkı değerlerinin diğerlerine nazaran daha az olduğu görülmektedir. Şekil 3.2.e'ye göre en etkili özelliklerden IOComment, d, i, Loc, uniq_Op özelliklerinin ilk dördünün yüksek özellik değerlerinin pozitif yönde olması nedeniyle tahmine pozitif yönde katkı sağladığı gözlemlenmektedir. t, v ve e özelliklerinin katkı değerinin diğer özelliklere nazaran daha az olduğu görülmektedir.

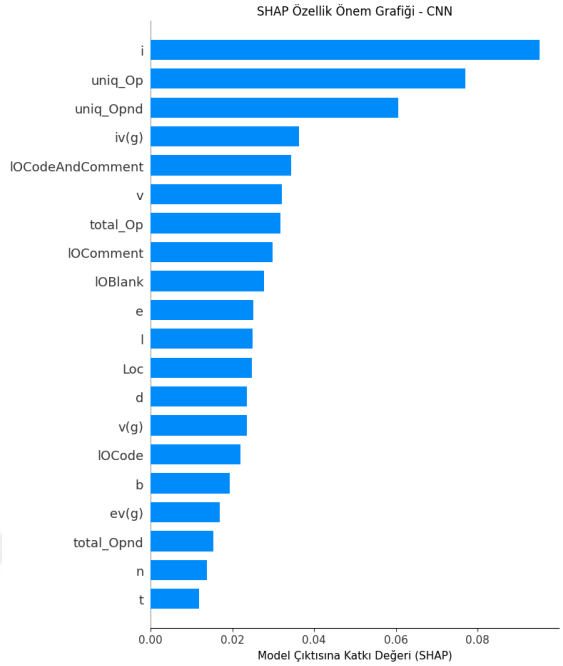
3.3.2. PC1 Veri Seti ile SHAP Yöntemi Kullanımı

- SHAP Özellik Önem Grafikleri

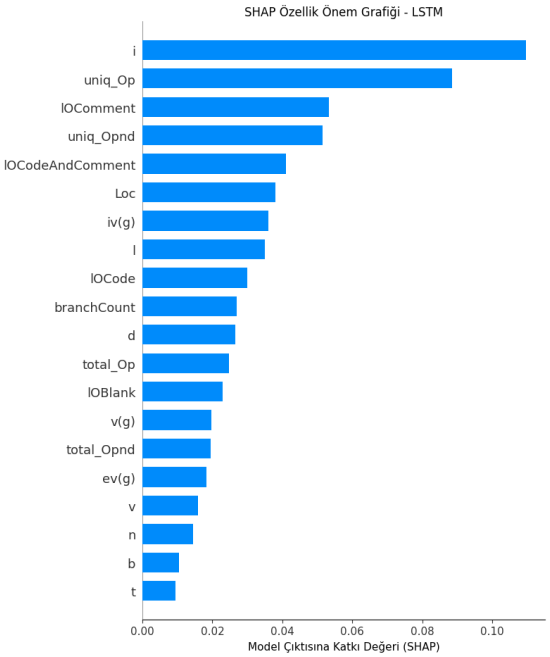


a) RNN tabanlı özellik önem grafiği

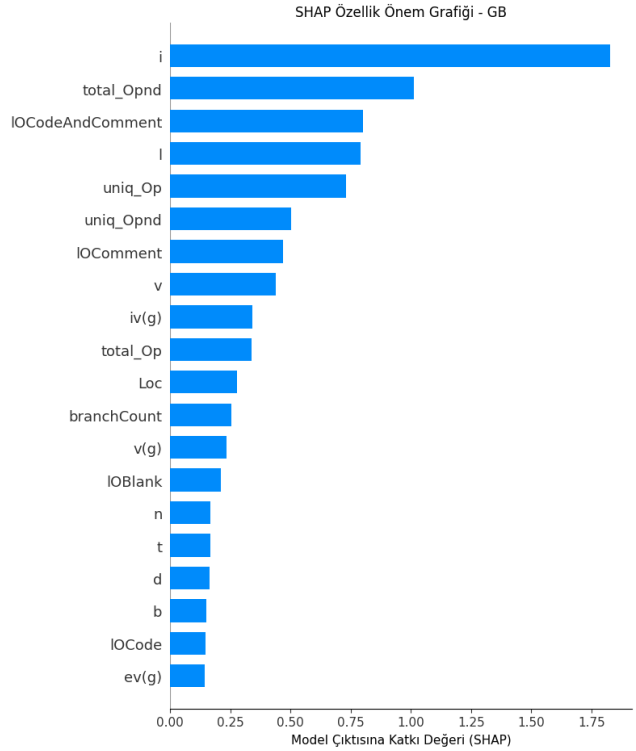
b) MLP tabanlı özellik önem grafiği



c) CNN tabanlı özellik önem grafiği



d) LSTM tabanlı özellik önem grafiği

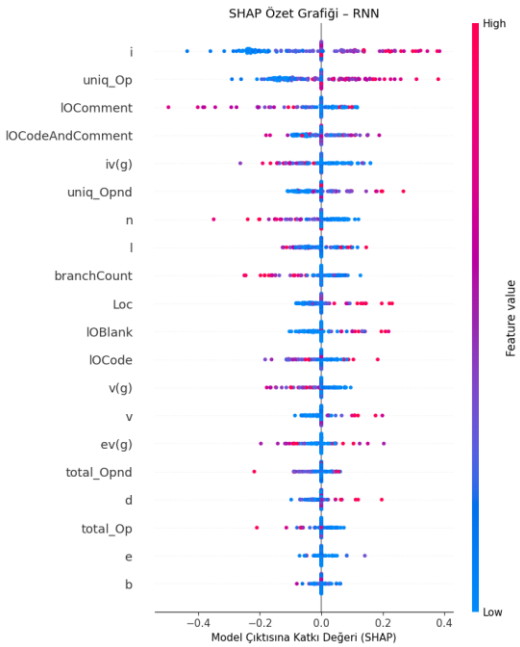


e) GB tabanlı özellik önem grafiği

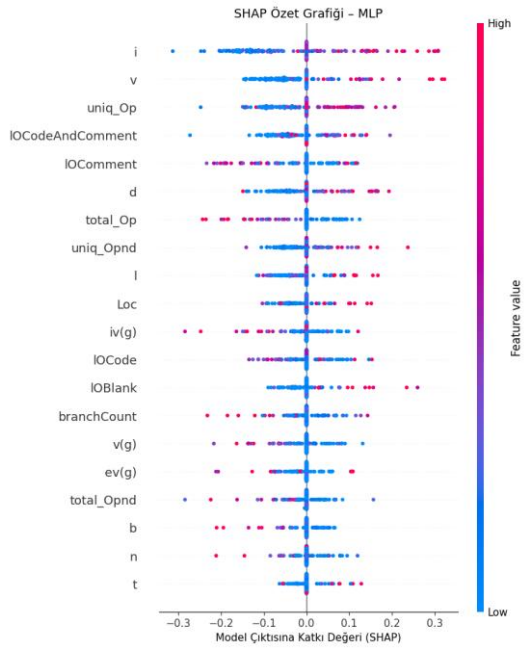
Şekil 3.3. PC1 Veri Seti için SHAP yöntemi kullanılarak hesaplanan özellik önem grafikleri

Şekil 3.3'te küresel açıklama yapabilmek amacıyla PC1 veri setine beş farklı modelin uygulanması ile elde edilen özellik önem grafiklerine ait bulgular bulunmaktadır. Şekil 3.3.a'da RNN modeline göre en yüksek etki değerine sahip özellik i olarak gözlemlenmektedir. i özelliğini takip eden özellikler, uniq_Op, IOComment, lOCodeAndComment, iv(g) şeklinde listelenmektedir. RNN modeline göre yazılım hata tahmininde PC1 veri setinde bulunan b, e ve total_Op özelliklerinin etki değerlerinin diğer özelliklere göre daha az olduğu gözlemlenmektedir. Şekil 3.3.b'de MLP modeline göre i, v, uniq_Op , lOCodeAndComment, ve IOComment özellikleri en etkili beş özellik olarak sıralanmaktadır. Yazılım hata tahmininde PC1 veri setinde bulunan t, n ve b özelliklerinin katkılarının en alt sıralarda yer aldığı gözlemlenmektedir. Şekil 3.3.c'de CNN modeline göre en etkili özellikler i, uniq_Op, uniq_Op, iv(g) ve IOCodeAndComment şeklinde sıralanmaktadır. Etki değeri en az olanlar ise t, n ve total_Opnd özellikleri olarak listelenmektedir. Şekil 3.3.d'de LSTM modeline göre i, uniq_Op, lOComment, uniq_Opnd, IOCodeAndComment özellikleri en etkili özellikler olarak t ve b özelliklerinin ise etki değerlerinin daha az olduğu gözlemlenmektedir. Şekil 3.3.e'de GB modeline göre hata tahmininde en etkili özellikler i, total_Opnd, lOCodeAndComment, l, uniq_Op vb. şeklinde sıralanabilir. ev(g), IOCode, b özelliklerinin etki değerlerinin diğerlerine nazaran daha az olduğu belirtilmektedir.

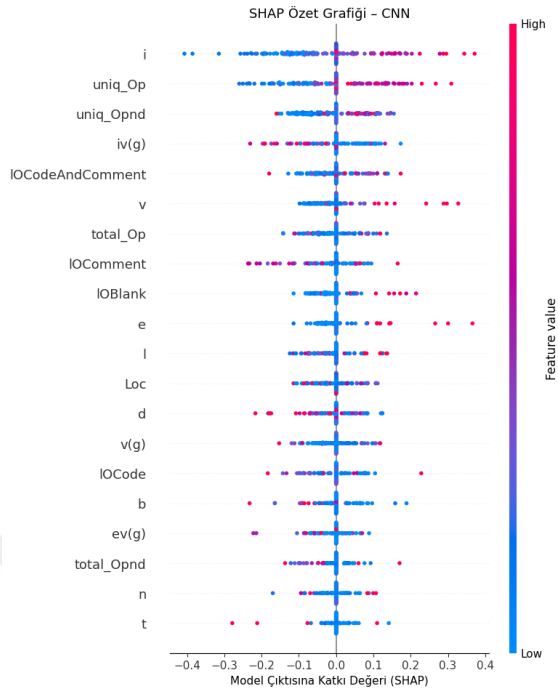
- SHAP Özet Grafikleri



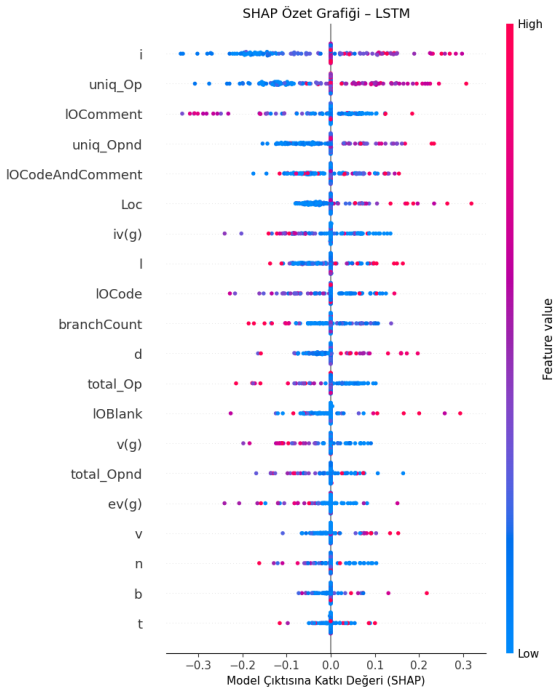
a) RNN tabanlı SHAP Özet Grafiği



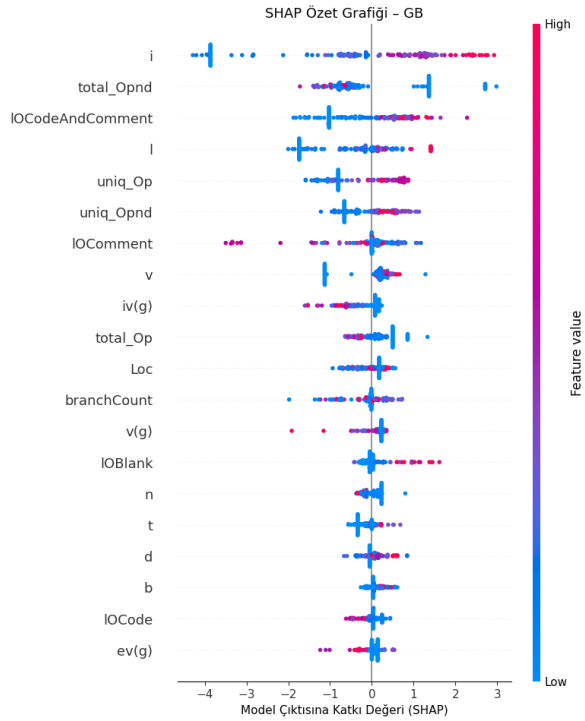
b) MLP tabanlı SHAP Özet Grafiği



c) CNN tabanlı SHAP Özet Grafiği



d) LSTM tabanlı SHAP Özet Grafiği



e) GB tabanlı SHAP Özet Grafiği

Şekil 3.4. PC1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP özet grafikleri.

Şekil 3.4.a'da tahmine en çok etki eden özellikler i, uniq_Op, IOComment, IOCodeAndComment, iv(g) şeklinde sıralanmaktadır. i ve uniq_Op özelliklerinin katkılarının pozitif, IOComment ve iv(g) özelliklerinin ise negatif etkisinin olduğu, b ve e özelliklerinin ise etki değerlerinin diğerlerine nazaran daha az olduğu gözlemlenmektedir. Şekil 3.4.b'de özellikler etki sırasına göre i, v, uniq_Op, lOCodeAndComment, IOComment şeklinde listelenmektedir. Grafikte i, v, uniq_Op, lOCodeAndComment özelliklerinin yüksek özellik değerlerinin pozitif yönde olması pozitif yönde, IOComment özelliğinin ise negatif yönde katkı sağladığı görülmektedir. t, n ve b özelliklerinin ise etki değerlerinin diğer özelliklere nazaran daha az olduğu gözlemlenmektedir. Şekil 3.4.c'de özellikler etki değerlerine göre i, uniq_Op, uniq_Opnd, iv(g), lOCodeAndComment şeklinde listelenmektedir. Bu özelliklerden i, uniq_Op, uniq_Opnd, lOCodeAndComment'in daha belirgin şekilde yüksek özellik değerlerinin pozitif yönde olması nedeniyle tahmine pozitif yönde, iv(g) özelliğinin ise negatif yönde katkı sağladığı, t ve n özelliklerinin ise etki değerlerinin diğer özelliklere göre daha az olduğu görülmektedir. Şekil 3.4.d'ye göre ilk beş özelliği i, uniq_Op, IOComment, uniq_Opnd, IOCodeAndComment şeklinde sıralanmaktadır. i, uniq_Op, uniq_Opnd özelliklerinin daha belirgin şekilde yüksek özellik değerlerinin pozitif yönde olması nedeniyle tahmine pozitif yönde, IOComment özelliğinin ise negatif yönde etkisinin olduğu, t, b, n ve v özelliklerinin ise en az etki eden özellikler olduğu görülmektedir. Şekil 3.4.e'ye göre i, total_Opnd, lOCodeAndComment, l, uniq_Op en etkili özelliklerindendir, i, lOCodeAndComment, l ve uniq_Op özellikleri yüksek özellik değerlerinin pozitif yönde olması nedeniyle tahmine pozitif yönde katkı sağladığı, lOCodeAndComment özelliğinin ise negatif katkı sağladığı gözlemlenmektedir. ev(g), IOCode, b ve d özelliklerinin katkı değerlerlerinin diğer özelliklere nazaran daha az olduğu gözlemlenmektedir.

AS2: CM1 veri setinde Loc, v(g) ve l metriklerinin hata tahmininde belirleyici olduğu; Loc'un hata olasılığını artırdığı, v(g) metriğinin artışıyla hata oranının azaldığı ve program uzunluğunu temsil eden l metriğinin yüksek katkı sunduğu belirlenmiştir. PC1 veri setinde ise i ve uniq_Op metriklerinin hata tahminine pozitif yönde katkı sağlayarak hata olasılığını artıran en baskın değişkenler olduğu görülmüştür.

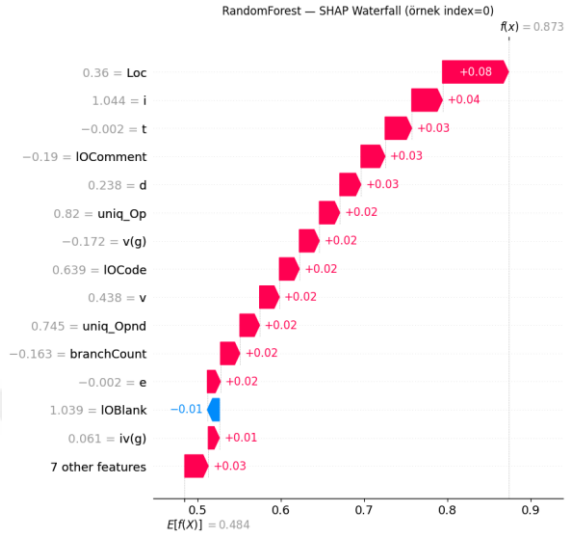
3.4. Yerel Açıklamalar Arasındaki Tutarlılığın Değerlendirilmesi

Şelale grafiği, teknik bilgiye sahip olmayan son kullanıcılara tahminin ardındaki mantığı aktarmak amacıyla, belirlenen bir örnek üzerinden modelin ürettiği tahminin gerekçelerini görsel ve sezgisel biçimde açıklamaktadır. Son kullanıcıya özelliklerin etki değerlerini ve tahmine olan katkılarını sunar. Karmaşık modeller için kullanışlı olup, tahmini daha küçük ve yönetilebilir parçalara ayırarak açıklama sürecini sadeleştirmektedir [12]. Yerel açıklanabilirlik için kullanılan şelale grafiklerinde pozitif (kırmızı) değer alanlar, modele hatalı olma yönünde katkı sağlayan özelliklerdir. Mavi değer alanlar ise modeli hatasız olma yönünde etkilemektedir. Bu değerlerin yanındaki + veya – ile gösterilen rakamsal ifadeler ise etki değerini gösterir. Görsellerdeki E[F(X)], yazılım hata tahmininin ortalama değerini gösterir. Yani E[F(X)] değeri, özelliklerin hiçbirinin etkisi dikkate alınmadığında beklenen model çıktı değeridir. f(x) değeri ise örneğin tahmin değeridir. Ortalama tahmin değeri (E[F(X)]) ile özelliklerin etki değerleri toplandığında örneğin tahmin değeri (f(x)) elde edilir.

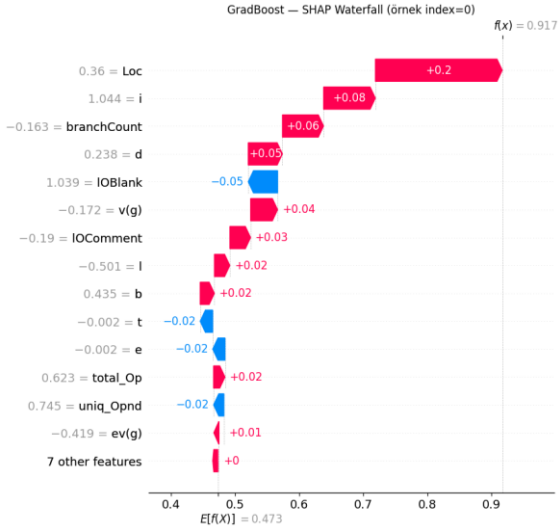
LIME, yerel düzeyde örnek bazlı görsel bulgular sayesinde açıklanabilirlik sunan yapay zekâ yöntemidir [16]. Turuncu renkli listelenen özellikler modele tahminin hatalı olması yönünde katkı sağlar. Mavi renkle belirtilen özellikler ise hata tahmini performansını düşürme yönünde katkı sağlar. Görsellerin sağında bulunan tabloda yerel olarak en etkili ilk on özellik sıralanmış olup, özellikler ve karşılarında özelliklerin örnek için almış olduğu değerler listelenmiştir. Bu değerler, özelliklerin hata tahminine katkısını temsil etmektedir.

3.4.1. CM1 Veri Seti için Yerel Açıklamalar

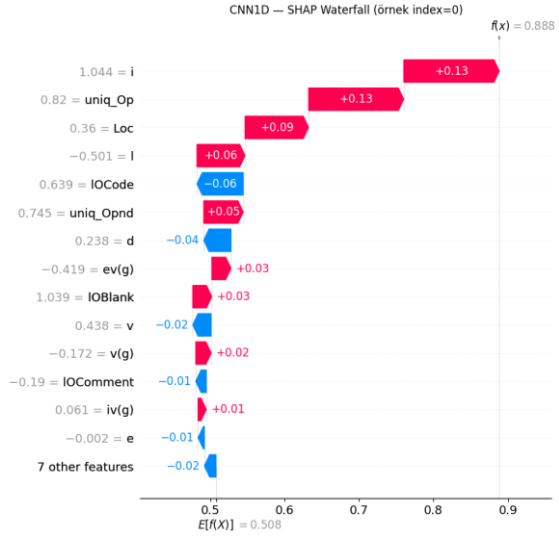
- SHAP Şelale Grafikleri Yöntemi



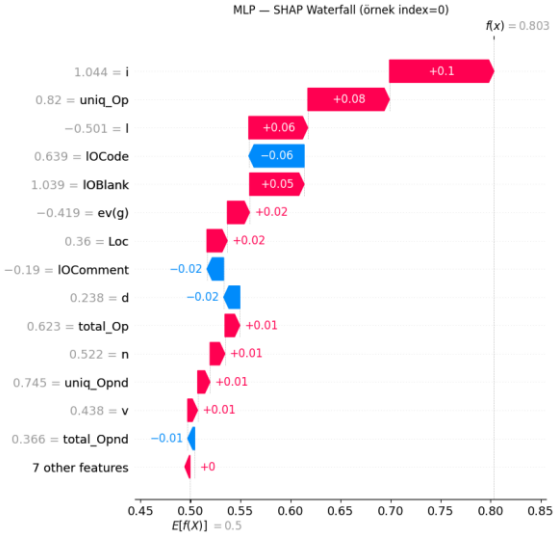
a) RF SHAP Şelale Grafiği



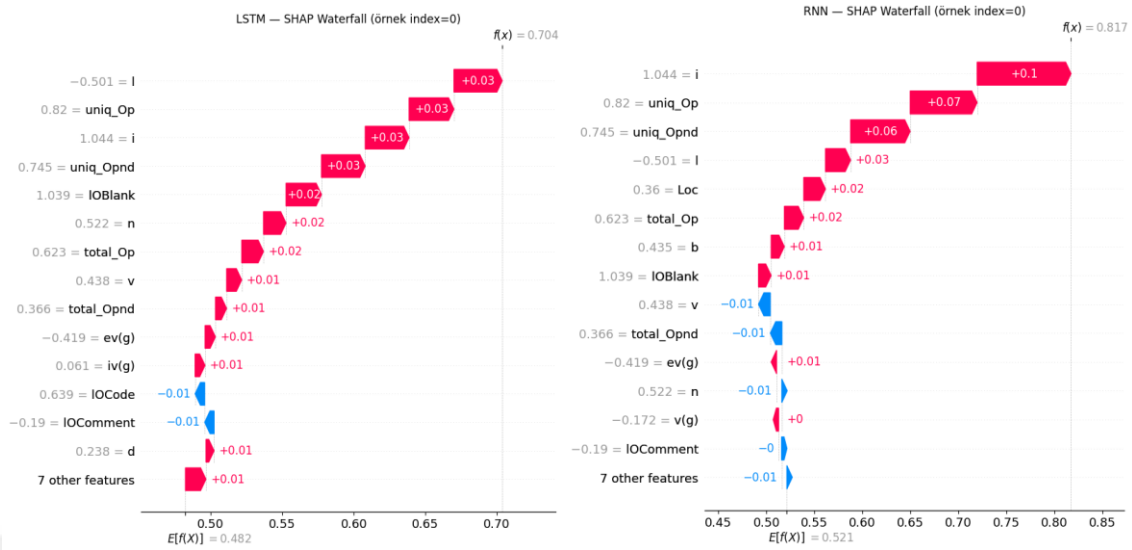
b) GB SHAP Şelale Grafiği



c) CNN SHAP Şelale Grafiği



d) MLP SHAP Şelale Grafiği

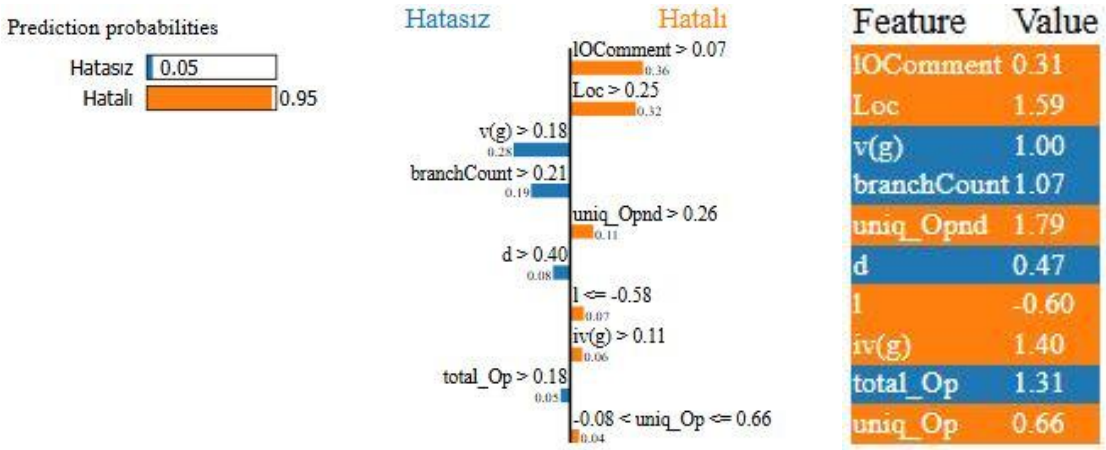


e) LSTM SHAP Şelale Grafiği f) RNN SHAP Şelale Grafiği

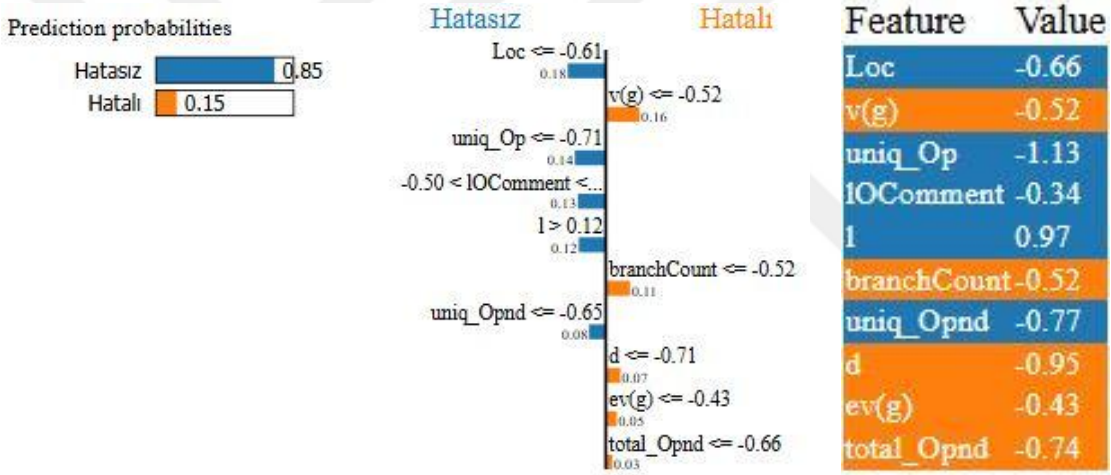
Şekil 3.5. CM1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP şelale grafikleri.

CM1 veri setinin 1. örneği için SHAP şelale grafikleri RF, GB, CNN, MLP, LSTM, RNN modellerine göre gösterimi yapılmıştır. Şekil 3.5.a'da en çok etkisi olan ilk beş özellik; Loc, i, t, IOComment, d olarak listelenir. Bu özelliklerin hepsi hata tahminini artırma yönünde katkıda bulunur. Şekil 3.5.b'de en etkili beş özellik; Loc, i, branchCount, d, IOBlank şeklinde sıralanır. Bu özelliklerden ilk dördü hata tahminini artırma yönünde katkı sağlamıştır fakat IOBlank 0,05 katkı değeri ile hata tahminini azaltma yönünde etki gösterir. Şekil 3.5.c'de, etki sırasına göre ilk beş özellik; i, uniq_Op, Loc, l, IOCode özellikleridir. Bu özelliklerden IOCode tahmini azaltma yönünde katkı sağlarken, diğer dördü, hata tahminini artırma yönünde etki gösterir. Şekil 3.5.d'de tahmine en çok katkı sunan ilk beş özellik; i, uniq_Op, l, IOCode, IOBlank şeklinde listelenir. IOCode 0,06 değeri ile azaltma eğiliminde olup diğer dört özellik tahmini artırma eğilimindedir. Şekil 3.5.e'de l, uniq_Op, i, uniq_Opnd, IOBlank katkı sunan ilk beş özelliktir. Bu beş özellikte hata tahminini artırmaya yönelik katkı sağlar. Şekil 3.5.f'de en etkili ilk beş özellik; i, uniq_Op, uniq_Opnd, l, Loc'tur. Bulgular doğrultusunda, tüm özellikler tahmini artırma yönünde katkı sunar.

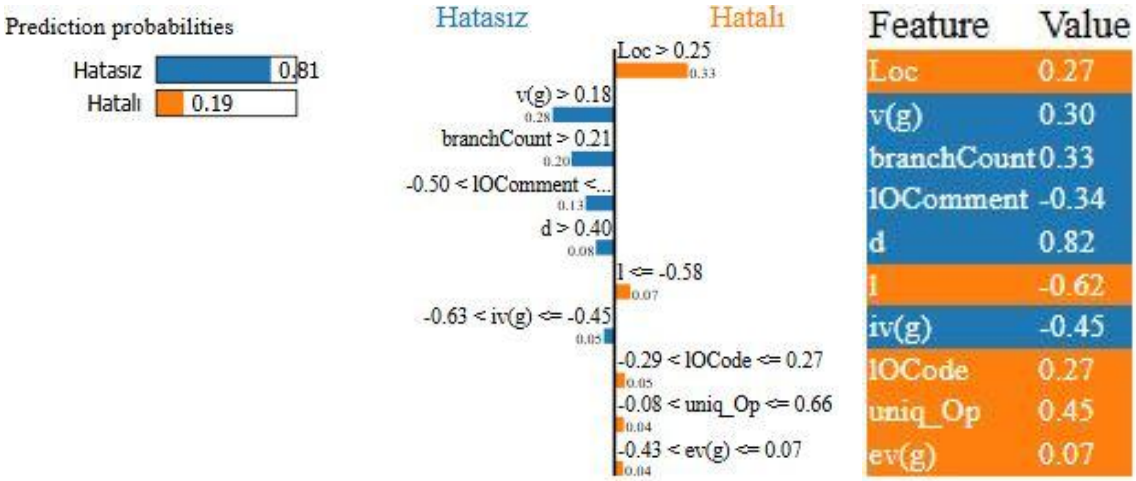
- LIME Yöntemi



Şekil 3.6.a CM1 veri setinin 1. Örneği için LIME Sonuçları



Şekil 3.6.b CM1 veri setinin 5. Örneği için LIME Sonuçları



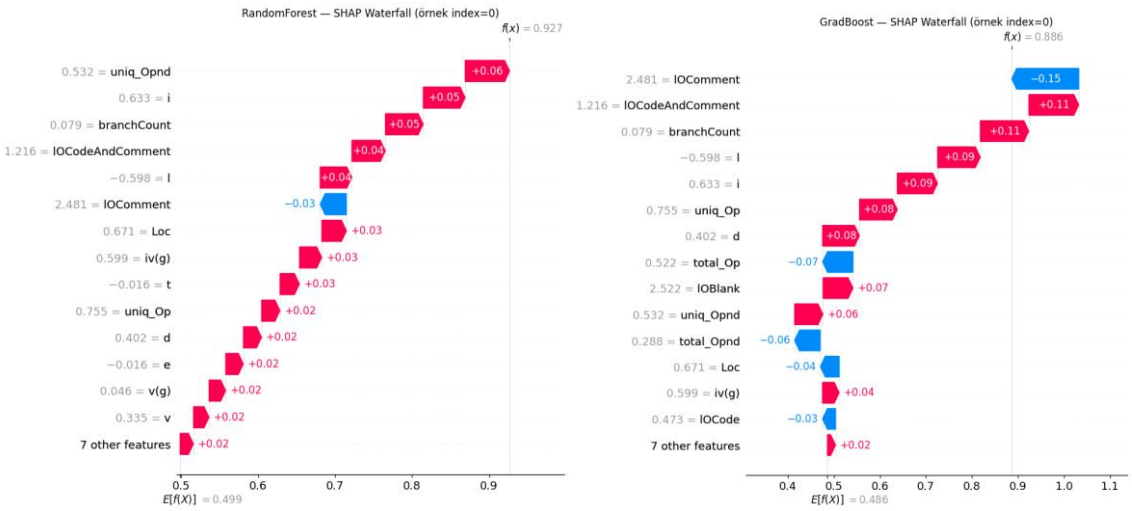
Şekil 3.6.c CM1 veri setinin 15. Örneği için LIME Sonuçları

CM1 veri setinin 3 farklı örneği üzerinde yerel açıklanabilir yapay zekâ yöntemlerinden LIME ile çalışılmıştır. Şekil 3.6.a'da görülen, "IOComment>0,07, Loc>0,25" gibi ifadeler, ilgili koşulun belirli bir örneklem için sağlandığını göstermektedir [18].

Şekil 3.6.a'da veri setin 1. örneği üzerinde çalışılmıştır, görsele göre çıktının hatalı olma olasılığı %95'dir. IOComment özelliği 0,36 puanla en etkili özellik olup hata tahminini artırma yönünde eğilime sahipken, uniq_Op özelliği en etkili ilk on özellikten 0,04 puan ile en az etkiye sahip olanıdır ve hata tahminini artırma yönünde bir eğilimi vardır. Şekil 3.6.b'de ise CM1'in 5. örneği ele alınmıştır. Sonuca göre tahminin hatasız olma olasılığı %85'tir. Loc, 0,18 değeri ile etki değeri en yüksek özellik olup tahminin hatalı olma olasılığını azaltma yönünde eğilim gösterirken, total_Opnd değeri 0,03 değeri ile tahmin performansını artırma yönünde eğilim gösterir. Şekil 3.6.c'de CM1 veri setinin 15. Örneği ele alınmıştır. Görsele göre en etkili özellik 0,33 katkı değeri ile Loc özelliğidir. Loc özelliğinin, tahminin hatalı olma olasılığını artırma yönünde eğilimi vardır. İlk on özellikte en az etkili özellik 0,04 katkı değeri ile tahminin hatalı olma olasılığını artırma yönünde eğilimi olan ev(g) özelliğidir.

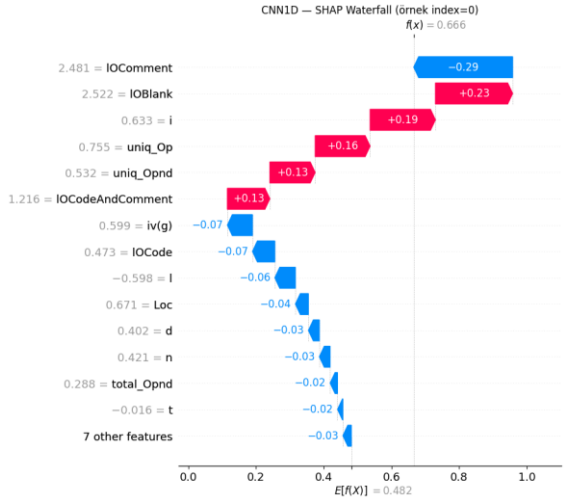
3.4.2. PC1 Veri Seti için Yerel Açıklamalar

- SHAP Şelale Grafikleri Yöntemi

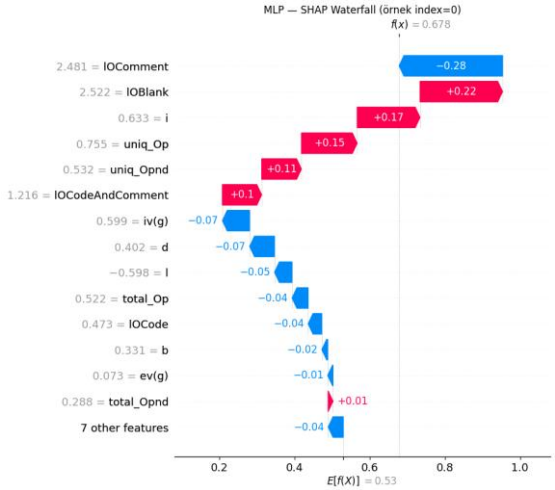


a) RF SHAP Şelale Grafiği

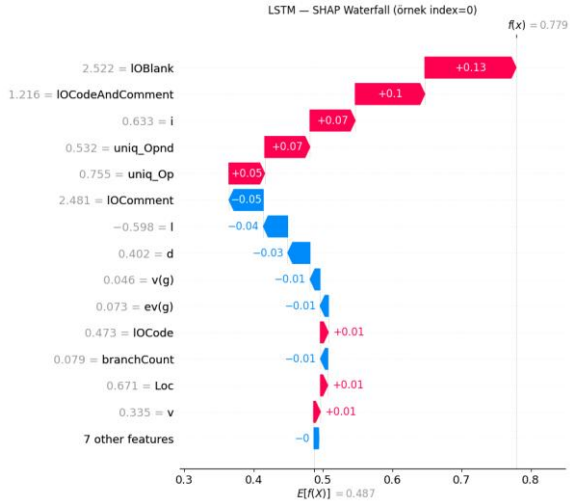
b) GB SHAP Şelale Grafiği



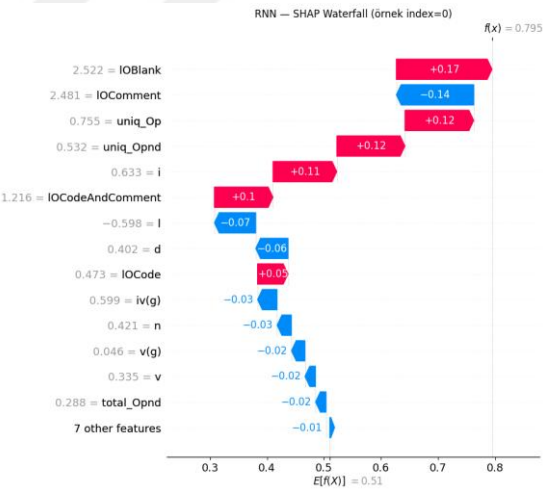
c) CNN SHAP Şelale Grafiği



d) MLP SHAP Şelale Grafiği



e) LSTM SHAP Şelale Grafiği



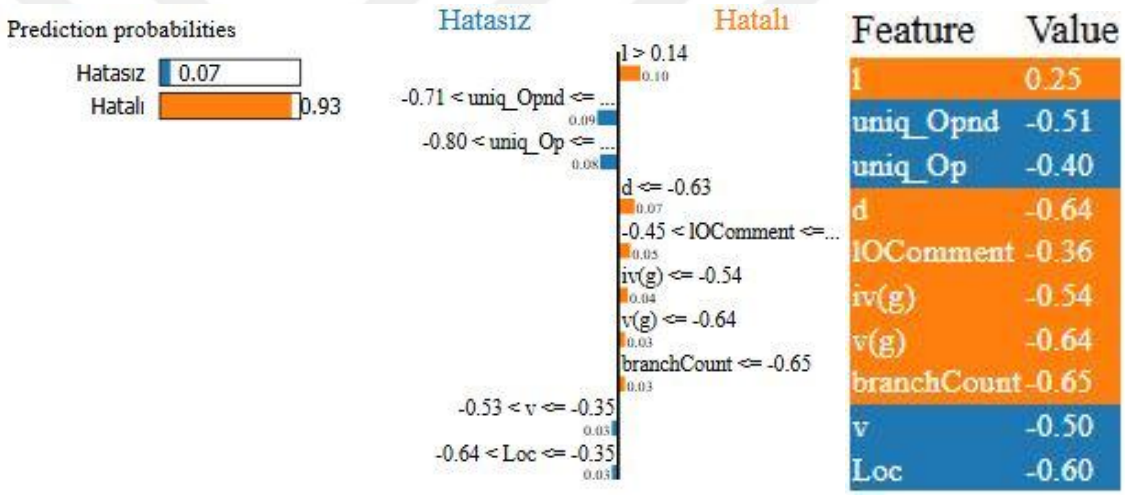
f) RNN SHAP Şelale Grafiği

Şekil 3.7. PC1 veri setine ait, farklı modeller kullanılarak oluşturulan SHAP şelale grafikleri.

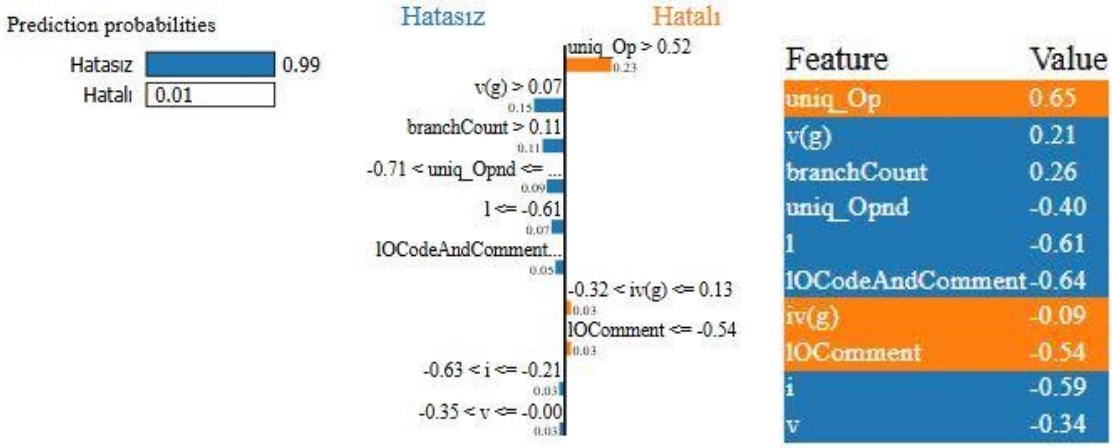
PC1 veri setinin 1. örneği için SHAP Şelale grafikleri RF, GB, CNN, MLP, LSTM, RNN modellerine göre gösterimi yapılmıştır. Şekil 3.7.a'da en çok etkisi olan ilk beş özellik; uniq_Opnd, i, branchCount, lOCodeAndComment, l olarak listelenir. Bu özelliklerin hepsi hata tahminini artırma yönünde katkıda bulunur. Şekil 3.7.b'de en etkili beş özellik; IOComment, lOCodeAndComment, branchCount, l, i şeklinde sıralanır. Bu özelliklerden IOComment 0,15 katkı değeri ile hata tahminini azaltma yönünde etki gösterir, diğer dört özellik hata tahminini artırma yönünde katkı sağlar. Şekil 3.7.c'de, etki sırasına göre ilk beş özellik; IOComment, IOBlank, i, uniq_Op, uniq_Opnd. Bu özelliklerden lOComment 0,29 değeri ile hata tahminini azaltma yönünde katkı sağlarken, diğer tüm özellikler hata

tahminini arttırma yönünde katkı sağlar. Şekil 3.7.d'de tahmine en çok katkı sunan özellikler; IOComment, IOBlank, i, uniq_Op, uniq_Opnd… şeklinde listelenir. Bu özellikler arasında 0,28 etki değeri ile IOComment tahmini azaltma yönünde katkı sunar, diğer dört özellik hata tahminini artırma yönünde katkı sunar. Şekil 3.7.e'de en çok etkisi olan ilk beş özellik; IOBlank, lOCodeAndComment, i, uniq_Opnd, uniq_Op özellikleridir. Tüm özellikler hata tahminini artırma yönünde eğilim gösterir. Şekil 3.7.f'de en etkili ilk beş özellik; IOBlank, IOComment, uniq_Op, uniq_Opnd, i özellikleridir. Bulgular doğrultusunda, IOComment 0,14 katkı değeri ile tahmini azaltma yönünde katkı sunarken, diğer dört özellik tahmini artırma yönünde katkı sunar.

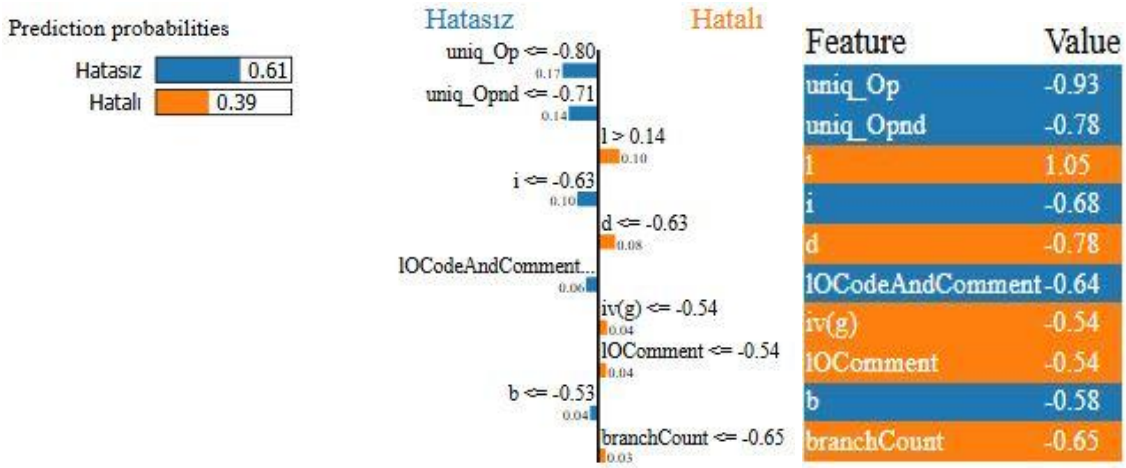
- LIME Yöntemi



Şekil 3.8.a PC1 veri setinin 1. Örneği için LIME Sonuçları



Şekil 3.8.b PC1 veri setinin 15. Örneği için LIME Sonuçları



Şekil 3.8.c PC1 veri setinin 30. Örneği için LIME Sonuçları

Şekil 3.8.a'da veri setin 1. örneği üzerinde çalışılmıştır, görsele göre çıktının hatalı olma olasılığı %93'tür. l özelliği 0,1 puanla en etkili özellik olup hata tahmini performansını artırma yönünde eğilime sahipken, Loc özelliği en etkili ilk on özellikten 0,03 puan ile en az etkiye sahip olanıdır ve hata tahminini azaltma yönünde bir eğilimi vardır. Şekil 3.8.b'de ise PC1'in 15. örneği ele alınmıştır. Sonuca göre tahminin hatasız olma olasılığı %99'dur. uniq_Op, 0,23 değeri ile etki değeri en yüksek özellik olup tahminin hatalı olma olasılığını artırma yönünde eğilim gösterirken, v özelliği, 0,03 puan ile etki değeri en yüksek olan ilk on özellikten sonuncusudur, tahmin performansını azaltma yönünde eğilim gösterir. Şekil 3.8.c'de PC1 veri setinin 30. Örneği ele alınmıştır. Görsele göre en etkili özellik 0,17 katkı değeri ile uniq_Op özelliğidir. uniq_Op özelliğinin, tahminin hatalı olma olasılığını azaltma yönünde eğilimi vardır. İlk on özellikte en az etkili özellik 0,03 katkı değeri ile tahminin hatalı olma olasılığını artırma yönünde eğilimi olan branchCount özelliğidir.

AS3: CM1 ve PC1 veri setlerinde gerçekleştirilen küresel açıklanabilirlik analizlerinde, SHAP özellik önem grafikleri ile SHAP özet (summary) grafiklerinden elde edilen katkı sıralamalarının büyük ölçüde tutarlı olduğu gözlemlenmiştir. Yerel açıklanabilirlik analizlerinde ise her iki veri seti için birinci örnek üzerinden değerlendirme yapılmıştır. Buna göre, CM1 veri setinde RF, GB ve MLP modellerinde 6/10, PC1 veri setinde ise RF modelinde 7/10 oranında en yüksek özdeşleşme düzeyi tespit edilmiştir.

3.5. Özellik Azaltımı Uygulamasının Model Performansına Etkisi

Modellerin performansı karşılaştırılırken beş farklı metriğin birlikte kullanılması hem performansın farklı boyutlarını yakalamayı hem de ağaç tabanlı ve derin öğrenme modellerinin hata tahmini davranışlarını bütüncül bir biçimde değerlendirmeyi mümkün kılmıştır. Böylece model başarımı yalnızca tek bir göstergeye göre değil, çok boyutlu bir değerlendirme çerçevesiyle ele alınmıştır. Daha sonra tahmin performansına görece düşük katkı sunduğu ve model çıktılarında gürültü etkisi oluşturabileceği öngörülen özelliklerin elimine edilmesine yönelik olarak gerçekleştirilen özellik azaltımı ile performans ölçümü tekrar yapılmış olup tahmin performansının, bulguların genelinde arttığı gözlemlenmiştir.

Tablo 3.3. CM1 Metriklerinin Sonuçları

Model	**Veri Set**	**Doğruluk**	**Kesinlik**	**Duyarlılık**	**F1-Puanı**	**AUC-ROC**
RF	Tümü	0.925439	0.879557	0.986124	0.929543	0.986227
	Azaltılmış	**0.93009**	**0.886761**	**0.986124**	**0.933646**	**0.984955**
GB	Tümü	0.918435	0.874267	0.976383	0.922427	0.969353
	Azaltılmış	**0.924283**	**0.879695**	**0.980934**	**0.927345**	**0.970615**
MLP	Tümü	0.921964	0.899157	0.951569	0.923834	0.962024
	Azaltılmış	**0.924303**	**0.901186**	**0.953401**	**0.925778**	**0.963908**
LSTM	Tümü	0.879988	0.852017	0.918105	0.883661	0.9311
	Azaltılmış	**0.886951**	**0.861086**	**0.923332**	**0.891004**	**0.939039**
RNN	Tümü	0.878777	0.859305	0.906313	0.882063	0.939396
	Azaltılmış	**0.871835**	**0.836084**	**0.922335**	**0.877019**	**0.932079**
CNN	Tümü	0.885815	0.867448	0.908683	0.887351	0.940787
	Azaltılmış	**0.884632**	**0.870628**	**0.901938**	**0.885768**	**0.942671**

Tablo 3.3'te CM1 veri setinin tüm özellikleri ile v, e ve total_Opnd özelliklerinin çıkarılmış hali için elde edilen sonuçları sunmaktadır. Bu özelliklerin çıkarılması, metriklerin genelinde tahmin performansını artırmıştır.

Tablo 3.4. PC1 Metriklerinin Sonuçları

Model	**Veri Set**	**Doğruluk**	**Kesinlik**	**Duyarlılık**	**F1-Puanı**	**AUC-ROC**
RF	Tümü	0.970422	0.955232	0.987528	0.971012	0.995321
	Azaltılmış	**0.972361**	**0.959658**	**0.986635**	**0.972887**	**0.995317**
GB	Tümü	0.966062	0.949584	0.984453	0.966647	0.993839
	Azaltılmış	**0.966062**	**0.949725**	**0.984570**	**0.966772**	**0.993059**
MLP	Tümü	0.950548	0.932623	0.971833	0.951586	0.981233
	Azaltılmış	**0.954911**	**0.937641**	**0.974718**	**0.955692**	**0.979061**
LSTM	Tümü	0.921446	0.900603	0.947754	0.923371	0.974357
	Azaltılmış	**0.933088**	**0.913976**	**0.956520**	**0.934486**	**0.972975**
RNN	Tümü	0.916591	0.89434	0.94475	0.918571	0.970013
	Azaltılmış	**0.926781**	**0.908707**	**0.948438**	**0.928044**	**0.971052**
CNN	Tümü	0.939866	0.92154	0.961984	0.940815	0.978579
	Azaltılmış	**0.948117**	**0.932638**	**0.965934**	**0.948837**	**0.984128**

Tablo 3.4'te, PC1 veri setinin tüm özellikleri ile b ve t özelliklerinin çıkarılmış hali için elde edilen sonuçları sunmaktadır. Bu özelliklerin çıkarılması, metriklerin genelinde tahmin performansını artırmıştır.

AS4: CM1 veri setinde v, e ve total_Opnd; PC1 veri setinde ise b ve t özelliklerinin çıkarılmasının, metriklerin genelinde tahmin performansını artırdığını ortaya koymuştur.

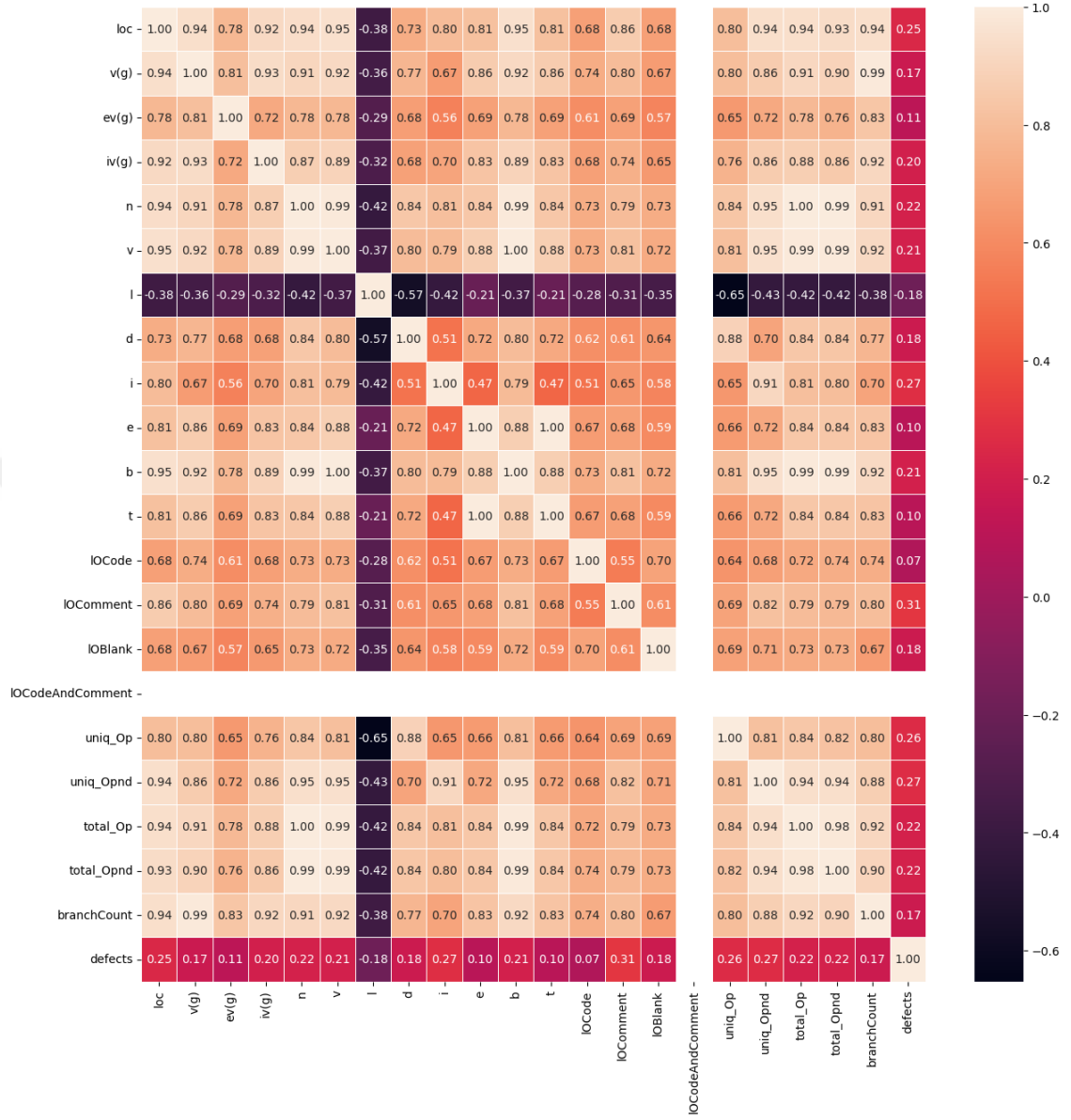
3.6. Korelasyon Tabanlı İlişki Analizi (Pearson Korelasyonu)

Korelasyon, rastgele iki değişken arasındaki ilişkinin yönünü ve derecesini ifade eder. Bu kavram, iki değişkenin karşılıklı değişimlerini ortaya koyar ve aralarındaki bağlantının değerlendirilmesine olanak tanır. Ancak korelasyon analizi, neden – sonuç ilişkisini doğrudan açıklamaz, yalnızca değişkenler arasındaki bağlantının yönünü ve derecesini ortaya koyabilir. Yüksek korelasyonlar dahi nedensel bir bağın göstergesi olmayabilir. Korelasyon katsayısı r, -1 ile 1 arasında değer alan bir istatiksel ölçüt olup; -1 tam ters yönlü, 0 ilişkisiz ve 1 ise tam doğrusal yönlü bağlantıyı göstermektedir [83].

CM1 ve PC1 veri setleri üzerinde gerçekleştirilen açıklanabilirlik analizleri, her iki veri setin de güçlü çoklu bağlantılar (multicollinearity), yoğun metrik kümelenmeleri ve belirgin doğrusal olmayan ilişki yapıları içerdiğini göstermektedir.

Isı haritası, makine öğrenmesi modellerinde yorumlanabilirlik ve açıklanabilirlik için kullanılan bir yöntemdir [9]. Korelasyonların önemini ve yönünü temsil etmek için renk yoğunluğunu kullanır.

CM1 Veri Seti



Şekil 3.9. CM1 veri setinin ısı haritası

Şekil 3.9'da CM1 veri setinin korelasyon ısı haritası, özelliklerinin büyük çoğunluğunun birbirleriyle yüksek ve pozitif ilişkiler içinde olduğunu göstermektedir ($r \approx 0{,}65$–$0{,}99$). Bu karmaşık korelasyon ısı haritası, CM1'in doğrusal modeller tarafından kolayca temsil edilemeyecek ölçüde yoğun bağımlılıklar içerdiğini ortaya koymaktadır. Loc ile karmaşıklık özellikleri (v(g), n, v, b) arasında $r \approx 0{,}94$–$0{,}95$ seviyesinde korelasyonlar, benzer biçimde uniq_Op, uniq_Opnd, total_Op, total_Opnd özellikleri arasında $r \approx 0{,}81$–$0{,}98$ düzeyinde yoğun kümelenmeler gözlenmiştir. Bu bulgu, SDP için PROMISE veri setlerinde özellikler arasında yapısal bağımlılıkların ve literatürde yüksek düzeyde çoklu

bağlantının yaygın olduğunu bildiren çalışmalarla tutarlılık göstermektedir [72], [84], [85]. Yüksek korelasyonun SHAP gibi XAI yöntemlerinde etki sırasını ve yönünü etkilediği ve modellerin doğrusal ilişkilerden sapmasına yol açtığı belirtilmektedir [86], [87]. Bu korelasyon seviyeleri SDP literatüründe yüksek çoklu doğrusallık (multicollinearity) göstergesi olarak değerlendirilmektedir.

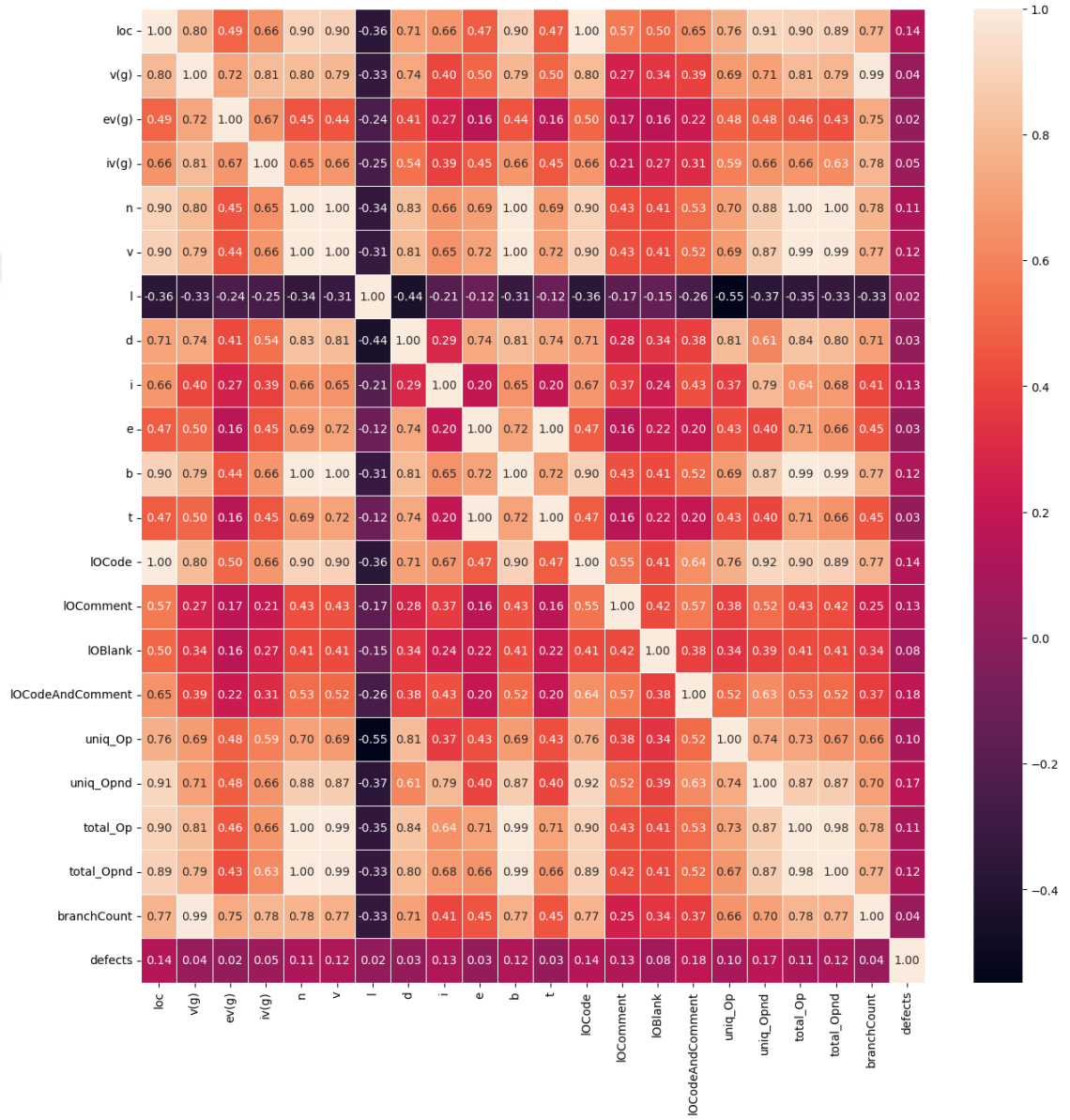
Tablo 3.5. CM1 veri seti için korelasyon sonuçları

	Pearson_Korelasyon	**P_Değeri**
Uniq_Op	0.360651	<0.001
i	0.346535	<0.001
uniq_Opnd	0.320521	<0.001
Loc	0.305652	<0.001
lOComment	0.303314	<0.001
n	0.275244	<0.001
total_Op	0.273631	<0.001
total_Opnd	0.273337	<0.001
b	0.258340	<0.001
v	0.258291	<0.001
d	0.249635	<0.001
lOBlank	0.234039	<0.001
iv(g)	0.226919	<0.001
branchCount	0.190019	<0.001
v(g)	0.177318	<0.001
e	0.127923	<0.001
t	0.127923	<0.001
ev(g)	0.085197	<0.05
lOCode	0.068458	<0.05
l	-0.36478	<0.001
lOCodeAndComment	NaN	NaN

CM1 veri setinin Pearson Korelasyonu Tablo 3.5'te, uniq_Op, i, uniq_Opnd, Loc ve lOComment gibi özelliklerin hata oluşumu ile en güçlü doğrusal ilişkiye sahip olduğunu göstermiştir ($p<0.001$). Buna karşılık SHAP açıklamaları incelendiğinde, özellikle derin öğrenme tabanlı modellerin (MLP, RNN, LSTM, CNN) bu doğrusal ilişkileri tam olarak takip etmediği, bazı modellerde korelasyon açısından önemi düşük özelliklerin görece katkı değerinin yüksek olduğu, buna karşın korelasyon açısından üst sıradaki özelliklerin bazı modellerde geriye düştüğü görülmüştür. SHAP Summary Importance grafiklerinde v(g), l, Loc, i ve IOComment özelliklerinin model türüne göre değişen şekilde öne çıktığı gözlemlenmiştir. Bu bulgular neticesinde; derin öğrenme modellerinin SDP'de doğrusal ilişkileri değil, zaman içinde oluşan karmaşık, doğrusal olmayan ve çok boyutlu yapısal

örüntüleri yakaladığı görülmüştür [86], bu nedenle derin öğrenme modelleriyle eğitilen veri setlerin SHAP önem sıralamalarının, Pearson korelasyonlarıyla tam olarak hizalanmaması literatürle uyumlu bir durumdur [67].

PC1 Veri Seti



Şekil 3.10. PC1 veri setinin ısı haritası

Şekil 3.10'da PC1 veri setine ait korelasyon ısı haritası bulunmaktadır. Isı haritası, özelliklerin büyük çoğunluğunun birbirleriyle son derece yüksek düzeyde korelasyon gösterdiğini ortaya koymaktadır ($r \approx 0{,}70$–$0{,}99$). CM1'e benzer biçimde, kod boyutu, karmaşıklık metrikleri ve Halstead ölçümleri PC1'de de birbirine sıkı şekilde bağlıdır;

ancak PC1'de özellikle uniq_Op, uniq_Opnd, total_Op, total_Opnd ve operand/operatör bazlı metriklerin Loc ve v(g) ile daha yoğun bir kümelenme sergilediği görülmektedir. Bu yapı PC1'in de doğrusal istatistiksel ilişkiler ile kolayca çözülemeyecek derecede çoklu bağlantı içerdiğini göstermektedir.

Tablo 3.6. PC1 veri seti için korelasyon sonuçları

Özellik	**Pearson_Korelasyon**	**P_Değeri**
lOCodeAndComment	0.273675	<0.001
uniq_Opnd	0.266377	<0.001
i	0.251114	<0.001
Loc	0.213374	<0.001
lOCode	0.205899	<0.001
lOComment	0.197019	<0.001
v	0.179443	<0.001
b	0.178858	<0.001
total_Opnd	0.174204	<0.001
n	0.174161	<0.001
total_Op	0.171962	<0.001
uniq_Op	0.164237	<0.001
lOBlank	0.114920	<0.001
iv(g)	0.087245	<0.001
l	0.069624	<0.01
v(g)	0.065849	<0.01
branchCount	0.065227	<0.01
e	0.055626	<0.05
t	0.055626	<0.05
d	0.041336	0.060562
ev(g)	0.008245	0.708272

PC1 veri setinin Pearson korelasyonu Tablo 3.6'de lOCodeAndComment, uniq_Opnd, i, Loc ve lOCode özelliklerinin hata değişkeni ile anlamlı biçimde ilişkili olduğunu göstermektedir ($p<0.001$). Buna karşın SHAP analizlerinde derin öğrenme modelleri i, uniq_Op, lOComment, lOCodeAndComment gibi değişkenlere ağırlık verir. CNN ve LSTM modellerinde uniq_Op / uniq_Opnd çiftinin yüksek SHAP önemine sahip olması, PC1 veri setinde operand / operatör seviyesindeki yapısal karmaşıklığın derin modeller tarafından doğrusal ilişkilerden daha güçlü biçimde yakalandığını göstermektedir.

Genel olarak CM1 ve PC1 veri setlerinin karşılaştırılması, yazılım hata tahmininde açıklanabilirliğin yalnızca öznitelik önemlerinin sıralanması ile değil, aynı zamanda bu önemlerin veri içi istatistiksel yapıyla ne ölçüde uyum sağladığının değerlendirilmesi

gerektiğini göstermektedir. Her iki veri setinde de SHAP ile Pearson sonuçlarının birebir örtüşmemesi, modellerin doğrusal ilişkilerin ötesine geçen daha karmaşık örüntüler öğrenebildiğini ve bu nedenle açıklanabilirliğin yalnızca korelasyon temelli yöntemlerle sınırlandırılmasının yeterli olmayabileceğini göstermektedir [16], [74]. Böylece CM1 ve PC1 birlikte değerlendirildiğinde, SDP modellerinin açıklanabilirlik çıktılarının veri setlerinin yapısal özellikleriyle önemli ölçüde şekillendiği ve derin modellerin korelasyonla sezgisel olarak açıklanamayan ilişkileri yakalama eğiliminde olduğu sonucuna ulaşılmaktadır.

4. BÖLÜM

TARTIŞMA, SONUÇ VE ÖNERİLER

4.1. Tartışma, Sonuç ve Öneriler

Yazılım hatalarının erken tespiti, sadece yazılım hatalarının azaltılmasını sağlamaz, bunun yanı sıra yazılımın uzun vadeli kalite sürekliliğini destekler. Bu çalışmada, SDP performansını iyileştirebilmek ve bu tahminin kararını yorumlanabilir ve açıklanabilir kılmak amacıyla derin öğrenme ve makine öğrenmesi modelleri kullanılmıştır. NASA PROMISE veri tabanında yer alan CM1 ve PC1 veri setleri üzerinde, özellik ölçekleme, aşırı örnekleme gibi veri ön işleme ve hazırlık işlemleri yapılmıştır. GB, RF, MLP, CNN, RNN, LSTM modellerine doğruluk, kesinlik, duyarlılık, F1-Puanı, AUC-ROC metrikleriyle hata tahmini performansı değerlendirilmiştir.

Geleneksel yazılım mühendisliği yaklaşımlarına ek olarak açıklanabilir yapay zekâ yöntemlerinin sürece entegre edilmesi, yalnızca hataların öngörülmesini değil, aynı zamanda bu hataların ortaya çıkış nedenlerinin anlaşılmasını sağlayarak geliştiricilerin daha bilinçli ve temellendirilmiş kararlar almasına katkıda bulunmaktadır. Çalışmamızda yerel açıklanabilirlik için LIME ve SHAP şelale grafikleri yöntemi, küresel açıklanabilirlik için ise SHAP özet grafikleri ve SHAP özellik önem grafikleri yöntemleri kullanılmıştır.

AS1 kapsamında elde edilen sonuçlar, CM1 ve PC1 veri setlerinde farklı modellerin öne çıktığını göstermektedir. CM1 veri setinde doğruluk, kesinlik ve F1-Puanı açısından MLP modelinin, duyarlılık ve AUC-ROC açısından ise RF modelinin daha başarılı olduğu görülmüştür. Buna karşılık PC1 veri setinde, performans metriklerinin büyük çoğunluğunda RF modeli öne çıkarken, duyarlılık açısından GB modelinin üstünlük sağladığı belirlenmiştir. Bu farklılığın temel nedenlerinden biri, CM1 ve PC1 veri setlerinin özellik dağılımları, boyutları ve hata sınıfı dengesizliklerinin birbirinden farklı

olmasıdır. Ağaç tabanlı modellerin PC1 veri setinde daha tutarlı sonuçlar üretmesi, bu veri setindeki özelliklerin doğrusal olmayan ayrımlara ve etkileşimlere daha elverişli bir yapı sunduğunu düşündürmektedir. CM1 veri setinde MLP'nin bazı metriklerde öne çıkması ise, ağ tabanlı modellerin çok boyutlu ve karmaşık örüntüleri yakalama kapasitesinin bu veri setinde daha etkin biçimde devreye girdiğini göstermektedir.

Bu bulgular, yazılım hata tahmininde model seçiminin veri setine özgü olarak yapılması gerektiğini ve tek bir modelin tüm veri setleri için genellenebilir üstünlük sunmadığını ortaya koymaktadır.

AS2 kapsamında yapılan küresel açıklanabilirlik analizleri, her iki veri setinde hata tahminini belirleyen özelliklerin farklılaştığını göstermiştir. CM1 veri setinde Loc, v(g) ve l özelliklerinin öne çıkması, kod büyüklüğü ve yapısal karmaşıklığın hata oluşumuyla doğrudan ilişkili olduğunu düşündürmektedir. Özellikle Loc değişkeninin hata oranını artırıcı yönde etkisi, literatürde sıklıkla vurgulanan "büyük kod, daha fazla hata riski" yaklaşımıyla örtüşmektedir.

Buna karşılık v(g) değişkeninin artışıyla hata oranının azalması, ilk bakışta sezgisel olmayan bir sonuç gibi görünse de bu durumun iyi yapılandırılmış ve modüler kodların daha fazla karar noktası içermesine rağmen daha kontrollü geliştirildiği projeleri temsil ediyor olabileceği değerlendirilmektedir. Program uzunluğunu temsil eden l değişkeninin yüksek katkısı ise, kod karmaşıklığının yalnızca satır sayısıyla değil, yapısal uzunlukla da ilişkili olduğunu göstermektedir.

PC1 veri setinde ise i ve uniq_Op özelliklerinin baskın hale gelmesi, algoritmik karmaşıklık ve kullanılan operatör çeşitliliğinin hata üretme potansiyelini artırdığını ortaya koymaktadır. Bu durum, PC1 veri setinin daha algoritma odaklı ve işlem yoğun modüller içermesiyle ilişkilendirilebilir.

Bu bulgular, hata tahmininde etkili olan özelliklerin veri setine özgü olduğunu ve tek tip bir özellik seçimi yaklaşımının genellenebilir olmadığını göstermektedir.

AS3 kapsamında gerçekleştirilen yerel açıklama analizleri, özellikle CM1 veri setinin 1. örneği üzerinden yapılan değerlendirmelerde, SHAP ve LIME yöntemlerinin bazı modellerde yüksek düzeyde örtüştüğünü göstermiştir. RF, GB ve MLP modellerinde ilk 10 özellikten 6'sının ortaklaşması, PC1 veri setinde RF modelinde bu örtüşme oranı 10

özellikten 7'dir. Bu durum modellerin karar mekanizmalarının daha kararlı ve yorumlanabilir olduğunu düşündürmüştür. Küresel açıklanabilirlik için ise SHAP özet grafikleri ve SHAP özellik önem grafikleri yöntemlerinin kullanılan tüm modeller için karşılaştırılmasıyla elde edilen bulgulara göre; CM1 veri seti için Loc, v(g), l'nin, PC1 veri seti için ise i, uniq_Op'in en etkili özellikler arasında listelendiği belirlenmiştir. Bu yöntemler için bulguların büyük ölçüde özdeşleşmesi açıklamaların tutarlılığını ve güvenilirliğini artıran önemli bir ampirik dayanak oluşturmuştur. CM1 veri seti için en az etkiye sahip olup gürültü oluşturduğu düşünülen v, e ve total_Opnd özellikleri ve PC1 veri seti için en az etkiye sahip olan özelliklerden b ve t silinip modellerin metrikler üzerindeki performansı karşılaştırılmıştır ve sonuçların çoğunluğunda iyileşme olduğu görülmüştür. Açıklanabilir yapay zekâ sayesinde bu tür bulgulara ulaşmak, yalnızca tahmin yapmakla kalmayıp, hataların ardındaki yapısal nedenleri ortaya koyarak geliştiricilerin hedef odaklı müdahalelerde bulunmasını mümkün kılmaktadır [16], [21]. Bu durum, proaktif hata önleme stratejilerinin geliştirilmesine katkı sunarken, yazılım yaşam döngüsü boyunca kalitenin korunmasına da yardımcı olmaktadır.

Bu bulgular neticesinde, SHAP tabanlı küresel açıklamaların model içi tutarlılık sağladığı, ancak yerel açıklamalarda model türüne bağlı farklılıkların ortaya çıktığı görülmüştür. Bu durum, yazılım hata tahmininde XAI yöntemlerinin tamamlayıcı olarak kullanılması gerektiğini göstermiştir.

AS4 kapsamında CM1 veri setinden v, e ve total_Opnd özelliklerinin çıkarılması, PC1 veri setinden b ve t özelliklerinin çıkarılmasıyla performans metriklerinin genelinde tahmin başarımında artış görülmektedir. Bu bulgu CM1 ve PC1 veri setlerinde hata tahminine sınırlı katkı sağlayan veya gürültü oluşturan özelliklerin çıkarılmasının, modellerin daha ayırt edici örüntülere odaklanmasına olanak tanıdığını ortaya koymaktadır.

CM1 ve PC1 veri setlerinin korelasyon ısı haritası incelendiğinde özelliklerin birbiriyle yüksek ilişkili olduğu gözükmektedir, bu durum özellikler arası çoklu bağlantı olduğunu gösteren bir durumdur. İki veri setinde Pearson Korelasyon tablosunda hata tahmini ile en güçlü doğrusal ilişkiye sahip özellikler yukarıdan aşağıya doğru sıralanmıştır. Derin öğrenme tabanlı modellerin bu doğrusal ilişkileri tam olarak takip etmediği, zaman içinde oluşan karmaşık, doğrusal olmayan ve çok boyutlu yapısal örüntüleri yakaladığı

görülmüştür. Bunun sonucunda SHAP bulguları ile Pearson korelasyonunun sıralamasının birbiriyle tam olarak örtüşmemesi beklenen bir durumdur [67].

Elde edilen sonuçlar, yazılım hata tahmininde yüksek başarı sağlayan derin öğrenme modellerinin açıklanabilirlik yöntemleriyle birlikte kullanılmasının literatüre katkı sağlayabilecek etkili bir yaklaşım olduğunu ortaya koymaktadır. İki farklı veri setin kullanılması elde edilen bulguların, farklı veri koşullarında da geçerliliğini koruyup korumadığının değerlendirilmesine fırsat sunarak, sonuçların genellenebilirliğini güçlendirmektedir. Gelecekteki çalışmaların, daha geniş ölçekli ve farklı yazılım projelerinden türetilmiş veri setlerini kapsayacak şekilde tasarlanması, önerilen yaklaşımın çeşitli kurumsal ve teknik bağlamlarda ne ölçüde geçerli olduğunu sınama açısından önemli olacaktır. Böylece, açıklanabilir yapay zekâ yöntemlerinin farklı veri koşullarında nasıl davrandığına dair daha derinlemesine içgörüler elde edilebilir ve bu yöntemlerin yazılım hata tahmini süreçlerinde daha güvenilir ve genellenebilir şekilde kullanılmasına katkı sunulabilir. Bir diğer öneri ise; Tam zamanlı yazılım hata tahmin modellerinin açıklanabilirlik yöntemleriyle entegre edilmesi, geliştiricilere kodlama anında bağlamsal ve yorumlanabilir risk geri bildirimi sunan bir karar destek mekanizmasının geliştirilmesine olanak sağlayacaktır.

KAYNAKÇA

1. Keskinkılıç, M., Özmen, E., 2018. Yazılım projelerinde yazılım geliştiricilerin yazılım süreç modellerini kullanım farkındalıkları. **Akademi Sosyal Bilimler Dergisi**, **5**(15): 61–78.
2. Bommi, N. S., Negi, A., 2023. A standard baseline for software defect prediction: Using machine learning and explainable AI, 1798–1803. *IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, June 2023, Torino, Italy, IEEE, 6 s.
3. Farid, A. B., Fathy, E. M., Eldin, A. S., Abd Elmegid, L. A., 2021. Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short term memory (Bi-LSTM). **PeerJ Computer Science**, **7**: e739.
4. Jiarpakdee, J., Tantithamthavorn, C., Dam, H. K., Grundy, J., 2020. An empirical study of model-agnostic techniques for defect prediction models. **IEEE Transactions on Software Engineering**, **48**(1): 166–185.
5. Wong, W. E., Li, X., Laplante, P. A., 2017. Be more familiar with our enemies and pave the way forward: A review of the roles bugs played in software failures. **Journal of Systems and Software**, **133**: 68–94.
6. Zheng, W., Shen, T., Chen, X., Deng, P., 2022. Interpretability application of the just-in-time software defect prediction model. **Journal of Systems and Software**, **188:** 111245.
7. Khleel, N. A. A., Nehéz, K., 2022. A new approach to software defect prediction based on convolutional neural network and bidirectional long short-term memory. **Production Systems and Information Engineering**, **10**(3): 1–18.
8. LaToza, T. D., Venolia, G., DeLine, R., 2006. Maintaining mental models: A study of developer work habits. **Proceedings of the 28th International Conference on Software Engineering (ICSE '06)**, 492–501.
9. Gezici Geçer, B., Kolukısa Tarhan, A., 2025. Explainable AI framework for software defect prediction. **Journal of Software: Evolution and Process, 37**(4).

10. Giray, G., Bennin, K. E., Köksal, Ö., Babur, Ö., Tekinerdogan, B., 2023. On the use of deep learning in software defect prediction. **Journal of Systems and Software**, **195:** 111537.
11. He, R., Li, Y., Sun, C., 2025. Understanding software defect prediction through explainable neural additive models. **IEEE Access**, **13**: 82860–82873.
12. Gezici, B., Tarhan, A. K., 2024. Advancing Software Defect Prediction Through Ensemble XAI Methods: Insights and Performance Evaluation. Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi, Ankara, 195 s.
13. Aydoğan, B., 2024. Comparative Analysis of Explanatory Methods in Explainable Artificial Intelligence: An Assessment of Robustness, Error, and Efficiency. Bahçeşehir Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul, 67 s.
14. Mustaqeem, M., Mustajab, S., Alam, M., Jeribi, F., Alam, S., Shuaib, M., 2024. A trustworthy hybrid model for transparent software defect prediction: SPAM-XAI. **PLOS ONE, 19**(7), e0307112.
15. Grattan, N., da Costa, D. A., Stanger, N., 2024. The need for more informative defect prediction: A systematic literature review. **Information and Software Technology**, **171**: 107456.
16. Ribeiro, M. T., Singh, S. Guestrin, C., 1135–1144. *"*Why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2016, San Francisco – USA, ACM - Association for Computing Machinery, 10 s.
17. Batool, I., Khan, T. A., 2022. Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review. **Computers and Electrical Engineering**, **100**(02): 107886.
18. Asal, B., Özdeş Demir, M., 2024. Enhancing software defect prediction through explainable AI: Integrating SHAP and LIME in a voting classifier framework. **2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)**, 1-7.
19. Pressman, R. S., Maxim, B. R., 2014. Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education, New York, 930 s.

20. Bacchelli, A., Bird, C., Expectations, outcomes, and challenges of modern code review, 712–721. *2013 35th International Conference on Software Engineering (ICSE),*2013, San Francisco, CA, USA, IEEE, 10 s.
21. Lundberg, S. M., Lee, S.-I., 2017. A unified approach to interpreting model predictions. **Advances in Neural Information Processing Systems**, **30:** 4765–4774.
22. Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., Ranjan, R., 2023. Explainable AI (XAI): Core ideas, techniques, and solutions. **ACM Computing Surveys**, **55**(9): 1–33
23. Shepperd, M., Song, Q., Sun, Z., Mair, C., 2013. Data quality: Some comments on the NASA software defect datasets. **IEEE Transactions on Software Engineering**, **39**(9): 1208–1215.
24. Jureczko, M., Madeyski, L., 2010. Towards identifying software project clusters with regard to defect prediction. **PROMISE'10 Proceedings of the 6th International Conference on Predictive Models in Software Engineering**, 1–10.
25. Xu, G., Zhu, Z., Guo, X., Wang, W., 2025. A joint learning framework for bridging defect prediction and interpretation. **arXiv preprint arXiv:2502.16429**, arXiv.
26. Wahono, R. S., 2015. A systematic literature review of software defect prediction: Research trends, datasets, methods and frameworks. **Journal of Software Engineering**, **1**(1): 1–16.
27. Gezici, B., Tarhan, A. K., 2022. Explainable AI for software defect prediction with gradient boosting classifier, 1-6. *7th International Conference on Computer Science and Engineering (UBMK),* Diyarbakir, Turkey, IEEE, 6 s.
28. Jiarpakdee, J., Tantithamthavorn, C. K., Grundy, J., 2021. Practitioners' perceptions of the goals and visual explanations of defect prediction models, 432–443. *IEEE/ACM International Conference on Mining Software Repositories (MSR)*, Madrid, Spain, IEEE, 12 s.
29. LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. **Nature**, **521**(7553): pp. 436–444.
30. Al Jumaili, S. H. S., 2025. Akıllı Ulaşım Sistemlerinde Trafik Akışı Tahmini İçin LSTM ve Gradyan Yükseltme ile Hibrit Bir Topluluk Yaklaşımı / A Hybrid Ensemble Approach With LSTM and Gradient Boosting for Traffic Flow

Prediction in Intelligent Transportation Systems. Çankaya Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Ankara, 107s.

31. Şen Kaya, Ö., 2023. Makine Öğrenmesi Algoritmaları ile Yazılım Hata Kestirimi. Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Eskişehir, 83s.
32. Guven Aydın, Z. B., Samli, R., 2020. Performance evaluation of some machine learning algorithms in NASA defect prediction data sets, 1-3. *5th International Conference on Computer Science and Engineering (UBMK)*, September 2020, Diyarbakır, Turkey, IEEE, 3 s.
33. Alsaeedi, A., Khan, M. Z., 2019. Software defect prediction using supervised machine learning and ensemble techniques: A comparative study. **Journal of Software Engineering and Applications**, **12**(3): 85–100.
34. Yang, X., Lo, D., Xia, X., Zhang, Y., Sun, J., 2015. Deep learning for just-in-time defect prediction, 17-26. *IEEE International Conference on Software Quality, Reliability and Security (QRS),* August 2015, Vancouver, BC, Canada, IEEE, 10 s.
35. Hoang, T., Dam, H. K., Kamei, Y., Lo, D., Ubayashi, N., 2019. DeepJIT: An end-to-end deep learning framework for just-in-time defect prediction. *IEEE/ACM International Conference on Mining Software Repositories (MSR),* 2019, Montreal, QC, Canada, IEEE, 12 s.
36. Qiao, L., Li, X., Umer, Q., Guo, P., 2020. Deep learning based software defect prediction. **Neurocomputing**, **385**: 100–110.
37. Li, J., He, P., Zhu, J., Lyu, M. R., 2017. Software defect prediction via convolutional neural network. **IEEE International Conference on Software Quality, Reliability and Security (QRS)**, 318–328.
38. Rai, A., 2020. Explainable AI: From black box to glass box. **Journal of the Academy of Marketing Science**, **48**(1): 137–141.
39. Chung, J., Gulcehre, Ç., Cho, K., Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv Ön Baskı,** arXiv:1412.3555: 1-9.
40. Thant, M. W., Aung, N. T. T., 2019. Software defect prediction using hybrid approach, 262–267. *2019 International Conference on Advanced Information Technologies (ICAIT),* November 2019, Yangon, Myanmar, IEEE, 16 s.

41. Arora, L., Girija, S. S., Kapoor, S., Raj, A., Pradhan, D., Shetgaonkar, A., 2025. Explainable Artificial Intelligence techniques for software development lifecycle: A phase specific survey, 2281-2288. *2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC), 8-11 July* 2025, *Toronto, ON, Canada*, 8 s.

42. Du, M., Liu, N., Hu, X., 2018. Techniques for interpretable machine learning. **arXiv preprint arXiv:1808.00033**, arXiv.

43. Charlotte, L. J., 2025. Explainable ai in software defect prediction: balancing accuracy and interpretability. (Web sayfası: **https://www.researchgate.net/publication/392708858** (Erişim tarihi: 1 Şubat 2026).

44. Lee, G., Ju, H., Lee, S. U.-J., 2025. Can we trust the actionable guidance from explainable AI techniques in defect prediction?, 476–487. *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER),* 4-7 March 2025*,* Montreal, QC, Canada, IEEE , 12 s.

45. Roy, S., Laberge, G., Roy, B., Khomh, F., Nikanjam, A., Mondal, S., 2022. Why don't XAI techniques agree? Characterizing the disagreements between post-hoc explanations of defect predictions, 444–448. *IEEE International Conference on Software Maintenance and Evolution (ICSME), 3-7 October 2022,* Limassol, Cyprus, IEEE, 5 s.

46. Krishna, S., Han, T., Gu, A., Wu, S., Jabbari, S., Lakkaraju, H., 2023. The disagreement problem in explainable machine learning: A practitioner's perspective. **Transactions on Machine Learning Research**, 1–15.

47. Jiarpakdee, J., Tantithamthavorn, C., Hassan, A. E., 2018. The impact of correlated metrics on defect models, 320-331. **IEEE Transactions on Software Engineering, 47**(2), 12 s.

48. Yu, J., Fu, M., Ignatiev, A., Tantithamthavorn, C., Stuckey, P. (2024). A formal explainer for just-in-time defect predictions. **ACM Transactions on Software Engineering and Methodology, *33***(7), 1–31.

49. Yang, F., Zeng, G., Zhong, F., Xiao, P., Zheng, W., Qiu, F., 2024. CfExplainer: Explainable just-in-time defect prediction based on counterfactuals. **Journal of Systems and Software**, **218:** 112182

50. Haldar, S., Capretz, L. F., 2023. Explainable software defect prediction from cross-company project metrics using machine learning, 150-157. *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS),* 17-19 May 2023, Madurai, India, IEEE, 8 s.

51. Zhao, Y., Damevski, K., Chen, H., 2022. A systematic survey of just-in-time software defect prediction. **ACM Computing Surveys**, **55**(10): 1–25.

52. Shin, J., Aleithan, R., Nam, J., Wang, J., Shiri Harzevili, N., Wang, S., 2023. An empirical study on the stability of explainable software defect prediction, 150-157. **30th Asia-Pacific Software Engineering Conference (APSEC)** (December 2023) Bildiriler, SSRN Electronic Journal, 8 s.

53. Begum, M., Shuvo, M. H., Ashraf, I., Mamun, A. A., Uddin, J., & Samad, M. A., 2023. Software defects identification: Results using machine learning and explainable artificial intelligence techniques. **IEEE Access**, **11:** 132750–132765.

54. Schulte, L., Ledel, B., Herbold, S., 2024. Studying the explanations for the automated prediction of bug and non-bug issues using LIME and SHAP, **Empir Software Eng, 29,** 93

55. Al-Smadi, Y., Eshtay, M., Al-Qerem, A., Nashwan, S., Ouda, O., Abd El-Aziz, A. A., 2023. Reliable prediction of software defects using Shapley interpretable machine learning models. **Egyptian Informatics Journal**, **24**(3): 100386.

56. Lee, G., Lee, S. U.-J., 2023. An empirical comparison of model-agnostic techniques for defect prediction models, 179–189. *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER),* 21-23 March 2023, Taipa, Macao, IEEE, 11 s.

57. Tantithamthavorn, C., Hassan, A. E., Matsumoto, K., 2018. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. **IEEE Transactions on Software Engineering**, **46**(11): 1200–1219.

58. Shin, J., Aleithan, R., Nam, J., Others, 2021. Explainable software defect prediction: Are we there yet? **arXiv preprint arXiv:2111.10901**, arXiv.

59. Gottumukkala, D. P., Reddy, P. V. G. D., Rao, S. K., 2025. Topic modeling-based prediction of software defects and root cause using BERTopic and multioutput classifier. **Scientific Reports**, **15:** 11458.

60. Pornprasit, C., Tantithamthavorn, C., Jiarpakdee, J., Fu, M., Thongtanunam, P., 2021. PyExplainer: Explaining the predictions of just-in-time defect models, 407-418.

2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), 15-19 November 2021, Melbourne, Australia, 12 s.

61. Öztürk, M. M. 2016. Tekrar Eden Veri Analizini Kullanarak Yazılım Geliştirme İçin İyileştirilmiş Hata Tahmini. Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar ve Bilişim Mühendisliği Anabilim Dalı, Doktora Tezi, Sakarya, 151 s.
62. Çetiner, M. 2020. Makine Öğrenmesi Yöntemleri ile Yazılım Hata Tahmini. İstanbul Kültür Üniversitesi, Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi, İstanbul, 63 s.
63. Bahaa, A., Fathy, E. M., Eldin, A. S., Abd-Elmegid, L. A., 2021. A systematic literature review of software defect prediction using deep learning. **Journal of Computer Science,** 17(5), 490–510.
64. Breiman, L., 2001. Random forests. **Machine Learning**, **45** (1): 5–32.
65. Friedman, J. H., 2001. Greedy function approximation: A gradient boosting machine. **Annals of Statistics**, **29** (5): 1189–1232.
66. Abdou, A. S., Darwish, N. R., 2018. Early prediction of software defect using ensemble learning: A comparative study. **International Journal of Computer Applications, 179**(27): 29–40.
67. Wen, M., 2020. How well do change sequences predict defects? Sequence learning from software changes. **IEEE Transactions on Software Engineering**, **46** (11): 1155–1175.
68. Abdu, A., Zhai, Z., Abdo, H. A., Algabri R., Al-masni M. A., Muhammad M. S, Gu Y. H., 2024. Semantic and traditional feature fusion for software defect prediction using hybrid deep learning model. **Scientific Reports, 14,** 14771.
69. 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.
70. Heaton J., 2016. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. **Genetic Programming and Evolvable Machines, 19,** 305-307.
71. Kingma, D. P., Ba, J., 2015. Adam: a method for stochastic optimization, pp. 1-9. *3rd International Conference on Learning Representations (ICLR 2015) (May 7-9, 2015, San Diego, CA, USA)* Bildiriler, 9 pp.

72. Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S., 2012. A systematic literature review on fault prediction performance in software engineering.**IEEE Transactions on Software Engineering, 38**(6), 1276–1304.

73. Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. **Proceedings of the 14th International Joint Conference on Artificial Intelligence, 2,** 1137–1143.

74. Molnar, C., 2022. Interpretable Machine Learning: A Guide For Making Black Box Models Explainable. Erişim adresi: **https://christophm.github.io/interpretable-ml-book** (Erişim tarihi: 21.11.2025).

75. Tutorials for eXplainable Artificial Intelligence (XAI) methods, 2023. (Web sayfası: **https://xai-tutorials.readthedocs.io/en/latest**), (Erişim tarihi: Mart 2026).

76. Nguyen, H. T. T., Cao, H. Q., Nguyen, K. V. T., Nguyen, K. V. T., Pham, N. D. K., 2021. Evaluation of explainable artificial intelligence: SHAP, LIME, and CAM. May 2021, *ResearchGate preprint*, 6 s.

77. Duell, J., Fan, X., Burnett, B., Aarts, G., & Zhou, S.-M., 2021. A comparison of explanations given by explainable artificial intelligence methods on analysing electronic health records, 1-4. *Proceedings of the 2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, 27-30 July 2021, Athens, Greece, IEEE, 4 s.

78. Shafiabady, N., Hadjinicolaou, N., Hettikankanamage, N., MohammadiSavadkoohi, E., Wu, R. M. X., & Vakilian, J., 2024. Explainable artificial intelligence (XAI) for improving organisational agility. **PLOS ONE, 19**(4)**:** e0301429.

79. Saranya, A., Subhashini, R., 2023. A systematic review of explainable artificial intelligence models and applications: Recent developments and future trends. **Data Analytics and Artificial Intelligence Journal, 7:** 100230.

80. Roshinta, T. A., Gábor, S., 2024. A comparative study of LIME and SHAP for enhancing trustworthiness and efficiency in explainable AI systems, 134-139. *Proceedings of the 2024 IEEE International Conference on Computing (ICOCO)*, 24 March 2024, Kuala Lumpur, Malaysia, IEEE, 6 s.

81. Stow, M., Stewart, A. A. (2025). Interpreting machine learning predictions with SHAP and LIME for transparent decision making. **International Journal of Computer Science and Mathematical Theory, 11**(8)**:** 22–49.

82. Givisis, I., Kalatzis, D., Christakis, C., & Kiouvrekis, Y., 2025. Comparing explainable AI models: SHAP, LIME, and their role in electric field strength prediction over urban areas. **Electronics, 14**(23): 4766.

83. Encyclopaedia Britannica, 2024. Pearson's correlation coefficient. (Web sayfası: **https://britannica.com/topic/Pearsons-correlation-coefficient),** (Erişim tarihi: Ekim 2025).

84. Radjenović, D., 2013. Software fault prediction metrics: A systematic literature review. **Information and Software Technology**, **55** (8): 1397–1418.

85. Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A. Ubayashi, N., 2013. A large-scale empirical study of just-in-time quality assurance. **IEEE Transactions on Software Engineering**, **39** (6): 757-773.

86. Santos, G., Figueiredo, E., Veloso, A., Viggiato, M. Ziviani, N., 2021. Predicting software defects with explainable machine learning. **SBQS'20: Proceedings of the XIX Brazilian Symposium on Software Quality, 18:** 1–10.

87. Tantithamthavorn, K. Hassan, A. E., 2018. An experience report on defect modelling in practice: Pitfalls and challenges, pp. 286-295. *2018 ACM/IEEE 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP 2018)*,30 May – 1 June 2018, Gothenburg, Sweden, IEEE, 10 s.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı: Halime BAYRAKTAR
Uyruğu: Türkiye (TC)
Doğum Tarihi ve Yeri:
Medeni Durumu:
Tel:
E-posta:
Yazışma Adresi:

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Lisans	Erciyes Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü	2017
Lise	Baki Ayşe Simitçioğlu Anadolu Lisesi, Kayseri	2011

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2021- 2026	Alanya Alaaddin Keykubat Üniversitesi	Mühendis
2026- Halen	Niğde Belediyesi	Mühendis

YABANCI DİL

İngilizce

YAYINLAR