T.C.
ONDOKUZ MAYIS ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
HESAPLAMALI BİLİMLER ANA BİLİM DALI

ANDROID CİHAZLARDA MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANILARAK KÖTÜCÜL YAZILIM TESPİTİ

Doktora Tezi

Durmuş Özkan ŞAHİN

Danışman

Prof. Dr. Erdal KILIÇ

SAMSUN
2022

TEZ KABUL VE ONAYI

Durmuş Özkan ŞAHİN tarafından, **Prof. Dr. Erdal KILIÇ** danışmanlığında hazırlanan **"ANDROID CİHAZLARDA MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANILARAK KÖTÜCÜL YAZILIM TESPİTİ"** başlıklı bu çalışma, jürimiz tarafından 22.6.2022 tarihinde yapılan sınav sonucunda oy birliği ile başarılı bulunarak Doktora Tezi olarak kabul edilmiştir.

	Unvanı Adı Soyadı **Üniversitesi** **Ana Bilim/Ana Sanat Dalı**	**İmza**	**Sonuç**
Başkan	Doç. Dr. Halil Murat Ünver Kırıkkale Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
Üye	Prof. Dr. Erdal KILIÇ Ondokuz Mayıs Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
Üye	Doç. Dr. Sedat AKLEYLEK Ondokuz Mayıs Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
Üye	Dr. Öğr. Üyesi İlyas EMİNOĞLU Ondokuz Mayıs Üniversitesi Elektrik Elektronik Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
Üye	Dr. Öğr. Üyesi Recep Sinan ARSLAN Kayseri Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret

Bu tez, Enstitü Yönetim Kurulunca belirlenen ve yukarıda adları yazılı jüri üyeleri tarafından uygun görülmüştür.

ONAY
… / … / …
Prof. Dr. Ali BOLAT
Enstitü Müdürü

BİLİMSEL ETİĞE UYGUNLUK BEYANI

Hazırladığım Doktora tezinin bütün aşamalarında bilimsel etiğe ve akademik kurallara riayet ettiğimi, çalışmada doğrudan veya dolaylı olarak kullandığım her alıntıya kaynak gösterdiğimi ve yararlandığım eserlerin Kaynaklar'da gösterilenlerden oluştuğunu, her unsurun enstitü yazım kılavuzuna uygun yazıldığını ve TÜBİTAK Araştırma ve Yayın Etiği Kurulu Yönetmeliği'nin 3. bölüm 9. maddesinde belirtilen durumlara aykırı davranılmadığını taahhüt ve beyan ederim.

Etik Kurul Gerekli mi?

Evet ☐ (Gerekli ise ekler kısmına ekleyiniz)

Hayır ☒

İmza
24/05/2022
Durmuş Özkan ŞAHİN

TEZ ÇALIŞMASI ÖZGÜNLÜK RAPORU BEYANI

Tez Başlığı: ANDROID CİHAZLARDA MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANILARAK KÖTÜCÜL YAZILIM TESPİTİ

Yukarıda başlığı belirtilen tez çalışması için şahsım tarafından 24/05/2022 tarihinde intihal tespit programından alınmış olan özgünlük raporu sonucunda;

Benzerlik oranı : % 9

Tek kaynak oranı : % 1 çıkmıştır.

İmza
24/05/2022
Prof. Dr. Erdal KILIÇ

ÖZET

ANDROID CİHAZLARDA MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANILARAK KÖTÜCÜL YAZILIM TESPİTİ

Durmuş Özkan ŞAHİN
Ondokuz Mayıs Üniversitesi
Lisansüstü Eğitim Enstitüsü
Hesaplamalı Bilimler Ana Bilim Dalı
Doktora, Haziran/2022
Danışman: Prof. Dr. Erdal KILIÇ

Android işletim sistemi, günümüzde en çok tercih edilen mobil işletim sistemidir. Android, mobil cihazlar başta olmak üzere günümüzde otomobiller, beyaz eşyalar, fotoğraf makineleri, akıllı saatler ve giyilebilir cihazlar gibi çok sayıda ürünün içerisinde bulunmaktadır. Bu nedenle, Android işletim sistemine sahip cihazları kötücül yazılımlardan ve saldırılardan korumak oldukça önemlidir. Bu tez çalışmasında, Android kötücül yazılımların tespitini daha verimli yapabilmenin yolları araştırılmaktadır. Bu bağlamda; Android işletim sisteminin güvenliğinde önemli yeri olan izinler öznitelik olarak değerlendirilmektedir. Ardından bu izinler makine öğrenmesi yaklaşımları ile birlikte kullanılarak kötücül yazılımlar ile iyicil yazılımların birbirlerinden ayrıştırılması sağlanmaktadır. Tez kapsamında ilk olarak, filtreleme tabanlı öznitelik seçme yöntemleri ile en ayırt edici özniteliklerin seçilmesi hedeflenmektedir. Veri kümelerinden elde edilen tüm izinleri kullanmak yerine etkili öznitelik seçme yöntemleri ile izinlerin %80'den fazlası elenerek iyi sınıflandırma sonuçları elde edilmektedir. Tez kapsamında ikinci olarak, doğrusal regresyona dayalı öznitelik seçme yöntemi ile geniş kapsamlı deneyler yapılarak çeşitli izin grupları ortaya çıkartılmaktadır. Aynı zamanda bu izin grupları bu alanda çalışacak olan araştırmacıların doğrudan kullanımına sunulmaktadır. Önerilen Android kötücül yazılım tespit sistemi, 27 uygulama iznini öznitelik olarak kullanarak MLP algoritması ile %96'dan fazla başarım oranına ulaşmaktadır. Genel olarak filtreleme tabanlı öznitelik seçme yöntemlerinin ve doğrusal regresyon tabanlı öznitelik seçme yönteminin izin tabanlı Android kötücül yazılım tespit sistemleri için oldukça faydalı olduğu gözlemlenmektedir. Öznitelik seçiminin dışında tez kapsamında bazı araştırmalar daha yapılmaktadır. Bunlardan birincisi, KNN algoritmasında yer alan parametrelerin izin tabanlı Android kötücül yazılım tespit sistemlerinde başarımı nasıl etkilediğinin incelenmesidir. İkincisi, doğrusal regresyona dayalı bir sınıflandırma algoritmasını kullanan Android kötücül yazılım tespit sisteminin nasıl sonuç vereceğinin araştırılmasıdır. Son olarak, bazı derin öğrenme tekniklerinin Android kötücül yazılım tespitindeki performanslarının karşılaştırılmasıdır.

Anahtar Sözcükler: Android güvenliği, Statik analiz, Makine öğrenimi, Android kötücül yazılım tespiti, Öznitelik seçimi.

ABSTRACT

MALWARE DETECTION USING MACHINE LEARNING ALGORITHMS ON ANDROID DEVICES

Durmuş Özkan ŞAHİN
Ondokuz Mayıs University
Institute of Graduate Studies
Department of Computational Sciences
Ph.D., June/2022
Supervisor: Prof. Dr. Erdal KILIÇ

The Android operating system is the most preferred mobile operating system nowadays. Android is present in many automobiles, white goods, cameras, smartwatches, and wearable devices, especially mobile devices. Therefore, it is essential to protect Android devices from malware and attacks. In this thesis, ways to detect Android malware more efficiently are investigated. In this context; permissions, which have an important place in the security of the Android operating system, are considered as attributes. Then, these permissions are used together with machine learning approaches to distinguish between malicious applications and benign applications. Within the scope of the thesis, firstly, it is aimed to select the most distinctive features with filter-based feature selection methods. Instead of using all the permissions obtained from the datasets, more than 80% of the permissions are eliminated with effective feature selection methods, and good classification results are obtained. Secondly, within the scope of the thesis, various permission groups are revealed by conducting extensive experiments with the feature selection method based on linear regression. At the same time, these permission groups are offered to the direct use of researchers who will work in this field. The proposed Android malware detection system achieves more than a 96% success rate with the MLP algorithm, using 27 application permissions as attributes. In general, it is observed that filter-based feature selection methods and linear regression-based feature selection methods are quite useful for permission-based Android malware detection systems. Apart from feature selection, some more researches are carried out within the scope of the thesis. The first of these is to examine how the parameters in the KNN algorithm affect the performance of permission-based Android malware detection systems. The second is to explore how the Android malware detection system, which uses a linear regression-based classification algorithm, will yield results. Finally, it is a comparison of the performance of some deep learning techniques in Android malware detection.

Keywords: Android security, Static analysis, Machine learning, Android malware detection, Feature selection.

ÖNSÖZ VE TEŞEKKÜR

Akademik eğitim sürecimde hep yanımda olan, çalışmalarım boyunca yardım ve desteğini benden esirgemeyen doktora tez danışmanım değerli hocam Sayın Prof. Dr. Erdal KILIÇ'a, yönlendirme ve yardımları için değerli hocam Doç. Dr. Sedat AKLEYLEK ve Öğretim Görevlisi Oğuz Emre KURAL'a teşekkürlerimi sunarım. Ayrıca tez izleme komitesinde yer alan, bilgi ve tecrübeleriyle tezimin ilerlemesinde katkıda bulunan Dr. Öğr. Üyesi İlyas EMİNOĞLU hocama teşekkürlerimi sunarım.

Hayatımın her anında olduğu gibi tez yazım süresince desteğini esirgemeyen, tez çalışmalarıma ağırlık verip kendisi ile ilgilenmem gereken zamanları azaltmak zorunda kaldığım kıymetli eşim Arş. Gör. Meryem SOYSALDI ŞAHİN'e çok teşekkür ederim.

Durmuş Özkan ŞAHİN

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

AAPT2	**:** Android Asset Packaging Tool
API	**:** Application Programming Interface (Uygulama Programlama Arayüzü)
APK	**:** Android Package Kit veya Android Application Package (Android Uygulama Paketi)
AMD	**:** Android Malware Dataset (Android Kötücül Yazılım Veri Kümesi)
CHI	**:** Chi-squared Test (Ki-kare Testi)
CNN	**:** Convolutional Neural Networks (Evrişimsel Sinir Ağları)
DBN	**:** Deep Belief Network (Derin İnanç Ağları)
DEX	**:** Dalvik Executable (Dalvik Yürütülebilir Dosya)
DF	**:** Document Frequency Thresholding (Doküman Frekansı Eşikleme)
DNN	**:** Deep Neural Networks (Derin Sinir Ağları)
DT	**:** Decision Trees (Karar Ağaçları)
DVM	**:** Dalvik Virtual Machine (Dalvik Sanal Makinesi)
GUI	**:** Graphical User Interface (Grafiksel Kullanıcı Arayüzü)
IDF	**:** Inverse Document Frequency (Ters Doküman Frekansı)
IG	**:** Information Gain (Bilgi Kazancı)
JAR	**:** Java Archive (Java Arşiv)
KNN	**:** K-Nearest Neighbors (K-En Yakın Komşu)
LDA	**:** Linear Discriminant Analysis (Doğrusal Ayırma Analizi)
LR	**:** Logistic Regression (Lojistik Regresyon)
LSTM	**:** Long Short Term Memory (Uzun Kısa Süreli Bellek)
MLP	**:** Multi-Layer Perceptron (Çok Katmanlı Algılayıcı)
NB	**:** Naive Bayes
OR	**:** Odds Ratio (Göreceli Olasılıklar Oranı)
PCA	**:** Principal Component Analysis (Temel Bileşenler Analizi)
PSO	**:** Particle Swarm Optimization (Parçacık Sürü Optimizasyonu)
RBF	**:** Radial Basis Function (Radyal Temelli Fonksiyon)
RF	**:** Random Forest (Rastgele Orman)
RFFS	**:** Relevance Frequency Feature Selection (İlişki Sıklığı Öznitelik Seçimi)
RNN	**:** Recurrent Neural Network (Yinelemeli Sinir Ağı)

SMO : Sequential Minimal Optimization (Sıralı Minimum Optimizasyonu)

SRBM : Subspace based Restricted Boltzmann Machines (Alt Uzay Tabanlı Kısıtlı Boltzmann Makinesi)

SSE : Sum of Squared Error (Kare Hata Toplamı)

SSL : Secure Sockets Layer (Güvenli Soket Katmanı)

SVM : Support Vector Machines (Destek Vektör Makinesi)

TF : Term Frequency (Terim Frekansı)

t-SNE : t-Distributed Stochastic Neighbor Embedding (t-Dağıtılmış Stokastik Komşu Gömme)

XML : Extensible Markup Language (Genişletilebilir İşaretleme Dili)

1D-CNN : 1-Boyutlu Evrişimsel Sinir Ağları

2D-CNN : 2-Boyutlu Evrişimsel Sinir Ağları

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1. GİRİŞ

Günümüzde, kişisel bilgisayar üzerinde gerçekleştirilen birçok işlem artık akıllı mobil cihazlar üzerinde de gerçekleştirilebilmektedir. Bu nedenle akıllı mobil cihaz kullanımında son yıllarda dikkate değer bir artış gözlenmektedir. 2020 yılı itibariyle 3.5 milyar akıllı mobil cihaz kullanıcısı olduğu raporlanırken, bu rakamın 2021 yılı itibariyle 3.8 milyara çıkacağı öngörülmektedir (Statista, 2021b). Bu akıllı cihazların çoğunluğunda işletim sistemi olarak Android kullanılmaktadır. (Statista, 2021a)'da verilen bir incelemeye göre, 2009 ile 2018 yılları arasında dünya çapında satılan akıllı telefonların %88'inde Android işletim sisteminin kullanıldığı tespit edilmektedir.

Android, Google tarafından geliştirilen açık kaynaklı ve Linux tabanlı bir mobil işletim sistemidir. Açık kaynaklı bir işletim sistemi olmasının getirdiği birçok avantaj olmasına karşın temel olarak izin etiketlemesine dayalı bir güvenlik mekanizmasına sahip olduğundan bazı güvenlik problemleri mevcuttur (W. Enck vd, 2009). Android için resmi ve üçüncü parti uygulamaları geliştiren çok sayıda geliştirici vardır. Geliştirilen uygulamaların detaylı bir şekilde incelenmeden resmi veya diğer uygulama depolarına yüklendiği göz önünde bulundurulursa, bir başka ciddi güvenlik probleminin daha olduğu göze çarpmaktadır. Hem geniş uygulama geliştirici kitlesi hem de işletim sisteminden kaynaklanan güvenlik problemleri nedeniyle Android, kötücül yazılım geliştiricilerinin temel hedef noktası haline gelmiş bulunmaktadır. Bilgisayar ve ağ güvenliği şirketi olan Kaspersky'a göre, 2020'nin 2. çeyreğinde 1.245.894 mobil kötü amaçlı yazılım tespit edildiği raporlanmaktadır (Kaspersky, 2021b). Bu sayıda önceki çeyreğe göre 93.232 artış gözlenmiştir. Tespit edilen mobil kötücül yazılımların büyük çoğunluğunun Android cihazları tehdit ettiği vurgulanmaktadır.

Donanım tabanlı, çekirdek tabanlı, donanım soyutlama katmanı tabanlı ve uygulama tabanlı saldırılar olmak üzere Android kötücül yazılımlar genel olarak bu dört grup altında değerlendirilmektedir (Bhat ve Dutta, 2019). Bu kötücül yazılımlar, kullanıcılar adına yetkisiz işlem yapmak, kullanıcıları robot ağ (botnet) saldırılarının bir parçası haline getirmek, kullanıcılardan önemli verileri toplamak, toplanan verilerden maddi kazanç sağlamak, cihazın donanımına zarar vermek gibi birçok maddi ve manevi zararlara sebep olmaktadır. Bu zararların önüne geçmek için hem araştırmacılar hem de güvenlik şirketleri başarım oranları yüksek kötücül yazılım tespit sistemleri tasarlamayı amaçlamaktadırlar.

Mobil kötücül yazılım tespiti statik, dinamik ve hibrit olmak üzere üç farklı şekilde yapılmaktadır (Faruki vd, 2015). Statik analiz, uygulamanın çalıştırılmasına gerek kalmadan uygulama dosyaları üzerinden yapılan test tekniğidir. Dinamik analiz tekniği ise uygulamanın gerçek veya sanal bir cihaz üzerinde çalıştırılmasıyla sergilediği davranışların izlenmesine dayanan analiz yöntemidir. Uygulama davranışları izlendiğinden dinamik analiz davranışsal analiz olarak da adlandırılmaktadır. Statik ve dinamik özelliklerin birlikte kullanılmasıyla yapılan analiz ise hibrit olarak isimlendirilmektedir. Statik ve dinamik analiz yöntemlerinin birbirlerine göre avantajları ve dezavantajları vardır. Bu avantaj ve dezavantajlar göz önünde bulundurularak hibrit analiz yöntemleri geliştirilmektedir. Statik ve dinamik analiz yöntemlerinin birbirlerine göre avantaj ve dezavantajları Tablo 1.1'de karşılaştırmalı olarak verilmektedir.

Tablo 1.1. Statik ve dinamik analizin karşılaştırılması

Statik Analiz	**Dinamik Analiz**
Uygulamalar çalıştırılmadan yapıldığı için daha hızlı gerçekleştirilir.	Uygulamalar gerçek veya sanal bir cihaz üzerinde çalıştırılır bu nedenle gerekli altyapının oluşturulması zahmetlidir.
Uygulamalar çalıştırılmadan gerçekleştirildiğinden kötücül yazılımın cihaza bulaşma durumu söz konusu değildir.	Uygulamalar çalıştırıldığından kötücül yazılım cihazı etkileyebilir. Bu nedenle gerçek cihazdan ziyade sanal bir cihaz tercih edilmektedir.
İlk gün ataklarına karşı savunmasızdırlar.	İlk gün ataklarına karşı daha başarılıdırlar.
Herhangi bir davranış analizi yapılmadığından kötücül yazılım davranışını değiştirebilen uygulama olsa bile statik olarak özellikler elde edilebilecektir.	Yeni nesil bazı kötücül yazılımlar sanal cihaz üzerinde analiz edildiklerini anlayıp davranışlarını değiştirirler. Bu tarz kötücül yazılımlar sanki iyicilmiş gibi savunma mekanizmasını aldatabilirler.

Makine öğrenimi yaklaşımları ile temel olarak veriler üzerinde çeşitli çıkarımlar yapılarak sınıflandırma, kümeleme veya regresyon modelleri oluşturulmaktadır. Bu modellerin oluşturulması adımında matematiksel ve istatistiksel yapılara ihtiyaç duyulmaktadır. Son yıllarda popüler bir çalışma alanı olan makine öğrenimi farklı disiplinlerde birçok alanda kullanılmaktadır. Tıbbı verilerin işlenmesi (Latif vd, 2020), ekonomi verilerinin analizi (X. Wei vd, 2021), yazılım geliştirme süreci (Srinivasan ve Fisher, 1995), sahte haberlerin sınıflandırılması (Hakak vd, 2021), konuşma analizi

(Padmanabhan ve Johnson Premkumar, 2015) ve nesnelerin interneti (L. Xiao vd, 2018) bu alanlara örnek olarak verilebilir. Bu çalışmalara ek olarak Android işletim sisteminin güvenliğinde de araştırmacılar makine öğrenmesi tekniklerinden faydalanmaktadır (K. Liu vd, 2020). Uygulama dosyalarından çıkartılan statik özellikler ile uygulama davranışlarından çıkartılan dinamik özellikler makine öğrenmesi algoritmalarına girdi olarak verilmektedir. Böylece Android kötücül yazılım tespiti veya kötücül yazılımların aile sınıflandırılması makine öğrenmesi yaklaşımları ile yapılarak yüksek başarımlar elde edilmektedir.

1.1. Tezin Amacı ve Katkı

Günümüzde cep telefonları ile kişisel bilgisayarlar üzerinde yapılabilecek çoğu işlemler yapılabilmektedir. İlk cep telefonları düşünüldüğünde, genellikle konuşma veya kısa mesaj işlemleri günlük hayatta cep telefonları ile gerçekleştirilmekteydi. Ancak günümüzde kullanılan cep telefonları ile bankacılık işlemleri, sosyal medya kullanımı, kişisel verilerin depolanması gibi önemli işlemler gerçekleşmektedir. Bu önemli işlemlerden dolayı kötücül yazılım geliştiricilerinin ana hedef noktasında mobil cihazlar yer almaktadır.

Android açık kaynaklı Linux tabanlı mobil bir işletim sistemidir. Açık kaynaklı ve ücretsiz olmasından ötürü birçok telefon veya tablet üretici firma cihazlarında bu işletim sistemini tercih etmektedir. Bu nedenle pazarın büyük çoğunluğu Android cihazlardan oluşmaktadır. Android'in açık kaynaklı bir işletim sistemi olmasının yanı sıra resmi uygulama depoları dışından da cihazlara uygulama sağlanıyor olması kullanıcılar için oldukça esnek bir durumdur. Bu nedenle dünya genelinde pek çok insan tarafından Android sıkça tercih edilmektedir.

Resmi olmayan uygulama depolarından veya üçüncü parti uygulama geliştiricilerinden temin edilen uygulamalar kullanıcılar için oldukça avantajlı olsalar da bu uygulamaların bazılarının kötücül yazılımlar olduğu göz ardı edilmemelidir. Resmi uygulama depolarında yer alan uygulamalar dikkatli bir şekilde analiz edilip uygulama depolarında yayınlanırlar. Buna rağmen resmi uygulama depolarında bile kötücül yazılımlara sıkça rastlanılmaktadır (Kaspersky, 2021c). (H. Wang vd, 2018) tarafından yapılan araştırmada, 17 tane uygulama mağazasından indirilen 6 milyondan fazla uygulama değerlendirilmektedir. Bu mağazalardan 16 tanesi Çin'de yaygın bir şekilde kullanılırken 1 tanesi de Google Play'dir. Genel olarak Google Play'in diğer

uygulama mağazalarına göre daha güvenilir olduğu ortaya çıkartılmaktadır. Ancak kötücül yazılımları hemen hemen bütün mağazalarda görmek mümkündür.

2015 yılının ilk 6 ayında neredeyse 1 milyon yeni kötücül yazılım tespit edilirken, 2019 yılının ilk 6 ayında ise 1.85 milyon yeni kötücül yazılım tespit edilmektedir (Gdatasoftware, 2021). Tüm önlemlere rağmen kötücül yazılım sayısında dikkate değer bir artış görülmektedir. Bu nedenle hem araştırmacılar hem de bilgisayar güvenliği üzerine çalışan şirketler mobil kötücül yazılımların tespiti için yeni yaklaşımlar sunmaktadırlar.

Bir uygulama cihaza yüklendikten sonra birçok izin kullanıcıdan talep edilmektedir. Uygulama arka planda çalışırken kullanıcının verdiği izinler doğrultusunda uygulama kötücül özelliğini gösterebilmektedir. Bu nedenle kullanıcıların talep edilen izinlere dikkat etmesi gerekmektedir. Bu tez çalışmasında genel olarak Android güvenliğinde önemli bir yeri olan uygulama izinleri makine öğrenmesi algoritmaları ile değerlendirilerek uygulamanın kötücül yazılım olup olmadığına karar verilmektedir. Uygulama izinleri öznitelik olarak kullanıldığı için kötücül yazılım tespit sistemi izin tabanlı kötücül yazılım tespit sistemi olarak adlandırılmaktadır. Tez kapsamında aşağıda verilen araştırma sorularına cevap verilerek tezin katkısı şöyle özetlenebilir:

- İzin tabanlı Android kötücül yazılım tespit sisteminde öznitelik seçimi ne derece önemlidir?
- Metin sınıflandırma yaklaşımında çok sayıda öznitelik ortaya çıkmaktadır. Bu nedenle metin sınıflandırmada öznitelik seçimi önemli bir araştırma konusudur ve çok sayıda öznitelik seçme yöntemi araştırmacılar tarafından önerilmektedir. Metin sınıflandırmada kullanılan bazı öznitelik seçme yöntemleri izin tabanlı Android kötücül yazılım tespit sistemlerinde kullanılabilir mi?
- Farklı öznitelik seçme yöntemlerinden elde edilen öznitelik alt kümeleri birleştirildiğinde elde edilen bu birleştirilmiş alt küme sınıflandırmayı nasıl etkiler?
- İzin tabanlı yaklaşımı benimseyerek oluşturulmuş bir yapısal veri kümesine doğrusal regresyon modeli uygulandığında elde edilen regresyon katsayılarına göre öznitelik seçimi yapılabilir mi?

- KNN algoritması kullanılarak gerçekleştirilen izin tabanlı Android kötücül yazılım tespitinde, KNN algoritmasının parametreleri sınıflandırma başarımını nasıl etkiler?
- Doğrusal regresyon modeline dayalı kural tabanlı sınıflandırıcılar ile Android kötücül yazılımların tespiti gerçekleştirilebilir mi?
- DNN, 1D-CNN ve 2D-CNN gibi farklı derin öğrenme tekniklerinin kullanılmasıyla gerçekleştirilen Android kötücül yazılım tespit sisteminin başarımı nasıldır?

Bu belirtilen araştırma sorularına çözüm sunabilmek için hazırlanan tezin ilk katkısı öznitelik seçme yöntemlerinin araştırılması olmuştur. Bu bağlamda, makine öğrenmesi algoritmalarının hem çalışma zamanını hem de verimliliğini arttırmak için 8 farklı öznitelik seçme yöntemi ile öznitelik seçme işlemi gerçekleştirilmiştir (Durmuş Özkan Şahin vd, 2021a). Bu yöntemlerden 4 tanesi Android kötücül yazılım tespit sistemlerinde kullanılırken geri kalan 4 yöntem ise metin sınıflandırma çalışmalarından bu alana uyarlanmıştır. Uyarlanan yöntemler hem çıkartılan öznitelikler bakımından hem de sınıflandırma sonuçları bakımından kıyaslanmıştır. Sonuçlar incelendiğinde, uyarlanan yöntemlerin sınıflandırma algoritmalarının verimliliğini arttırdığı ve Android kötücül yazılım tespiti alanında kullanılabilirliği gösterilmiştir. Ayrıca tüm izinleri kullanmak yerine etkili öznitelik seçme yöntemleri ile izinlerin %80'den fazlası elenerek iyi sınıflandırma sonuçları elde edilmiştir. En iyi öznitelik seçme yöntemleri açgözlü yaklaşım ile birleştirildiğinde verimli yeni öznitelik alt kümelerinin oluşturulması sağlanmıştır.

Tezin ikinci katkısı doğrusal regresyon modeline dayalı öznitelik seçme yönteminin izin tabanlı Android kötücül yazılım sistemlerinde verimli bir şekilde kullanılmış olmasıdır (Durmuş Özkan Şahin vd, 2021b). İzinlerden oluşturulmuş bir veri kümesine doğrusal regresyon modeli uygulayarak her bir iznin katsayısı (regresyon katsayısı) hesaplanmıştır. Daha sonra bu katsayılar değerlendirilerek özniteliklerin seçimi gerçekleştirilmiştir. Öznitelik sayısı ciddi bir oranda azalırken sınıflandırma başarımı neredeyse hiç değişmemiştir. Ayrıca sınıflandırma modelinin eğitim süresi ciddi oranda azalmıştır.

KNN algoritmasında, k değeri ve en yakın örneğin bulunmasında kullanılan uzaklık metriği sınıflandırma başarımını doğrudan etkilemektedir. Ayrıca izin tabanlı çalışmalarda KNN algoritmasının parametrelerini detaylı olarak inceleyen çalışma

sınırlıdır. Bu nedenle farklı veri kümeleri altında 5 farklı k değeri ve 5 farklı uzaklık metriği kullanılarak kötücül yazılım tespit sisteminin başarımları karşılaştırmalı olarak verilmiştir (Sahin vd, 2021). KNN algoritmasına dayalı geniş kapsamlı deneylerin sunulması tezin üçüncü katkısıdır. Sonuçlar incelendiğinde, k değerine 1, 3 gibi değerler verildiğinde ve Chebyshev uzaklık metriğinin yerine Euclidean, Minkowski gibi metriklerin seçilmesiyle daha yüksek sınıflandırma başarımlarının elde edildiği gözlemlenmiştir.

Tezin dördüncü katkısı doğrusal regresyona dayalı kural tabanlı basit sınıflandırma yaklaşımlarıdır (Durmuş Özkan Şahin vd, 2022). Bu yaklaşım farklı veri kümeleri üzerinde SVM, KNN, NB ve DT gibi temel makine öğrenmesi teknikleri ile kıyaslanmıştır. Genel olarak doğrusal regresyona dayalı sınıflandırıcının KNN ve NB'den daha iyi başarımlar verdiği görülmüştür. Ayrıca en başarılı sınıflandırıcılar torbalama tekniğine göre bir arada kullanılarak topluluk öğrenmesine dayalı sınıflandırma yaklaşımları ile sonuçlar elde edilmiştir.

Tezin son katkısı ise derin öğrenme yaklaşımlarının kıyaslanmasıdır. İlk olarak öznitelik vektörleri doğrudan DNN ve 1D-CNN'e verilerek sınıflandırma yapılmıştır. Daha sonra statik özelliklerden oluşan vektörler 15×15 boyutlarında siyah beyaz görüntülere dönüştürülerek 2D-CNN ile de sınıflandırılma gerçekleştirilmiştir. Bu üç farklı model iki farklı veri kümesi üzerinde kıyaslanmıştır. Sonuçlar göz önünde bulundurulduğunda statik özelliklerin birleşiminden elde edilen görüntülerin de diğer temsil etme yöntemleri kadar başarılı olduğu gösterilmiştir. Uygulama dosyalarının görüntülere dönüştürülmesiyle son yıllarda farklı kötücül yazılım tespit yaklaşımları ortaya çıktığından bu çalışmada ile literatüre alternatif bir yöntem sunulmuştur.

1.2. Tezin Organizasyonu

Tezin geri kalan bölümleri şöyle organize edilmektedir: Bölüm 2'de, Android işletim sisteminin mimarisine, statik ve dinamik analiz araçlarına, literatür araştırmasına, sınıflandırma algoritmalarına ve değerlendirme metriklerine değinilecektir. Bölüm 3'te, var olan ve uyarlanan öznitelik seçme yöntemleri detaylandırılıp sonuçları tartışılacaktır. Bölüm 4'te, doğrusal regresyona dayalı öznitelik seçme yöntemi anlatılacaktır. Ayrıca bu yöntem ile elde edilen öznitelik alt kümelerinin farklı sınıflandırma algoritmaları altındaki sonuçları verilecektir. Bölüm 5'te, doğrusal regresyona dayalı kural tabanlı sınıflandırma algoritmalarının

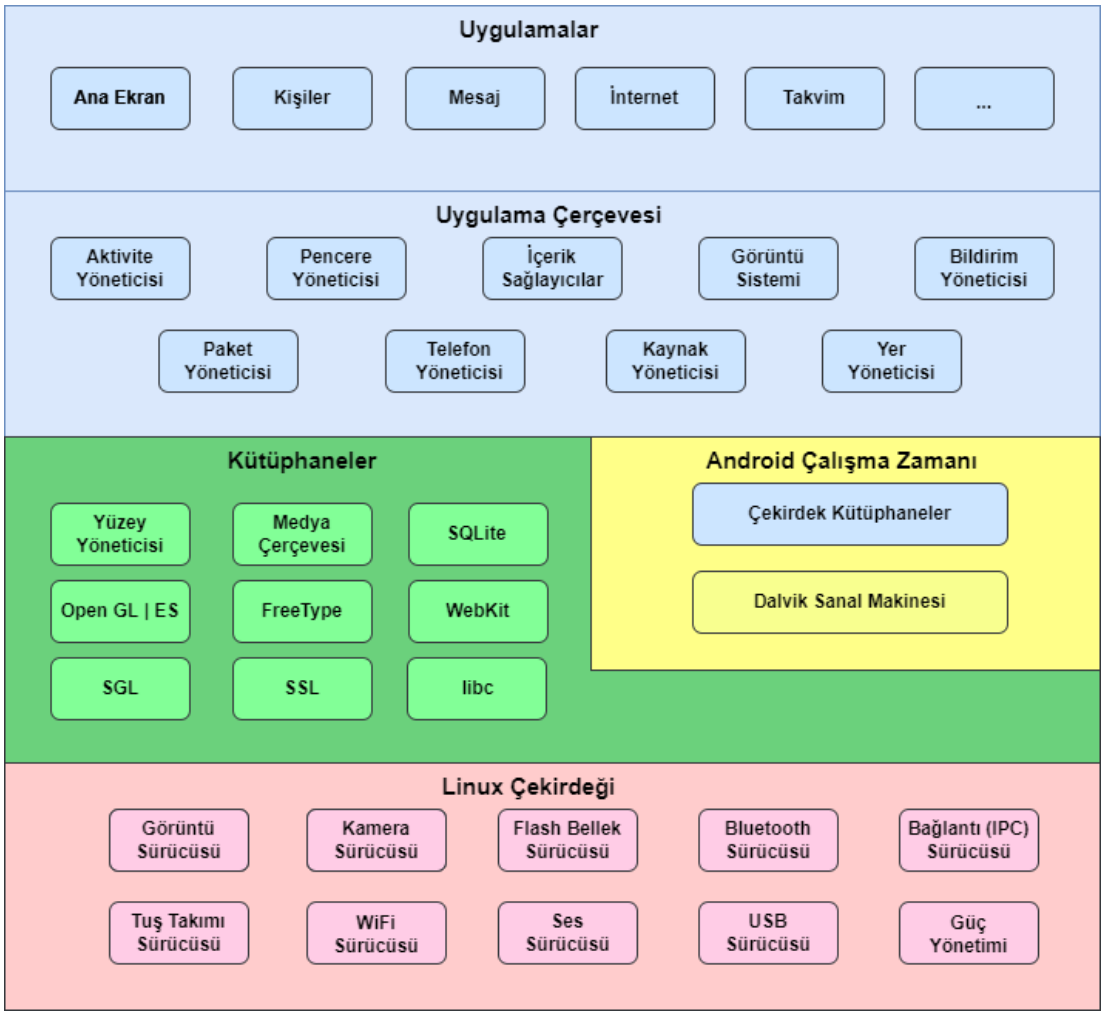
sonuçlarına değinilecektir. Ayrıca en iyi sonucu veren sınıflandırma algoritmaları torbalama tekniklerine göre birlikte kullanılmaktadır. Bölüm 6'da, KNN kullanılarak gerçekleştirilen Android kötücül yazılım tespit sisteminin detaylı sonuçlarına yer verilecektir. Bölüm 7'de, derin öğrenme tekniklerinden elde edilen sonuçlar verilecektir. Son olarak, Bölüm 8'de ise genel değerlendirmeler yapılacaktır. Ayrıca Android kötücül yazılım tespiti alanında çalışmak isteyen araştırmacılara mevcut açık problemlerden bahsedilecektir.

2. TEMEL YAPILAR VE KAYNAK ÖZETLERİ

Bu bölümde ilk olarak Android işletim sisteminin mimarisine, statik ve dinamik analiz araçlarına değinilecektir. Daha sonra sınıflandırma algoritmaları ve değerlendirme metrikleri detaylandırılacaktır. Son olarak, literatür araştırması özetlenecektir.

2.1. Android İşletim Sistemi

Android, başta Google olmak üzere 84 firmanın oluşturduğu uluslararası bir birlik olan Open Handset Alliance tarafından akıllı telefonlar ve tablet bilgisayarlar gibi mobil cihazlar için geliştirilen Linux tabanlı açık kaynaklı bir işletim sistemidir (Android, 2021). Katmanlı bir yapıdan oluşan Android işletim sisteminin mimari yapısı Şekil 2.1'de verilmektedir (Doğru ve Dincer, 2017).



Şekil 2.1. Android işletim sisteminin mimari yapısı

4 katman ve 5 bölümden oluşan Android işletim sisteminin en alt katmanında Linux çekirdeği yer almaktadır. Bu katman aygıt donanımı arasında soyutlama sağlamaktadır. Temel olarak güvenlik, hafıza yönetimi, süreç yönetimi, kamera, tuş takımı, ekran ve benzeri gibi temel donanım sürücülerini içermektedir.

Linux çekirdek katmanının hemen üstünde kütüphaneler ve Android çalışma zamanı (Android runtime) bölümleri bulunmaktadır. Açık kaynaklı Web tarayıcı motoru olan Webkit, iyi bilinen kütüphane olan libc, uygulama verilerinin depolanması ve paylaşılması için SQLite veritabanı, grafik ve arayüz kütüphaneleri ile internet güvenliğinden sorumlu olan SSL kütüphaneleri bu katmanda yer alan kütüphaneler kümesini oluşturmaktadır. Bu katmanda bulunan diğer bir bölüm ise Android çalışma zamanı bölümüdür. Bu bölüm, Android için özel olarak tasarlanmış ve optimize edilmiş Java Sanal Makinesinin bir türü olan Dalvik Sanal Makinesini içermektedir. Dalvik Sanal Makinesi sayesinde her Android uygulama kendi Dalvik Sanal Makinesi üzerinde çalışmaktadır. Bu bölümde Dalvik Sanal Makinesinin yanı sıra Android uygulama geliştiricilerinin standart Java programlama dili ile Android uygulamaları yazabilmeleri için bir çekirdek kütüphane de bulunmaktadır.

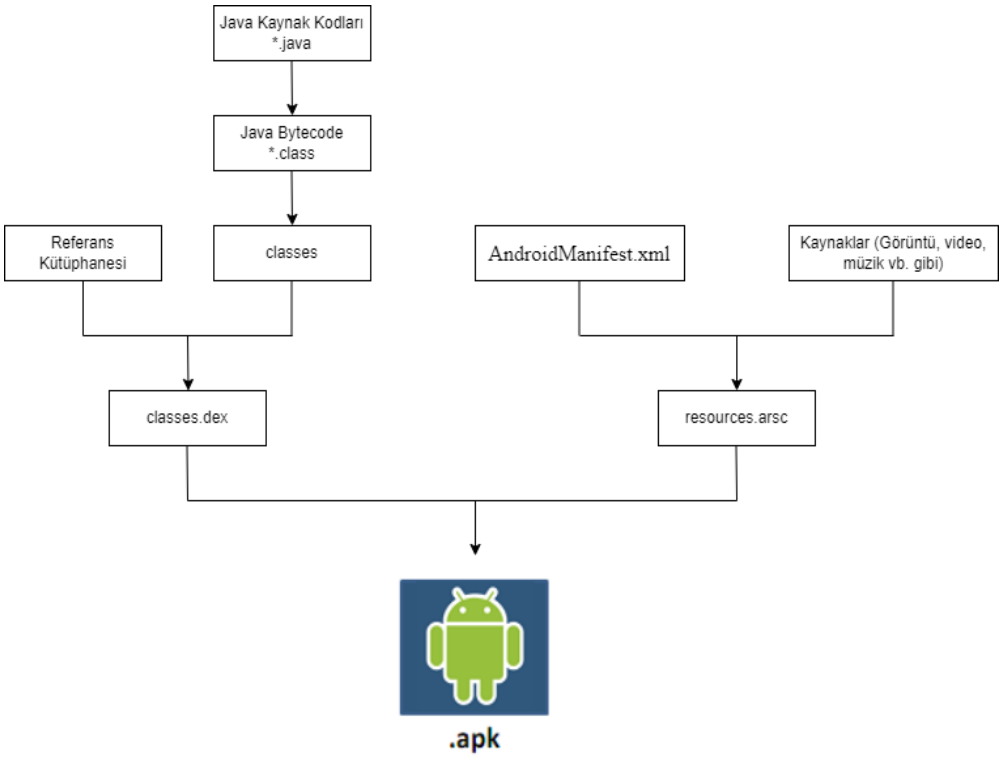
Android işletim sisteminin ikinci katmanı uygulama çerçevesi (application framework) katmanıdır. Uygulama geliştiricilerinin uygulama geliştirirken faydalanabilecekleri Java sınıfları biçiminde pek çok üst düzey servis sağlamaktadır. En üst katman ise uygulama katmanıdır. Yazılan tüm Android uygulamalar bu katmanda bulunmaktadır. Bir Android uygulamanın temel yapı taşları uygulama bileşenleridir. Bu bileşenler, uygulamanın her bileşenini ve birbirlerini nasıl etkileyeceğini tanımlayan AndroidManifest.xml uygulama bildirim dosyası tarafından birleştirilmektedir. Bir Android uygulaması aktiviteler, servisler, yayın alıcıları ve içerik sağlayıcılar olmak üzere dört bileşenden oluşmaktadır.

- Bir aktivite, bir kullanıcı arayüzü ile tek bir ekranı temsil etmektedir. Aktivite ekranda eylemler gerçekleştirmektedir.
- Servis, Android uygulamasında uzun süreli işlemleri gerçekleştirmek için arka planda çalışan bir bileşendir. Örneğin, servis sayesinde kullanıcı farklı bir uygulamadayken arka planda müzik çalabilir veya kullanıcının bir aktivite ile etkileşimini engellemeden ağ üzerinden veri getirebilir.
- Yayın alıcıları, diğer uygulamalardan veya sistemden gelen yayın mesajlarına basit bir şekilde yanıt vermektedir.

- İçerik sağlayıcılar, istek üzerine bir uygulamadan diğerlerine veri sağlamaktadır. Bu tarz istekler, ContentResolver sınıfının metotları tarafından işlenirken veriler dosya sisteminde, veritabanında veya başka bir yerde saklanmaktadır.

APK, Android işletim sistemi tarafından mobil uygulamaların dağıtımı ve kurulumu için kullanılan paket dosyası biçimi olarak bilinmektedir. Windows işletim sistemine sahip cihazlara bir uygulama yüklemek için .exe uzantılı dosyalar kullanılmaktadır. Benzer şekilde, Android işletim sisteminde ise APK dosyalarına ihtiyaç duyulmaktadır. APK dosyaları sıkıştırılmış dosyalar gibi düşünülebilir. Genel olarak bu dosyalar içerisinde uygulama kaynak kodları, uygulama izinleri, uygulamalarda yer alan görüntü ve video dosyaları gibi uygulamaların çalıştırılabilmesinde ihtiyaç duyulan yapılar bulunmaktadır.

Android uygulamalar, çoğunlukla Java programlama dili kullanılarak yazılmaktadır. Geliştiriciler tarafından yazılan Java kaynak kodları derlenerek bytecode'a dönüştürülmektedir. Java sanal makinesinin kurulu olduğu Windows veya Linux tabanlı bir işletim sistemine sahip bilgisayarlar ele alındığında derlenen bu bytecode'lar işlenerek ilgili işletim sistemi üzerinde çalıştırılabilen yapıya dönüştürülmektedirler. Ancak, Android işletim sisteminde bytecode'lar doğrudan çalıştırılamamaktadır. Bu nedenle, bytecode'lar üzerinde bir işlem daha gerçekleştirilerek bytecode'lar çalıştırılabilir Dalvik bytecode'lara dönüştürülmektedir. Böylece Dalvik Sanal Makinesi yardımıyla bu Dalvik bytecode'lar artık çalıştırılabilmektedir. Sonuç olarak yazılan uygulamaların cihaz üzerinde çalıştırılması sağlanmaktadır. Java kaynak kodlarından APK uygulama dosyalarının oluşturulması adım adım Şekil 2.2'de verilmektedir. Şekil 2.2'de derleme işlemi gösterilmektedir. APK dosyalarından bilgi çıkarma işlemi ise derleme işleminin tersidir.



Şekil 2.2. APK dosyalarını oluşturma adımları

APK dosyalarının çalıştırılmadan bilgilerin çıkartılması işlemine statik analiz denilmektedir. Herhangi bir APK dosyası çıkartıldığında Şekil 2.3'te görüldüğü gibi bazı klasör ve dosyalar meydana gelmektedir. Elde edilen bu dosya veya klasörler işlenerek statik özellikler ortaya çıkartılmaktadır. Bu çalışmada, APK dosyalarından çıkartılan AndroidManifest.xml dosyaları değerlendirilerek uygulama izinlerine erişilmektedir. EK1'de bu tez çalışmasında ele alınan bazı izinler ve karşılıkları yer almaktadır. Şekil 2.4'te AndroidManifest.xml dosyasında yer alan bazı izinler gösterilmektedir.

Şekil 2.3'te yer alan AndroidManifest.xml dosyası bir uygulamanın en önemli birleşenidir. Bu dosya içerisinde uygulamanın adı, uygulamada kullanılacak yazılım ve donanım özellikleri, uygulamanın istediği izinler gibi birçok bilgi bulunmaktadır. XML dosyaları derlenerek Resources.arsc dosyası içerisinde bir araya getirilmektedir. Java kaynak kodların derlenmesiyle oluşturulan Dalvik bytecode'lar bir araya getirilerek Classes.dex dosyasının içerisinde yer almaktadırlar. Uygulamanın sertifika ve imzaları META-INF klasöründe bulunmaktadır. Uygulamalarda yer alan video, resim, müzik, ses gibi kaynak dosyalar res klasörü içerisinde tutulmaktadır. Assets klasöründe de res klasöründe olduğu gibi kaynak dosyalar bulunmaktadır. Ancak

Assets ve res klasörleri arasında bir takım farklılıklar vardır. Son olarak, lib klasöründe ise uygulama tarafından ihtiyaç duyulan kütüphaneler yer almaktadır.

Şekil 2.3. APK dosyasının işlenmesi sonucunda elde edilen dosyalar

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="com.android.alarm.permission.SET_ALARM" />
<uses-permission android:name="android.permission.VIBRATE" />
```

Şekil 2.4. AndroidManifest.xml dosyasında yer alan bazı izinler

APK dosyalarını statik olarak işlemek veya uygulamaların davranışlarını incelemek için çok sayıda araç geliştirilmiştir. Bu araçlardan bazıları ve bunların kullanım amaçları şöyle özetlenmektedir:

Androguard: Android dosyalarıyla oynamak için tasarlanan bir Python kütüphanesidir (Androguard, 2021). Sadece Python 3 ile çalışmak üzere tasarlanmıştır. Bu kütüphane ile çok sayıda statik analiz işlemi gerçekleştirilmektedir.

dex2jar: Classes.dex dosyasını JAR dosyasına dönüştürmek için kullanılan bir araçtır (dex2jar, 2021).

JD-GUI: Android uygulamasının kaynak kodlarını görüntülemeye yardımcı olan bir araçtır (JD-GUI, 2021). Classes.dex dosyasının dex2jar yardımıyla JAR dosyasına dönüştürülmesinin ardından JD-GUI aracı vasıtasıyla elde edilen JAR dosyası açılarak uygulamanın kaynak kodlarına erişilebilmektedir.

AAPT2: Uygulamaların kaynak kodlarını derlemek ve paketlemek için Android Studio ve Android Gradle Plugin'in kullandığı bir oluşturma aracıdır (AAPT2, 2021). AAPT2, kaynak kodları Android platformu için optimize edilmiş bir ikili biçime ayrıştırarak indeksler ve derler. Aynı zamanda izinlerin verimli bir şekilde elde edilmesinde de bu araçtan faydalanılabilir.

Apktool: APK dosyalarını çözümleyebilmek için içerisindeki dosyaları okunabilir hale getirmeye yarayan bir araçtır (Apktool, 2021). Kaynak kodların üzerinde bazı değişikliklerin yapılmasına yardımcı olur. Değişiklikler yapıldıktan sonra oluşturulan uygulamanın yeniden kullanılabilmesini sağlamaktadır.

DroidBox: Android uygulamalarını dinamik olarak analiz etmek için geliştirilmiş bir araçtır (DroidBox, 2021). Gelen ve giden ağ verilerinin takibi, dosya okuma yazma takibi, giden SMS ve arama takibinin yanında şifreleme işlemlerinin takibinin gerçekleştirilmesinde yardımcı olmaktadır. Bunların yanında ağ, dosya ve SMS yoluyla bilgi sızıntısının olup olmadığına yönelik analiz sunmaktadır.

Santoku: Analiz ve güvenlik odaklı önyüklenebilir bir Linux dağıtımıdır (Santoku, 2021). Bu işletim sistemi mobil kötücül yazılım analizi, mobil adlı tıp ve mobil güvenlik testi işlemlerinin yapılmasına yardımcı olmaktadır. Kötü amaçlı yazılım analizinde bazı faydaları bulunmaktadır. Bunlar; dinamik analiz için ağ servislerinin simülasyonunu sağlaması, kaynak kodu ayrıştırma ve birleştirme imkânı sunması, kötü amaçlı yazılım veri tabanlarına erişebilmesi ve mobil aygıt emülatörü olması olarak özetlenebilir.

uniPDroid: Androguard aracının genişletilmiş sürümüdür (Fereidooni vd, 2016). Statik analiz aşamasında kullanılan hemen hemen tüm öznitelikler bu araç sayesinde çıkartılabilmektedir. Python programlama dilinde yazılmış olan bu araç Python programlama dili ile kullanılabilmektedir.

JADX: Android işletim sistemine özgün DEX ve APK dosyalarından Java kaynak kodu üretmek için komut satırı ve GUI sunan bir araçtır (JADX, 2021).

2.2. Sınıflandırma Algoritmaları

Bu bölümde, literatürde yer alan Android kötücül yazılım tespit sistemlerinin altyapısında sıkça kullanılan makine öğrenmesi tekniklerine değinilecektir.

2.2.1. K-En Yakın Komşu

Örnek tabanlı bir sınıflandırma algoritmasıdır. Çoğu denetimli makine öğrenmesi algoritmalarının aksine eğitim aşaması bulunmamaktadır. Çeşitli uzaklık metrikleri ile test verisi içerisinde yer alan bir örneğin (etiketi bilinmeyen), eğitim için ayrılan veri kümesi üzerindeki en yakın k tane örneğe bakılması ile etiket tahmini yapılmaktadır. Bu algoritma içerisinde kullanılan bazı uzaklık metrikleri Tablo 2.1'de yer almaktadır. Eğitim kümesinde yer alan $A(x_1, x_2, x_3, \dots, x_n)$ örneği ile test kümesinde yer alan $B(y_1, y_2, y_3, \dots, y_n)$ örneğinin uzaklıkları Tablo 2.1'e göre çeşitli uzaklık metrikleri ile hesap edilebilmektedir.

Tablo 2.1. Bazı uzaklık metrikleri

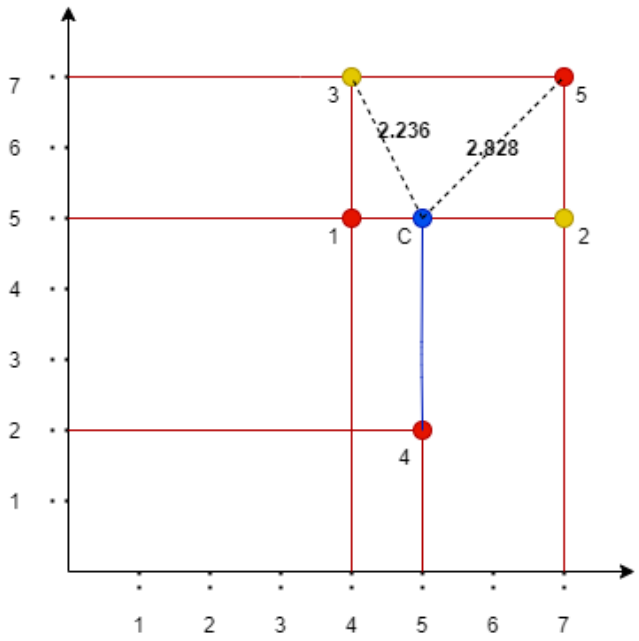
Metrik Adı	Matematiksel Gösterimi
Manhattan Uzaklığı	$\sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Euclidean Uzaklığı	$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
Minkowski Uzaklığı	$(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p)^{\frac{1}{p}}$
Chebyshev Uzaklığı	$_{i=1}^{n}max \lvert x_i - y_i \rvert$

Tablo 2.2. Örnek veri kümesi

Örnek Numarası	Öznitelik 1	Öznitelik 2	Sınıf Etiketi
1	4	5	A sınıfı
2	7	5	B sınıfı
3	4	7	B sınıfı
4	5	2	A sınıfı
5	7	7	A sınıfı

Tablo 2.2'de 5 örnekten oluşan 2 sınıflı bir veri kümesi yer almaktadır. Bu veri kümesi göz önünde bulundurularak sınıf etiketi bilinmeyen $C(5, 5)$ verisinin sınıf etiketi tahmini şöyle hesaplanmaktadır:

Adım 1: $C(5,5)$ verisinin Tablo 2.2'de yer alan her bir örnek için Euclidean uzaklığına bakılır. Tablo 2.2'de yer alan örneklerin ve etiketi bilinmeyen $C(5,5)$ örneğinin iki boyutlu düzlemdeki temsili Şekil 2.5'te yer almaktadır. 1, 4 ve 5 numaralı örnekler A sınıfında yer aldığı için kırmızı renk ile temsil edilirken, 2 ve 3 numaralı örnekler B sınıfında yer aldığı için sarı renk ile temsil edilmektedir. Etiketi bilinmeyen $C(5,5)$ örneği ise mavi renk ile temsil edilmektedir. Şekil 2.5'te görüldüğü gibi her örnek için elde edilen mesafe sonuçları sırasıyla 1, 2, 2.236, 3 ve 2.828'dir.



Şekil 2.5. Tablo 2.2'de verilen örneklerin iki boyutlu uzaydaki temsili

Adım 2: Algoritmadaki k değeri 1 seçilirse $C(5,5)$ verisine en yakın örnek Tablo 2.1'de yer alan 1 numaralı örnek olduğundan $C(5,5)$ verisinin etiketi A olacaktır. KNN algoritmasında, k değerinin ne seçileceği oldukça önemlidir. Örneğin bu örnekte k değeri 2 seçilirse $C(5,5)$ verisine en yakın iki örnek Tablo 2.1'deki 1 ve 2 numaralı örneklerdir. Bu iki örneğin etiketleri farklı olduğundan C(5, 5) verisine hangi etiketin atanacağı belli olmayacaktır. Bu problemden ötürü k değerine genelde tek sayı verilmektedir. k değeri 3 seçilirse $C(5,5)$ verisine en yakın 3 örnek Tablo 2.1'deki 1, 2 ve 3 numaralı örneklerdir. 1 numaralı örneğin etiketi A, 2 ve 3 numaralı örneklerin etiketi B olduğu için çoğunluk oylaması sonucunda $C(5,5)$ verisinin sınıfı B etiketi olarak tahmin edilecektir.

2.2.2. Naive Bayes

Olasılık ilkesine dayanan bir sınıflandırma algoritmasıdır. İstatistikteki Bayes kuralı esas alınarak etiketi bilinmeyen örnekler için çeşitli etiket tahmininde bulunulur. Sınıfı bilinmeyen örneğin etiketi en yüksek olasılık değerine sahip olan etiket olmaktadır. Eşitlik 2.1'de koşullu olasılıklar ile marjinal olasılıklar arasında ilişkiyi gösteren Bayes teorimi verilmektedir.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{2.1}$$

Eşitlik 2.1'de $P(A|B)$, B olayının gerçekleştiği durumda A olayının meydana gelme olasılığını, $P(B|A), A$ olayının gerçekleştiği durumda B olayının meydana gelme olasılığını, $P(A)$ ve $P(B)$ ise sırasıyla A ve B olaylarının marjinal olasılıklarını göstermektedir. Sınıflandırma problemlerinde, veri kümesi çok sayıda öznitelik ve en az iki adet sınıf etiketinden meydana gelmektedir. Bu nedenle Eşitlik 2.1'de verilen ifade genişletilerek Eşitlik 2.2'ye dönüştürülmektedir.

$$P(etiket|\text{öz}n_1, \text{öz}n_2, \dots, \text{öz}n_n) = \frac{P(etiket)P(\text{öz}n_1, \text{öz}n_2, \dots, \text{öz}n_n|etiket)}{P(\text{öz}n_1, \text{öz}n_2, \dots, \text{öz}n_n)} \tag{2.2}$$

Eşitlik 2.2 göz önünde bulundurulursa NB algoritması bütün koşullu olasılıkların çarpımıdır. Tablo 2.3'te temsili bir veri kümesi verilmektedir.

Tablo 2.3. NB algoritması için temsili veri kümesi

Uygulama	İzin-1	İzin-2	İzin-3	İzin-4	Etiket
1	Evet	Hayır	Hayır	Hayır	Kötücül
2	Evet	Evet	Evet	Evet	İyicil
3	Hayır	Hayır	Hayır	Hayır	Kötücül
4	Evet	Evet	Evet	Evet	Kötücül
5	Evet	Hayır	Evet	Hayır	İyicil
6	Hayır	Hayır	Hayır	Hayır	İyicil
7	Evet	Evet	Evet	Evet	Kötücül
8	Hayır	Hayır	Hayır	Hayır	Kötücül
9	Evet	Hayır	Hayır	Hayır	Kötücül
10	Evet	Evet	Evet	Hayır	İyicil

Bu temsili veri kümesinde özniteliklerin uygulama izinleri olduğunu farz edelim. Bu veri kümesinde İzin-1, İzin-2, İzin-3 ve İzin-4 olmak üzere 4 tane öznitelik bulunmaktadır. Ayrıca, 10 tane uygulama bu veri kümesinde yer almaktadır. Uygulama içerisinde ilgili izin varsa "Evet", yoksa "Hayır" şeklinde değerler atanarak temsili veri kümesi oluşturulmaktadır. En son sütun ise veri kümesinin sınıfını yani etiketini göstermektedir.

Tablo 2.3'te verilen veri kümesinin NB algoritması için eğitim kümesi olduğunu farz edelim. İzin-1 = Evet, İzin-2 = Evet, İzin-3 = Hayır ve İzin-4 = Hayır şeklinde etiketi bilinmeyen bir uygulamanın etiketi NB algoritması ile Tablo 2.4'ten yararlanılarak şöyle hesaplanmaktadır:

Tablo 2.4. NB için olasılıklar ve sayılar

	İzin-1		**İzin-2**		**İzin-3**		**İzin-4**	
Etiket	**Kötücül**	**İyicil**	**Kötücül**	**İyicil**	**Kötücül**	**İyicil**	**Kötücül**	**İyicil**
Evet	4	3	2	2	2	3	2	1
Hayır	2	1	4	2	4	1	4	3
Evet/Toplam	4/6	3/4	2/6	2/4	2/6	3/4	2/6	1/4
Hayır/Toplam	2/6	1/4	4/6	2/4	4/6	1/4	4/6	3/4

Adım 1: Etiket = Kötücül için olasılık hesabı Eşitlik 2.3'te verilmektedir.

$$P(etiket = kötücül|sabit) = \frac{P(etiket = kötücül)P(sabit|etiket = kötücül)}{P(sabit)}$$

$$P(sabit|etiket = kötücül) = \left(\frac{4}{6}\right) * \left(\frac{2}{6}\right) * \left(\frac{4}{6}\right) * \left(\frac{4}{6}\right) = \frac{8}{81} \quad \textbf{(2.3)}$$

$$P(etiket = kötücül|sabit) \approx \frac{\left(\frac{8}{81}\right)\left(\frac{6}{10}\right)}{P(sabit)} = \frac{0.0593}{P(sabit)}$$

Adım 2: Etiket = İyicil için olasılık hesabı Eşitlik 2.4'te verilmektedir.

$$P(etiket = iyicil|sabit) = \frac{P(etiket = iyicil)P(sabit|etiket = iyicil)}{P(sabit)}$$

$$P(sabit|etiket = iyicil) = \left(\frac{3}{4}\right) * \left(\frac{2}{4}\right) * \left(\frac{1}{4}\right) * \left(\frac{3}{4}\right) = \frac{9}{128} \quad \textbf{(2.4)}$$

$$P(etiket = iyicil|sabit) \approx \frac{\left(\frac{9}{128}\right)\left(\frac{4}{10}\right)}{P(sabit)} = \frac{0.0281}{P(sabit)}$$

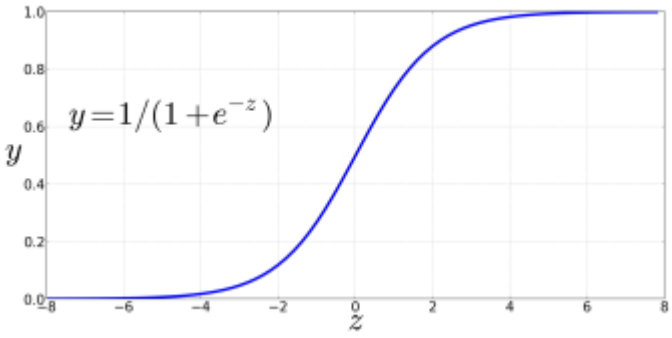
$P(etiket = kötücül|sabit)$ olasılık değeri $P(etiket = iyicil|sabit)$ olasılık değerinden büyük çıktığından İzin-1 = Evet, İzin-2 = Evet, İzin-3 = Hayır ve İzin-4 =

Hayır şeklinde izin listesine sahip bir uygulamanın etiketi NB algoritmasına göre kötücül olarak sınıflandırılmaktadır.

2.2.3. Lojistik Regresyon

Denetimli makine öğrenmesi tekniklerinden olan ikili sınıflandırma problemlerinde çalışabilen istatistiksel modele dayalı bir sınıflandırma algoritmasıdır. Her ne kadar algoritmanın adında regresyon kelimesi geçse de algoritma temel olarak sınıflandırma problemine dayanmaktadır. Lojistik regresyonun amacı, yeni bir girdi gözleminin sınıfı hakkında ikili bir karar verebilen sınıflandırıcı oluşturmaktır. İkili karara bir mailin istenmeyen veya normal olması, uygulamanın iyicil veya kötücül olması ya da bir tümörün zararlı veya zararsız olması örnek olarak verilebilir. LR algoritmasında sınıf etiketi kararının verilmesi sigmoid fonksiyonu aracılığıyla gerçekleştirilmektedir (Zou vd, 2019). Sigmoid fonksiyonu, Eşitlik 2.5'te gösterilmiş olup Şekil 2.6'da verildiği gibidir.

$$y = \frac{1}{1 + e^{-z}} \tag{2.5}$$



Şekil 2.6. Sigmoid fonksiyonu

Sigmoid fonksiyonunun birçok avantajı vardır. Bu avantajlardan en önemlisi nümerik bir değeri alarak bu değere karşılık (0, 1) aralığında değer üretir. Böylece etiketi bilinmeyen bir örneğin etiketinin tahmini kolaylaşacaktır. Sınıf tahmini için bir eşik değer belirlenmektedir. Bu değer genelde 0.5 seçilmektedir. Tahmin bu eşik değerin altında ise sınıf etiketi 0, eşit ve üstünde ise 1 olacaktır.

$(x_1, x_2, \dots, x_n)$ özellik vektörü ile temsil edilen x girdi gözlemi ve sınıflandırıcı çıktısı y 1 veya 0 olacak şekilde verilen bir gözlemin $P(y = 1|x; w)$ olma olasılığı LR ile hesaplanabilmektedir. Özellik veya öznitelikler bir uygulamadaki izinlerin sayısını

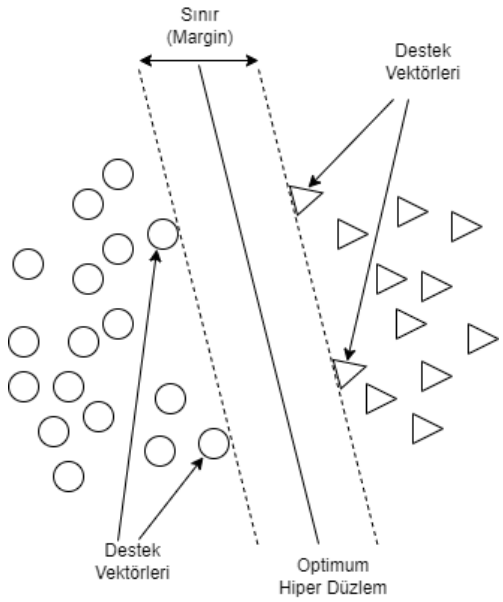
temsil etmektedir. $P(y = 1|x; w)$ uygulamanın iyicil olma olasılığıdır. $P(y = 0|x; w)$ ise uygulamanın kötücül olma olasılığıdır. Bu olasılık değerinin hesaplanabilmesi için eğitim setine, ağırlık vektörüne ve sapma değerine ihtiyaç duyulmaktadır. Her ağırlık $(w = w_1, w_2 \ \dots \ , w_n)$ sayısal değerlerden meydana gelen bir vektör olup $(x_1, x_2, \dots \ , x_n)$ giriş özelliklerinden biri ile ilişkilidir. Kesişim olarak da adlandırılan sapma terimi, ağırlıklı girişlere eklenen başka bir değerdir. Herhangi bir test örneğinin sınıf etiketine karar vermek için, ilk olarak öznitelik değerlerinin olduğu *x* vektörü ile ağırlık değerlerinden oluşan w vektörü çarpılır. Ardından çarpım sonucunda oluşan değer ile sapma değeri toplanır. En son durumda oluşan değer sınıf tahmini yapmak için kullanılmaktadır. Eşitlik 2.6'da bu durum gösterilmektedir.

$$y = (\sum_{i=1}^{n} w_i * x_i) + b \qquad \textbf{(2.6)}$$

Eşitlik 2.6'da n öznitelik sayısını, b sapma değerini, y ise sınıf etiket değerini göstermektedir. Lojistik regresyonun doğrusal regresyondan farkı doğrusal regresyonda hatalarının toplamı en aza indirilerek parametrelerin hesaplanması gerçekleştirilmektedir. Lojistik regresyonda ise örneklere ait olasılık değerini en yükseğe çıkaran parametrelerin seçilmesi sağlanmaktadır. Maksimum olabilirlik kestirimi, gradyan inişi algoritmaları lojistik regresyonda parametrelerin hesaplanmasında kullanılan yöntemlerden bazılarıdır (Zou vd, 2019).

2.2.4. Sıralı Minimum Optimizasyon ve Destek Vektör Makinesi

SVM, hem sınıflandırma hem de regresyon problemleri için kullanılan istatistiksel bir sınıflandırma algoritmasıdır (Cortes ve Vapnik, 1995). Bu denetimli öğrenme algoritması, başlangıçta ikili sınıflandırma problemleri için tasarlanmıştır, ancak çok sınıflı sınıflandırma problemlerine de uygulanabilmektedir. SVM'nin genel yapısı Şekil 2.7'de gösterilmektedir. SVM, hiper düzlem ile en yakın örnek arasındaki mesafeyi maksimize ederek farklı sınıfların örneklerini ayırmak için en uygun hiper düzlemleri bulmayı amaçlamaktadır (Cortes ve Vapnik, 1995). SVM, yüksek boyutlu özelliklerin yanı sıra seyrek verileri işleme yeteneği nedeniyle birçok sınıflandırma problemlerinde sıkça tercih edilmektedir. Ancak büyük ve doğrusal olmayan veriler SVM ile sınıflandırıldığında hesaplama maliyeti fazla olabilmektedir.



Şekil 2.7. SVM'nin gösterimi

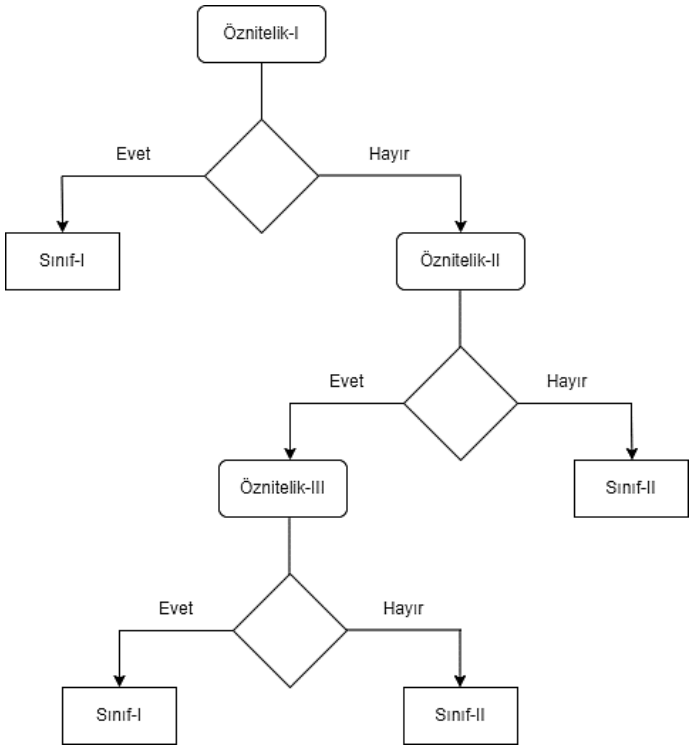
SVM'nin yapısında doğrusal çekirdek, polinom çekirdeği ve RBF gibi farklı çekirdek fonksiyonları kullanılabilmektedir. Polinom çekirdeği ve RBF, verileri düz bir karar sınırı kullanarak ayıran doğrusal çekirdeğin aksine, doğrusal olmayan çekirdek fonksiyonlarıdır. Polinom çekirdeği ve RBF farklı sınıfların doğrusal olarak ayrılamayan özelliklerine uyması için eğri karar sınırlarını kullanmaktadır.

SMO algoritması, standart ikinci dereceden fonksiyonu çözmek için daha verimli optimizasyon algoritmaları kullanarak destek vektör makinesinin eğitim aşamasını hızlandırmaya çalışmaktadır (Platt, 1998). Eğitim verisini analitik olarak çözülebilen daha küçük problemlere bölmek için sezgisel yöntemler kullanır. Weka aracında (Hall vd, 2009) yer alan destek vektör makinesi SMO algoritmasını kullanmaktadır.

2.2.5. Karar Ağaçları ve Rastgele Orman

Karar ağacı, sınıflandırma ve tahmin problemlerinde sıkça tercih edilen popüler yaklaşımlardan bir tanesidir. Karar ağacında, her bir düğüm öznitelik üzerindeki testi ifade etmektedir. Bu ağaç yapısında her dal testin bir sonucunu temsil etmektedir. Buna karşın her yaprak düğüm bir sınıf etiketine sahip olmaktadır. Genel olarak bu ağaç yapısı bir akış şemasına benzemektedir. Ağaç oluşturulduktan sonra çeşitli kurallar ile sınıf etiketlerinin tahmini yapılabilmektedir. Şekil 2.8'de 3 tane özniteliğe ait temsili bir karar ağacı yapısı yer almaktadır. Bu temsili ağaç yapısına göre test edilecek bir veri kümesinde Öznitelik-I varsa test edilen örneğin etiketi Sınıf-1 olarak

sınıflandırılacaktır. Test edilecek örnekte Öznitelik-I ve Öznitelik-II bulunmuyorsa bu örneğin etiketi Sınıf-II olarak sınıflandırılacaktır. Bu şekilde eğitim kümesinden elde edilen karar ağacına göre kurallar oluşturulup test örneklerinin sınıf etiketleri belirlenmektedir. Karar ağacının oluşturulmasında bilgi kazancı ve Gini indeksi gibi yöntemler kullanılabilmektedir.



Şekil 2.8. Karar ağaçlarının genel çalışma yapısı

Rastgele orman sınıflandırıcısı ise farklı şekil ve boyutlara sahip farklı türdeki karar ağaçlarından oluşan bir topluluk öğrenmesi modelidir. Standart ağaçlarda her düğüm, tüm değişkenler arasında en iyi bölünme kullanılarak gerçekleştirilir. Buna karşın rastgele orman algoritmasında, her düğüm, o düğümde rastgele seçilen bir tahminin alt kümesi arasından en iyisi seçilerek bölünür.

2.2.6. Yapay Sinir Ağları ve Derin Öğrenme Yaklaşımları

Artan veri boyutu ve bilgisayarların artan işlem kapasitesi ile birlikte günümüzde artık derin öğrenme adı verilen teknikler kullanılmaktadır. Geleneksel makine öğrenimi teknikleri, verileri ham formlarında işleyecek şekilde sınırlı tekniklerdir (Yann LeCun vd, 2015). Derin öğrenme geleneksel yöntemlerin aksine, öğrenmenin veri temsillerine dayandığı daha geniş bir makine öğrenimi yöntemleri ailesidir. Geleneksel yöntemlere kıyasla derin öğrenme yöntemleri daha fazla soyut bilgi öğrenmek için derin mimariler oluşturma avantajına sahiptir.

Yapay Sinir Ağları, bir girdi ve bir çıktı katmanından oluşan ve bu iki katman arasında bir tane gizli katmanın bulunduğu sığ ağlara dayanmaktadır. Derin öğrenme ağlarında ise girdi ve çıktı katmanları da dâhil olmak üzere üçten fazla katman bulunduğunda nitelikli modeller elde edilmektedir (Dargan vd, 2020). Bu nedenle katman sayısı arttıkça ağ derinleşmektedir. Derin sinir ağları, çeşitli şekillere sahip yapay sinir ağlarının giriş katmanları ile çıkış katmanları arasına gizli katmanların eklenmesi ile oluşturulabilir. Bu katmanlar farklı kavram düzeylerine karşılık gelir ve üst düzey kavramlar alt düzey kavramlar kullanılarak tanımlanır (Berman vd, 2019). Yapay sinir ağlarında yalnızca bir gizli katman bulunduğundan, gelişmiş özellik çıkarma becerisinden yoksundurlar ve derin sinir ağlarının öğrenebileceği üst düzey kavramları öğrenemezler. Bu aynı zamanda diğer makine öğrenimi algoritmaları için de geçerlidir. Bir yapay sinir ağında veya DNN'de önceki katmandaki birimlerin ağırlıklı toplamına doğrusal olmayan bir işlem uygulanır ve bu işlem aktivasyon fonksiyonları ile yapılır. Bunun için kullanılabilecek birçok fonksiyon vardır ancak en yaygın olanları sigmoid, softmax, tanjant hiperbolik ve ReLu fonksiyonlarıdır.

Derin öğrenme yöntemleri doğrusal olmayan ilişkileri modelleyebilir ve bu yöntemler arasında en popülerlerinden bir tanesi CNN'dir. 1990'lı yıllarda (Y. Lecun vd, 1998) tarafından yapılan çalışmada, CNN'ler üzerinde gradyan tabanlı bir öğrenme algoritması uygulayarak el yazısı rakam sınıflandırması problemi için başarılı sonuçlar elde etmişlerdir. CNN'in DNN'lere göre görüntüleri işlemek için daha yüksek oranda optimize edilmiş olmaları ve görüntüler üzerindeki soyut özellikleri öğrenmede daha etkili olmaları gibi avantajları bulunmaktadır (Alom vd, 2019). Ek olarak, bağlı ağırlıklara sahip seyrek bağlantılardan oluşan bir CNN benzer boyuttaki tam bağlı bir derin ağdan çok daha az parametreye sahiptir. CNN'lerin genel mimarisi evrişimli katmanlar, havuzlama katmanları ve sınıflandırma olmak üzere üç katmandan oluşmaktadır. Geleneksel bir yapay sinir ağında her gizli katman ağırlıklar, girdi ve çıktı içerir. Ancak görüntülerin 2 boyutlu yapısı nedeniyle CNN'deki her nöron çekirdek (kernel) olarak bilinen ağırlıklar için 2 boyutlu düzlemler ve özellik haritası olarak adlandırılan girdileri ve çıktıları içerir (Kiranyaz vd, 2021). Evrişim katmanlarındaki düğümler giriş görüntülerinden özellikleri çıkartmaktadır. Giriş görüntüleri ağa verildiğinde evrişim ve havuzlama katmanlarının çıktıları özellik haritası adı verilen 2 boyutlu bir düzlemde gruplanır. Bir katmandaki her bir düzlem önceki katmanlardan elde edilen çıktıların birleşiminden oluşturulmaktadır. Bu şekilde

görüntülerdeki daha yüksek seviyeli (görüntünün ayrıştırılmasını sağlayan belirgin özellikleri) alt seviyedeki katmanlardan yayılarak türetilmektedir. Özellikler üst katmanlara yayıldıkça, özellik boyutları, evrişim ve havuzlama katmanlarında kullanılan filtre boyutuna göre küçülmektedir. Ancak sınıflandırma başarısını artırmak için giriş görüntülerinden en iyi özellikleri sağlamak adına özellik haritalarının sayısı artmaktadır. Havuzlama katmanları, oluşturulan modellerde genellikle evrişim katmanlarından sonra kullanılmaktadır. Evrişim ve havuzlama işlemleri yapıldıktan sonra elde edilen özellikler bir vektöre dönüştürülerek sınıflandırma performansının daha iyi olduğu bilinen tam bağlantılı katmanlar kullanılarak yapılmaktadır (Alom vd, 2019).

1D-CNN modelleri ise sınırlı sayıda etiketli veriye ve farklı kaynaklardan elde edilen sinyaller üzerindeki uygulamalarda oldukça iyi performans göstererek son zamanlarda ön plana çıkmaktadır (Kiranyaz vd, 2021). Uygulaması 2D-CNN ile çok benzerdir. Tek fark işlemlerin 2 boyut yerine tek boyut kullanılarak gerçekleştirilmesidir. Derin öğrenme ağları genellikle geri yayılım algoritması olarak bilinen bir yöntem kullanılarak eğitilmektedir. Algoritmanın her adımında evirişim veya tam bağlı katmanların ağrılıkları gibi her ağ parametresinin gradyan büyüklüğü hesaplanır. Bu değerler devamında belirli bir durdurma kriterine ulaşana kadar model parametrelerini güncellemek için kullanılır. Literatürde geri yayılım algoritması için kullanılmakta olan Stokastik Gradyan İnişi (SGD), Momentumlu SGD, AdaGrad, RMSProp ve Adam gibi gradyan-iniş optimizasyon yöntemleri bulunmaktadır.

2.3. Performans Ölçütü

Makine öğrenmesinde kullanılan sınıflandırma algoritmaların performans değerlendirmeleri Tablo 2.5'te verilen karmaşıklık matrisi kullanılarak elde edilmektedir. Karmaşıklık matrisi gerçek sınıf ve tahmin edilen sınıf arasındaki ilişkiyi göstermektedir.

Tablo 2.5. Karmaşıklık matrisi örneği

		Tahmini Sınıf	
		Kötücül	**İyicil**
Gerçek Sınıf	**Kötücül**	Doğru Negatif	Yanlış Pozitif
	İyicil	Yanlış Negatif	Doğru Pozitif

Veri kümesinin sınıflandırılması sonucunda dört muhtemel sonuç ortaya çıkmaktadır. Bunlar Doğru Pozitif (DP), Doğru Negatif (DN), Yanlış Pozitif (YP) ve Yanlış Negatif (YN) durumlarıdır. DP, gerçekte iyicil sınıfında olup sınıflandırma sonucunda iyicil ile sınıflandırılan örneklerin sayısıdır. DN, gerçekte kötücül sınıfında olup sınıflandırma sonucunda kötücül ile sınıflandırılan örneklerin sayısıdır. YP, gerçekte kötücül sınıfında olup sınıflandırma sonucunda iyicil ile sınıflandırılan örneklerin sayısıdır. Son olarak, YN ise gerçekte iyicil sınıfında olup sınıflandırma sonucunda kötücül ile sınıflandırılan örneklerin sayısıdır.

DP, DN, YP ve YN değerlerinden yararlanılarak Eşitlik 2.7'de doğruluk, Eşitlik 2.8'de kesinlik ve Eşitlik 2.9'da ise duyarlılık metriklerinin matematiksel gösterimleri verilmektedir. Eşitlik 2.8 ve Eşitlik 2.9'un harmonik ortalaması olan F-ölçütü ise Eşitlik 2.10'da gösterilmektedir. Tablo 2.5'e göre, Eşitlik 2.8, Eşitlik 2.9 ve Eşitlik 2.10 yalnızca bir sınıf için hesaplanan değerlerdir. Benzer şekilde bu metriklerin diğer sınıf içinde değerleri hesaplanarak bu metriklere ait ortalama sonuçlar ile sınıflandırma algoritmalarının performansları elde edilmektedir.

$$doğruluk = \frac{\mathrm{DP} + \mathrm{DN}}{\mathrm{DP} + \mathrm{DN} + \mathrm{YP} + \mathrm{YN}} \tag{2.7}$$

$$kesinlik = \frac{\mathrm{DP}}{\mathrm{YP} + \mathrm{DP}} \tag{2.8}$$

$$duyarlılık = \frac{\mathrm{DP}}{\mathrm{YN} + \mathrm{DP}} \tag{2.9}$$

$$F - \text{ölçütü} = \frac{2 \times kesinlik \times duyarlılık}{\text{kesinlik} + \text{duyarlılık}} \tag{2.10}$$

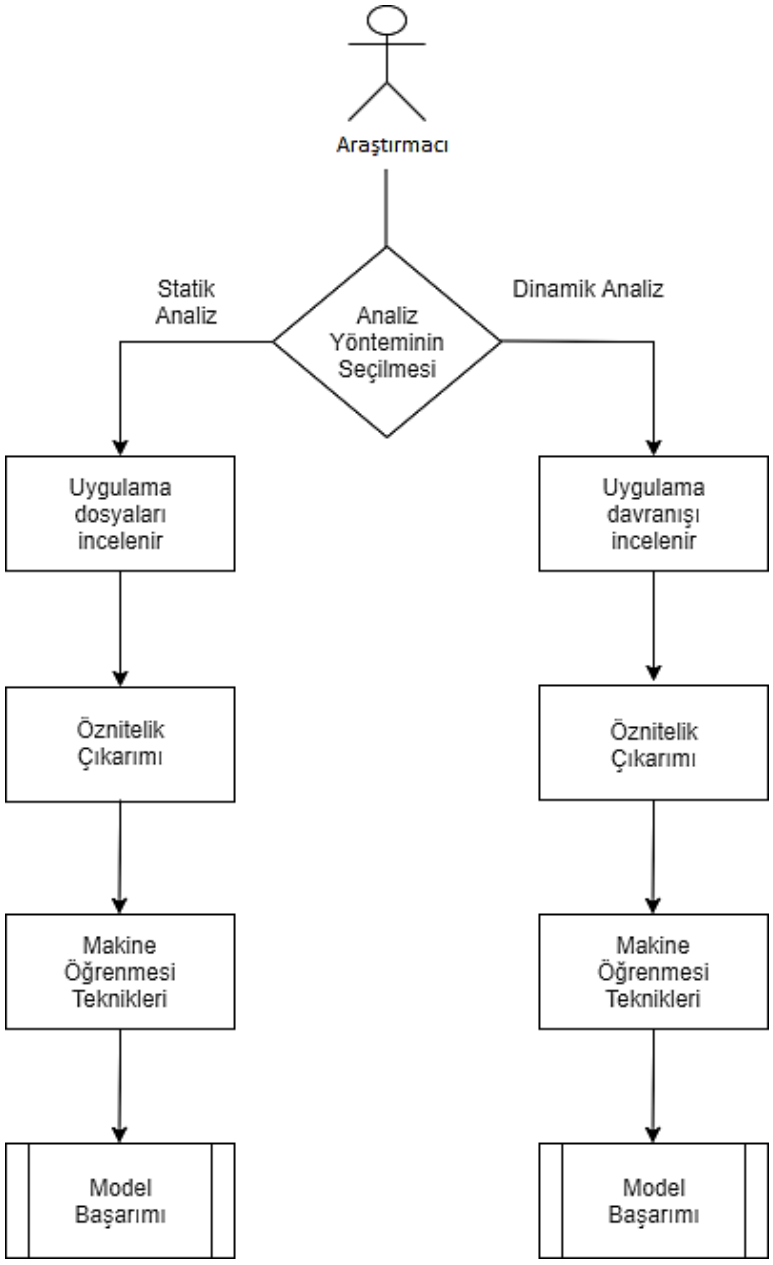
Doğruluk metriği, doğru olarak sınıflandırılmış örnek sayısının toplam örnek sayısına olan oranını ifade etmektedir. Sınıf örnek sayıları arasında dengesizlik olduğunda, doğruluk metriği yerine kesinlik ve geri çağırma metriklerinin harmonik ortalaması olan F-ölçütü metriğini kullanmak daha iyi yaklaşımdır.

2.4. Literatür Araştırması

Android işletim sistemlerine özgü kötücül yazılım tespitinde araştırmacıların genelde izledikleri yol Şekil 2.9'da verilmektedir. Şekil 2.9'da verilen yapıya göre araştırmacılar öncelikle tasarlanması istenilen mimarinin hangi analiz yöntemine göre

çalışacağına karar vermektedirler. Bu seçimin ardından veri kümelerinde yer alan uygulamalara tercih edilen analiz yaklaşımına göre ilgili araçlar vasıtasıyla çeşitli işlem adımları uygulanarak öznitelik çıkarımı yapılmaktadır. Öznitelik çıkarımının yapılmasıyla Android uygulama dosyalarından oluşan veri kümesi yapısal hale getirilecektir. Bu sayede oluşturulan yapısal veri, makine öğrenmesi algoritmasının anlayabileceği şekle dönüşmüş olacaktır. Son olarak önerilen modelin başarımı hesaplanacaktır.

Araştırmacılar kötücül yazılım sisteminde sınıflandırma başarımını arttırmak için çoğunlukla öznitelik çıkarımı ve makine öğrenimi aşamalarında yenilikler sunmaktadır. Öznitelik çıkarımı aşamasında genellikle iyicil yazılımlar ile kötücül yazılımlar arasında farklılıklar gösteren özniteliklerin bulunması amaçlanmaktadır. Makine öğrenimi aşamasında ise klasik makine öğrenimi yaklaşımlarının yanında derin sinir ağları ve topluluk öğrenmesine dayalı modellerin tasarlanması gerçekleştirilerek sınıflandırma performansının arttırılması araştırmacılar tarafından hedeflenmektedir. Bu gibi gelişmelerin yanında mobil cihaz üzerinde hızlı ve verimli bir şekilde çalışan makine öğrenimi algoritmalarının tasarlanması da ayrı bir araştırma konusudur. Ayrıca, sınıflandırma modellerinin daha hızlı ve verimli oluşturulması için öznitelik seçme teknikleri son yıllarda bu alanda popüler araştırma konularından biri olmuştur. Android kötücül yazılım tespiti özelinde ele alınan bazı çalışmalar şöyle özetlenmektedir:



Şekil 2.9. Android kötücül yazılım tespitinde araştırmacıların izledikleri yol

(Wu vd, 2012) DroidMat adını verdikleri çerçeve ile Android kötücül yazılımların tespit edilmesini amaçlamaktadır. Dinamik analizin maliyetinden ötürü statik analiz tercih edilmektedir. Çalışmada izinler, API çağrıları başta olmak üzere çeşitli statik özellikler kullanılmaktadır. K-ortalama ve maksimum beklenti kümeleme algoritmaları kullanılarak farklı kötücül yazılım grupları oluşturulmaktadır. Başlangıçtaki küme sayısına tekil değer ayrışımı aracılığıyla karar verilmektedir. Bu aşamanın ardından KNN algoritması kullanılarak uygulamanın kötücül veya iyicil olarak sınıflandırılması yapılmaktadır. Önerilen yöntem çeşitli kötücül yazılım ailelerinden alınan 238 kötücül uygulama ve 1500 iyicil uygulama üzerinde test

edilmektedir. Çalışmanın sonuçları incelendiğinde K-ortalama ve KNN algoritmaları birlikte kullanıldığında en yüksek sınıflandırma başarımına ulaşılmaktadır. Doğruluk metriğine göre DroidMat çerçevesinin en yüksek başarımı 0.9787 olarak bulunmaktadır.

(Suarez-Tangil vd, 2014) DENDROID adını verdikleri sistemde, uygulamaların kaynak kodlarını metin madenciliği ve bilgi getirme teknikleriyle analiz ederek Android kötücül yazılımları türlerine ayrıştırmaktadırlar. Mobil kötücül yazılım tespitinde, bu çalışmanın metin madenciliği ile kaynak kod analizini gerçekleştiren ilk çalışma olduğu vurgulanmaktadır. İncelenen her uygulama bir dokuman olarak temsil edilirken, uygulamaya ait kodlarda kelime gibi temsil edilmektedir. Her uygulama ve kod bir vektöre dönüştürüldükten sonra analiz aşamasına geçilmektedir. Bu aşamada kötü amaçlı yazılım ailelerinin filogenetik ağaçlar olarak anlaşılabilmesi için hiyerarşik kümeleme tekniği olan dendrogramlar kullanılmaktadır. Dendogram kullanılmasıyla kötücül yazılım aileleri arasındaki ortak özelliklerin çıkarılması hedeflenmiştir. Bu ortak özelliklere belirli kod bloklarının veya satırların benzerlikleri örnek olarak verilebilir.

(Fereidooni vd, 2016) ANASTASIA adını verdikleri çerçeve ile Android kötücül yazılımların tespit edilmesini amaçlamaktadır. Çalışmada statik analiz tekniği tercih edilmektedir. Python programlama dili kullanılarak uniPDroid adında bir araç geliştirilmektedir. Bu araç ile statik öznitelikler çıkartılmaktadır. Niyet filtreleri, izinler, sistem komutları, API çağrıları, IMEI okuma ve soket açma gibi kötücül aktivite oluşturabilecek özellikler kullanılmaktadır. Rastgele karar ağacı vasıtasıyla öznitelik seçimi yapılmaktadır. XGboost, Adaboost, RF, SVM, KNN, LR, NB ve derin öğrenme algoritmaları kullanılarak Android uygulamaların sınıflandırılması sağlanmaktadır. Önerilen yöntem 18677 kötücül ve 11187 iyicil uygulama üzerinde test edilmektedir. Çalışmada en iyi sınıflandırma sonucu XGboost algoritması kullanılarak elde edilmektedir. Bu sonuç doğru pozitif oranına göre %97.3 olarak raporlanmaktadır.

(Arp vd, 2014) Drebin adını verdikleri çerçevede statik analiz ve makine öğrenmesi yaklaşımlarını birleştirerek Android kötücül yazılım tespitinde bulunmaya çalışmaktadır. Drebin, uygulamanın kaynak kodunu ve AndroidManifest.xml dosyasını kullanarak uygulamaya ait izinler, API çağrıları ve ağ adresleri gibi çeşitli özelliklerden yararlanmaktadır. Bu özellikler, vektör uzay modeline dönüştürülerek

SVM algoritması aracılığıyla Android kötücül yazılım tespiti gerçekleştirilmektedir. Drebin yöntemi 5560 kötücül ve 123453 iyicil uygulama üzerinde test edilmektedir. Çalışmanın sonuçlarına göre Drebin çerçevesinin %93.90 oranında başarım verdiği görülmektedir. Aynı veri kümesi 10 tane antivirüs programı kullanılarak sınıflandırıldığında Drebin yönteminin bu antivirüs programlarının sadece bir tanesinden kötü sonuç verdiği gözlemlenmektedir.

(Burguera vd, 2011) önerdikleri Crowdroid isimli çerçeve ile anormal davranış gerçekleştiren Android uygulamaların tespit edilmesi sağlamaktadır. Çalışma davranış temelli dinamik analiz tekniğine dayanmaktadır. Crowdroid'in altyapısında, sistem çağrılarını toplamak için Linux'ta bulunan "Strace" adlı bir araçtan yararlanılmaktadır. Bu araç ile çok sayıda sistem çağrısı değerlendirilebilmektedir. Linux işletim sisteminde her sistem çağrısına ait eşsiz bir numara bulunmaktadır. Bu sayede bir uygulamanın hangi sistem çağrılarını kullandığı ortaya çıkartılmış olacaktır. Böylece kötücül aktivitelere yol açan sistem çağrılarının kullanılıp kullanılmaması durumuna göre uygulamaların iyicil veya kötücül olarak sınıflandırılması gerçekleştirilmektedir.

(Abdullah vd, 2017) önerdikleri yaklaşımda Android işletim sistemi kullanıcılarına yönelik botnet saldırı tiplerinden olan kötücül yazılımlar ile iyicil yazılımlar sınıflandırılmaktadır. Çalışmada öznitelik olarak uygulamaların talep ettikleri izinler kullanılmaktadır. Özellik vektörünün oluşturulmasından sonra filtrelemeli öznitelik seçme yöntemlerinden biri olan bilgi kazancı ile ayırt edici izinlerin seçilmesi gerçekleştirilmektedir. Tüm izinler arasından en yüksek bilgi kazancı değerine sahip 20 izin seçilmektedir. Bu 20 izin içerisinden SET_ALARM ve PACKAGE_USAGE_STATS izinlerinin ayrım gücü olmadığı düşünülerek bu izinlerde özellik vektöründen çıkartılarak özellik vektörünün boyutu 18 izne indirgenmektedir. Bu adımların uygulanmasıyla, izinlerin sayısında %87 oranında azalma sağlanmaktadır. Sınıflandırma adımında NB, RF ve C4.5 algoritmaları kullanılmaktadır. Önerilen yöntem 1505 kötücül ve 850 iyicil uygulama üzerinde test edilmektedir. Çalışmada en iyi sınıflandırma sonucu RF algoritması kullanılarak elde edilmektedir. Bu sonuç doğru pozitif oranına göre %94.6 olarak raporlanmaktadır.

(Sanz vd, 2013) PUMA adını verdikleri çerçevede uygulamanın izinlerinden elde edilen bilgileri kullanan makine öğrenimine dayalı bir Android kötü amaçlı yazılım algılama yöntemi önermektedirler. PUMA çerçevesini değerlendirmek için, Android Market'ten çeşitli kategorilerde 1811 iyicil uygulama ve VirusTotal veri

tabanından 249 benzersiz kötü amaçlı uygulama kullanılmaktadır. Bazı izinlerin hem iyicil uygulamalarda hem de kötücül uygulamalarda sıklıkla görüldüğü öne sürülerek çeşitli izin karşılaştırmaları yapılmaktadır. Bu karşılaştırmalar göz önünde bulundurulup her bir uygulamayı temsil etmek için uygulamanın talep ettiği izinler ve cihazın özellikleri bilgisi kullanılarak özellik vektörü oluşturulmaktadır. Özellik vektörleri çok sayıda makine öğrenmesi algoritmasına girdi olarak verilip çeşitli sınıflandırma başarımları elde edilmektedir. Bütün sonuçlar incelendiğinde en başarılı sınıflandırma algoritmasının RF olduğu tespit edilmektedir. Çalışmada elde edilen en yüksek başarım doğruluk metriğine göre %86.41'dir.

(Suleiman Y Yerima vd, 2013) Bayes sınıflandırmasını kullanarak bir Android kötü amaçlı yazılım algılama yöntemi önerdi. Çalışmalarında, optimum sınıflandırma performansını elde etmek için karşılıklı bilgi yöntemi ile özellik sıralama ve seçme işlemini gerçekleştirmişlerdir. Statik kod analizi ile uygulamalardan toplam 58 kod tabanlı özellik elde etmişlerdir. Öznitelik seçim sürecinden sonra Bayes sınıflandırıcısı ile yaptıkları deneylerde optimum performansı elde etmek için 15-20 özniteliğin yeterli olduğunu bildirmişlerdir. Deneylerinde elde ettikleri en yüksek doğruluğu 0.92 olarak bildirmişlerdir.

(Damshenas vd, 2015) davranış temelli yeni Android kötücül yazılım tespit etme aracı önermişlerdir. Önerilen yöntem dinamik analiz olup iki aşamadan meydana gelmektedir. Birinci aşama sunucu tarafı olurken ikinci aşama istemci tarafıdır. Sunucu tarafında test edilecek uygulamanın analizi gerçekleştirilerek uygulama hakkında çeşitli bilgiler elde edilmektedir. Bilgilerin elde edilebilmesinde Linux sistem çağrılarından yararlanılmaktadır. Bunun nedeni ise uygulamaların benzersiz bir imza veri tabanını oluşturmak içindir. İmzalar üzerinde z-skoru normalizasyon tekniği ve Spearman sıra korelasyon katsayısı uygulanarak veri tabanındaki mevcut imzalarla test edilecek uygulamanın veritabanları karşılaştırılmaktadır. İstemci tarafı, yani mobil cihazın üzerinde çalışan kısımsa mobil cihaza yeni bir uygulama yüklendiğinde devreye girerek sunucuyu tetiklemeye çalışan çok fazla hesaplama gücü gerektirmeyen yapıdadır.

(Yildiz ve Doğru, 2019) statik özelliklerden olan uygulama izinlerini öznitelik olarak kullanmaktadır. Özellik vektörünün oluşturulmasından sonra sarmalamalı öznitelik seçme yöntemlerinden olan genetik algoritma ile ayırt edici izinlerin seçilmesi gerçekleştirilmektedir. Sınıflandırma adımında Naive Bayes, karar ağaçları

ve SVM algoritmaları kullanılmaktadır. Önerilen yöntem 1119 kötücül ve 621 iyicil uygulama üzerinde test edilmektedir. Bu uygulamalardan toplamda 152 izin çıkartılmaktadır. Öznitelik seçme adımı yapılmadan yani 152 izin ile sınıflandırma yapıldığında elde edilen en iyi başarım SVM algoritması kullanılarak bulunurken bu sonuç F-ölçütü metriğine göre 0.959'dur. Çalışmada elde edilen en yüksek başarım genetik algoritma ile 16 izin seçilip SVM algoritması ile sınıflandırma yapıldığında bulunmaktadır. Bu sonuç F-ölçütü metriğine göre 0.981'dir.

(Bhattacharya vd, 2019) Android işletim sistemine özgü kötücül yazılımların tespitinde öznitelik olarak uygulama izinlerini kullanmaktadır. Çalışmanın diğer izin tabanlı kötücül yazılım tespit yaklaşımlarından farkı öznitelik seçme aşamasında kaba küme teorisi ve parçacık sürü optimizasyon algoritmasının kullanılmasıdır. Parçacık sürü optimizasyon algoritması üzerinde iyileştirme yapılarak daha iyi öznitelik seçme işlemi gerçekleştirilmektedir. PSO algoritması metasezgisel bir algoritmadır. Metasezgisel algoritmaların hesaplama maliyetleri oldukça fazladır. Bu nedenle öznitelik seçiminin metasezgisel bir algoritma ile yapılması önerilen yöntemin dezavantajı olarak kabul edilebilir. Ancak, PSO algoritmasının hesaplama maliyeti genetik algoritma gibi diğer metasezgisel algoritmalara göre daha az maliyetli olduğu için araştırmacılar bu algoritma üzerine yoğunlaşmaktadırlar. Önerilen yöntemin başarımını değerlendirmek için Android uygulama dosyalarından oluşan veri kümeleri haricinde farklı veri kümeleri de kullanılmaktadır. Android veri kümelerinin ilkinde 504 iyicil ve 213 kötücül uygulama vardır. Bu veri kümesi toplam 82 izinden oluşmaktadır. Önerilen yöntemle öznitelik sayısı 32'ye inmektedir. 32 izinle sınıflandırma yapıldığında F-ölçütü metriğine göre 0.911875 başarım elde edilmektedir. İkinci veri kümesinde ise 2500 iyicil ve 1150 kötücül uygulama yer almaktadır. Bu veri kümesi toplam 88 izinden oluşmaktadır. Önerilen yöntemle öznitelik sayısı 16'ya inmektedir. 16 izinle sınıflandırma yapıldığında F-ölçütü metriğine göre 0.9785 başarım elde edilmektedir.

(H. Yuan vd, 2020) hem kötücül yazılımların tespit edilmesi hem de kötücül yazılımların ailelerine göre sınıflandırılması işlemlerini gerçekleştirmektedir. Metin sınıflandırmada kullanılan TF-IDF tekniği uygulama izinleri üzerine uygulanarak izinlerin ağırlıklandırılması yapılmıştır. Ardından çeşitli makine öğrenmesi teknikleri ile sınıflandırma adımına geçilmektedir. Çalışmada NB, Bayes ağları, C4.5, Rastgele ağaç, RF ve KNN olmak üzere 6 farklı sınıflandırma algoritması kullanılmaktadır.

Önerilen yaklaşımı değerlendirmek için 6070 iyi huylu uygulama ve 9419 kötü amaçlı yazılım kullanılmıştır. Sistemin doğruluğunu tam olarak gösterebilmek için veri kümesi çeşitli gruplara ayrılmaktadır. Kötücül yazılımlar ailelerine göre sınıflandırıldığında elde edilen başarım 0.99'dan fazladır. Ayrıca kötücül yazılım tespitindeki başarımında 0.99'dan fazla olduğu raporlanmaktadır.

(Amin vd, 2020) statik analiz tekniğine dayalı uçtan uca derin öğrenme mimarilerini kullanan Android kötücül yazılım tespit sistemi önermektedirler. Uygulama dosyaları içinde bulunan .dex dosyalarından elde edilen bytecode'lardan çıkartılan opcode'lar çeşitli derin öğrenme ağlarına verilmektedir. Sınıflandırma için başlıca kullanılan algoritma çift yönlü LSTM derin öğrenme tekniğidir. Çift yönlü LSTM tekniği ile karşılaştırma yapmak için CNN, Derin İnanç Ağları, RNN ve LSTM gibi farklı derin öğrenme algoritmaları da kullanılmaktadır. Algoritmalar arasında en yüksek başarıma çift yönlü LSTM tekniği ile ulaşılmış olup bu sonuç doğruluk metriğine göre 0.999'dur.

(Bai vd, 2020) hem kötü amaçlı yazılım tespiti hem de aile sınıflandırması yapabilen bir çerçeve önermektedir. AndroidManifest.xml dosyasından alınan izinler ile class.dex dosyasından elde edilen opcode dizileri birleştirilerek özellik vektörü oluşturulmaktadır. İki özellik arasındaki korelasyonu ölçmek için simetrik bir belirsizlik kullanan özellik vektörlerine hızlı korelasyon tabanlı filtre algoritmasını uygulayarak boyut azaltması yapılmaktadır. Deneylerde 100, 200, 300, 400 ve 500 özniteliği seçerek geniş kapsamlı değerlendirmeler yapılmaktadır. 500 öznitelik ile en iyi sonucu elde etmişlerdir. Sınıflandırma işlemleri için Yandex tarafından 2017 yılında açık kaynak haline getirilen CatBoost (Dorogush vd, 2018) sınıflandırıcısı kullanılmıştır. İki farklı veri seti kullanılarak yapılan deneylerde kötü amaçlı yazılım tespiti için 0.974 ve aile sınıflandırması için 0.9738 ile en iyi doğruluğun elde edildiği bildirilmektedir.

(Alazab vd, 2020) uygulama izinlerini ve API çağrılarını birleştiren sınıflandırma tabanlı bir Android kötü amaçlı yazılım algılama modeli önermektedir. İyi huylu ve kötü niyetli uygulamalarda API çağrılarını kullanım sıklıklarına göre gruplayarak, ayrım gücü açısından en değerli API çağrılarını ortaya çıkarmayı hedeflemektedirler. API çağrılarının kötü amaçlı uygulamalardaki görünümüne göre üç grup oluşturulmaktadır: belirsiz API çağrıları (neredeyse aynı sayıda iyi ve kötü niyetli uygulamada görülür), riskli API çağrıları (kötü amaçlı uygulamalarda iyi huylu

uygulamalara göre daha sık görülür) ve yıkıcı API çağrıları (yalnızca kötü amaçlı uygulamalarda görülür). Gruplama işlemi ile birlikte bilgi kazancı algoritmasını kullanarak, değerli özellik alt grubunun seçilmesi sağlanmaktadır. Son olarak, TF algoritması boyut indirgeme gerçekleştirilmektedir. Toplamda 27891 uygulama içeren bir veri seti kullanarak 5 farklı sınıflandırıcı ile deneyler gerçekleştirilmektedir. Deneysel sonuçlarında elde edilen en yüksek sonucu F-ölçütü metriğine göre 0.943 olarak bildirilmektedir.

(Salah vd, 2020) URL bilgisi, donanım bilgileri, izinler ve API çağrıları gibi farklı statik özellikleri birleştirerek Android kötü amaçlı yazılım tespit yöntemi önermektedir. Metin madenciliğinde sıklıkla kullanılan Terim Frekansı-Ters Belge Frekansı (TF-IDF) algoritmasını kötü amaçlı yazılım tespit alanına uyarlayarak, Özellik Frekansı Uygulama Frekansı (FF-FA) adı verilen frekans tabanlı özellik seçim yöntemini sunmaktadırlar. Özellik seçiminden geçen yüksek frekanslı özellikleri uygulama URL'lerinden elde ettikleri URL_score ile birleştirerek sonuç özellik vektörünü elde ettiler. Destek Vektör Makinesi, Lojistik Regresyon, AdaBoost, Stokastik Gradient Descent (SGD), Latent Drichlet Allocation (LDA) algoritmaları ile yaptıkları deneylerde, doğruluk metriğine göre 0.99 başarımı elde edilmektedir. Elde edilen en yüksek başarım doğrusal SVM kullanılarak elde edilmektedir.

(Z. Liu vd, 2021) Android kötü amaçlı yazılım tespitinde denetimsiz öğrenme tekniği ile öznitelik seçimi yapmıştır. İlk gün ataklarını tespit etmede başarılı olan denetimsiz öğrenme tekniklerinin ön işlem adımında öznitelik seçme yöntemi olarak kullanılabilir mi sorusunun cevabı çalışmada araştırılmaktadır. Bu durum çalışmanın ana motivasyonunu oluşturmaktadır. Sonuç olarak denetimsiz bir yaklaşım olan Restricted Boltzmann Machines (RBM) temelinde Subspace based Restricted Boltzmann Machines (SRBM) yöntemi önerilmiştir. Önerilen bu yöntemle öznitelik seçme işlemi gerçekleştirilmiştir. Çalışmada 3 farklı veri kümesi kullanılarak SRBM yönteminin performansı test edilmektedir. SRBM yönteminin başarımları RBM, Stacked Auto Encoder (SAE), PCA (Principal Components Analysis) ve Aglomeration algoritmaları ile kıyaslandığında daha başarılı sonuç vermiştir.

(Ananya vd, 2020) dinamik analiz tekniği ile Android kötücül yazılım tespiti yapmıştır. Çalışmada dinamik analiz ile çıkartılan sistem çağrıları unigram, bigram ve trigram gibi n-gram yapılarına dönüştürülerek kullanılmıştır. Deneyler, Drebin veri kümesinden alınan 2474 kötücül ve 9Apps'dan indirilen 2475 iyicil uygulama ile

gerçekleştirilmiştir. Çok sayıda sistem çağrısı meydana geldiği için SAILS adında öznitelik seçme yöntemi önerilmiştir. SAILS öznitelik seçme yönteminin alt yapısında, karşılıklı bilgi, ayırt edici özellik seçici ve Galavotti–Sebastiani–Simi gibi geleneksel öznitelik seçme yöntemlerinden elde edilen öznitelik skorları kullanılarak öznitelik seçme işlemi gerçekleştirilmiştir. Önerilen SAILS tekniğinin genel olarak karşılıklı bilgi, ayırt edici özellik seçici ve Galavotti–Sebastiani–Simi yöntemlerinden daha iyi olduğu vurgulanmıştır. LR, sınıflandırma ve regresyon ağaçları (CART), R F, XGBoost ve DNN teknikleri ile sınıflandırma yapılmıştır. Veri kümesinin %60'ı eğitim %40'ı ise test işlemine ayrılarak sınıflandırma sonuçları hesaplanmıştır. Unigram, bigram ve trigram modellerinin klasik makine öğrenmesine verilmesiyle yapılan deneylerde, unigramlı özellik vektörü için en yüksek doğruluğun RF algoritmasıyla, bigramlı özellik vektörü için en iyi sonucun XGBoost algoritmasıyla ve trigramlı özellik vektörü için en yüksek doğruluğun LR algoritmasından alındığı raporlanmıştır. Unigram modelinde, RF algoritması F-ölçütü metriğine göre %95.87 oranında başarım göstermiştir. Bigram modelinde ise XGBoost algoritması F-ölçütü metriğine göre %99.4 oranında başarım vermiştir. Trigram modelinde ise LR algoritması F-ölçütü metriğine göre %99.34 oranında başarım göstermiştir.

(Haq vd, 2021) derin öğrenme tekniklerini karma bir şekilde kullanarak Android kötücül yazılımların tespit edilmesini gerçeklemişlerdir. Oluşturulan karma yapı CNN ve LSTM mimarileri temel alınarak oluşturulmuştur. APK dosyasında yer alan kaynak kodlar, classes ve manifest dosyalarından bilgiler çıkartılarak özellik vektörü oluşturulmuştur. Elde edilen özellik vektörleri oluşturulan karma derin öğrenme ağına verilmiştir. Oluşturulan veri kümesi üzerinde öznitelik seçme ve normalizasyon gibi ön işlem adımları uygulanarak sınıflandırmanın etkili olması amaçlanmıştır.

(Alswaina ve Elleithy, 2018) 1233 Android kötücül yazılımı türlerine ayrıştırmaya çalışmaktadırlar. Toplamda 28 farklı türde Android kötücül yazılım türlerine göre sınıflandırılmaktadır. Öznitelik olarak uygulama izinleri makine öğrenmesi algoritmalarına girdi olarak verilmektedir. Bazı izinler çok tehlikeli grup altındayken bazı izinlerde nispeten daha az tehlikeli grup altındadır. Bu farklılıkları sayısallaştırmak ve sınıflandırma algoritmalarının performansını arttırabilmek için Extremely Randomized Tree adını verdikleri tekniği önermektedirler. Önerilen yöntem aynı zamanda öznitelik seçme görevini de yerine getirmektedir. Çalışmada 6 farklı sınıflandırma algoritması kullanılmaktadır. Bunlar destek vektör makinesi, ID3

karar ağacı, rastgele orman, yapay sinir ağları, en yakın komşu ve torbalama tekniğine dayalı algoritmalardır. En iyi sınıflandırma sonucu rastgele orman algoritması ile elde edilmektedir. Rastgele orman ile elde edilen sınıflandırma sonucu ise %95.97'dir.

(J. Li vd, 2018) makine öğrenmesi algoritmalarını kullanarak izin tabanlı Android kötücül yazılım tespit sistemi önermektedirler. Significant Permission IDentification (SIGPID) olarak adlandırılan yöntemle tüm izinleri kullanmak yerine iyicil yazılımlar ile kötücül yazılımların ayrıştırılmasını kolaylaştıracak izinlerin seçilmesi sağlanmaktadır. Önerilen yöntemle 135 izin 22 izne indirilmektedir. 22 izinle sınıflandırma yapıldığında, daha başarılı ve hızlı sonuç elde edilmektedir. Ayrıca çalışmada destek vektör makinesi ile %90'ın üzerinde sınıflandırma başarımının elde edildiği vurgulanmaktadır.

(Tao vd, 2018) yaptıkları çalışmada 31185 iyicil ve 15336 kötücül Android uygulamayı kullanmaktadırlar. MalPat adını verdikleri kötücül yazılım tespit sisteminde izinler ve API çağrıları öznitelik olarak kullanılmaktadır. Çalışmanın sınıflandırma aşamasında rastgele orman algoritması kullanılmaktadır. Deneysel sonuçlar incelendiğinde F-ölçütüne göre %98.24 oranında sınıflandırma başarımı elde edilmektedir.

(Pektaş vd, 2016) yaptıkları çalışmada 18 aileden oluşan 2000 kötücül uygulamayı ailelerine göre sınıflandırmayı amaçlamaktadırlar. Uygulamalar Cuckoo Sandbox vasıtasıyla işlenerek kötücül aileleri birbirlerinden ayıran en belirgin davranışsal özellikler çıkartılmaktadır. Elde edilen öznitelikler çevrimiçi makine öğrenmesi adı verilen bir sisteme verilerek kötücül yazılımların ailelerine göre sınıflandırılması gerçekleştirilmektedir. Yapılan deneylerde 7 sınıfta yer alan uygulamaların tamamı doğru olarak sınıflandırılmaktadır. Başarım oranının en düşük olduğu sınıf ise android.trojan.smskey ailesi olarak tespit edilmektedir.

(Dimjašević vd, 2016) dinamik analize dayalı kötücül yazılım tespit sistemi önermektedirler. Toplamda 12000'den fazla uygulama değerlendirilmektedir. Bu uygulamaların 4289 tanesi kötücül iken, 8371 tanesi ise iyicildir. Kötücül uygulamalar Drebin veri kümesinden elde edilirken, iyicil uygulamalar ise Google Play'den indirilmektedir. Sistem çağrıları dinamik olarak çıkartılarak makine öğrenmesi algoritmaları için öznitelik olarak kullanılmaktadır. Sistem çağrılarının oluşturulması kum havuzu ile gerçekleştirilmektedir. Uygulamaların işletim sistemi üzerinde neler

gerçekleştirdiği günlük dosyalarına kaydedilmektedir. Böylece her uygulamaya ait davranışlar kronolojik olarak oluşturulmaktadır. Sistem çağrılarına erişilmesi esnasında kötücül yazılımların bu çağrıları etkilemesine izin verilmemektedir. Bu sayede kötücül yazılımların davranışlarını değiştirme durumları da ortadan kalkmaktadır. Önerilen sistem bu özelliği sayesinde kötücül yazılımlarda sıkça görülen basit şaşırtma tekniklerine karşı dirençli olmaktadır. Elde edilen günlük dosyaları işlenerek özellik vektörleri oluşturulmaktadır. En son adımda ise bu özellik vektörleri makine öğrenmesi yaklaşımları ile değerlendirilerek iyicil ve kötücül yazılımların sınıflandırılması gerçekleştirilmektedir. Sınıflandırma aşamasında SVM, RF, LASSO ve ridge regularization gibi makine öğrenmesi tekniklerinden yararlanılmaktadır. En iyi performans RF algoritmasından elde edilmektedir.

(Arslan, 2021) derin sinir ağlarına dayalı Android kötücül yazılım tespit sistemi önermektedir. Statik analiz tekniğinden yararlanılarak çıkartılan uygulama izinleri öznitelik olarak kullanılmaktadır. Çalışmada derin sinir ağları çok sayıda geleneksel makine öğrenmesi yaklaşımları ile kıyaslanarak geniş kapsamlı deneyler yapılmaktadır. Yapılan deneylerde 7622 uygulama değerlendirilmektedir. Bu uygulamalardan 6661 tanesi kötücül uygulama iken 961 tanesi ise iyicil uygulamadır. Veri kümesinin %80'i eğitim %20'si ise test için bölünmektedir. En yüksek başarım derin sinir ağları ile elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.9820 olarak raporlanmaktadır. Derin sinir ağlarının geleneksel makine öğrenmesi yaklaşımlarından daha iyi sonuçlar verdiği gözlemlenmektedir.

(Milosevic vd, 2017) makine öğrenmesi yaklaşımlarından yararlanarak statik analize dayalı iki farklı yaklaşım sunmaktadırlar. İlk yaklaşımda statik analiz ile uygulama izinleri çıkartılmaktadır. İkinci yaklaşımda ise kelime çantası modeli ile kaynak kod analizi yapılmaktadır. İlk yaklaşımın hesaplama maliyetinin ikinci yaklaşıma göre oldukça düşük olduğu belirtilmektedir. Hem kümeleme hem de sınıflandırma algoritmalarından yararlanılarak çok sayıda deney gerçekleştirilmektedir. C4.5 karar ağacı, RF, Bayes ağları, SMO, Repeated Incremental Pruning (JRip), LR kullanılan algoritmalardan bazılarıdır. Ayrıca sınıflandırma algoritmaları birleştirilerek torbalama tekniklerine dayalı modeller de geliştirilmektedir. Makine öğrenimi algoritmaları 200 kötücül ve 200 iyicil Android uygulamadan oluşan M0DROID veri kümesi üzerinde çalıştırılmaktadır. İzin tabanlı yaklaşımda elde edilen en yüksek başarım SMO algoritması ile elde edilmektedir. Bu

başarım F-ölçütü metriğine göre 0.879'dur. Farklı torbalama teknikleri denenerek bu başarımın 0.894'e kadar çıkartılması sağlanmaktadır. Kaynak kod analizinde ise elde edilen en yüksek başarım SMO algoritması ile gerçekleştirilmektedir. Bu başarım F-ölçütü metriğine göre 0.951'dir. Farklı torbalama teknikleri denenerek bu başarımın 0.9560'a kadar çıkartılması sağlanmaktadır.

(Luo vd, 2017) dinamik analiz tekniği ile Android kötücül yazılımların tespit edilmesini sağlamaktadırlar. Dinamik analiz aşamasında sistem çağrıları ele alınarak uygulamaların davranışları analiz edilmektedir. Önerilen mimari ANDROIDETECT olarak adlandırılmaktadır. ANDROIDETECT, anlık saldırıların tespit edilmesini sağlayan makine öğrenimine dayalı bir Android kötü amaçlı uygulama algılama yöntemidir. Etkili özellik vektörlerinin oluşturulması sayesinde önerilen tespit yönteminin sınıflandırma sonucu düşük yanlış pozitif oranına sahiptir. Sistem çağrısı işlevi çıkartılarak özellik vektörleri oluşturulmaktadır. Ardından bu özellik vektörleri sınıflandırma algoritmaları ile değerlendirilmektedir. Çalışmada NB ve J48 karar ağaçları olmak üzere iki farklı sınıflandırma algoritması kullanılmaktadır. Deneyler 100 iyicil ve 100 kötücül uygulama ile gerçekleştirilmektedir. NB sınıflandırıcısından elde edilen sonuç F-ölçütü metriğine göre 0.825'dir. Buna karşın, J48 sınıflandırıcısından elde edilen sonuç ise F-ölçütü metriğine göre 0.86'dır.

(Dini vd, 2012) MADAM adını verdikleri çerçeve ile anomali tabanlı kötücül yazılım tespit sistemi geliştirmektedirler. Sistem her ne kadar Android işletim sistemi için tasarlanmış olsa da araştırmacılar önerilen mobil saldırı tespit sisteminin diğer işletim sistemlerine uygulanabileceğini belirtmektedir. Önerilen sistem kötücül yazılımların cihaza bulaşmasını engellemek için hem çekirdek düzeyinde hem de kullanıcı düzeyinde öznitelikler çıkartmaktadır. Çıkartılan bu öznitelikler KNN algoritması ile kullanılarak kötücül yazılımların tespit edilmesi sağlanmaktadır. MADAM'ın kötücül yazılım tespit başarım oranı %93 olurken yanlış pozitif oranı ise 0.0001 olarak raporlanmaktadır.

(Utku ve Dogru, 2017) statik analiz tekniğinden yararlanarak kötücül yazılımlar ile iyicil yazılımların sınıflandırmasını gerçekleştirmektedir. Çalışmada öznitelik olarak izinler tercih edilirken sınıflandırma algoritmaları olarak KNN ve NB algoritmaları kullanılmaktadır. Kullanılan veri kümesinde 5555 adet kötücül ve 1339 adet iyicil uygulama yer almaktadır. Veri setinin yaklaşık olarak %80'i eğitim %20'si ise test için kullanılmaktadır. KNN algoritmasında uzaklık metriği olarak Euclidean

mesafesi kullanılırken farklı k değerleri kullanılarak sınıflandırma yapılmaktadır. İyicil uygulamalar ile kötücül uygulamaların ayrıştırılmasında NB algoritması doğruluk metriğine göre 0.939 oranında başarım verirken KNN algoritması ise 0.8939 oranında başarım vermektedir. Eğitim setinde yer alan kötücül yazılımlar her iki algoritma ile %97 oranında kötücül olarak sınıflandırılmaktadır.

(Arslan ve Yurttakal, 2020) statik analize dayalı Android kötücül yazılım tespiti yapmaktadır. Önerilen kötücül yazılım tespit sisteminin sınıflandırma aşamasında KNN algoritması kullanılırken öznitelik olarak uygulama izinleri tercih edilmektedir. Kullanılan veri kümesinde toplamda 1189 uygulama yer almaktadır. Bu uygulamalardan 697 tanesi iyicil geri kalan 492 uygulama ise kötücüldür. KNN algoritmasında, k değerine 1'den 10'a kadar tüm değerler verilirken en iyi sınıflandırma sonucu 5 değerinde elde edilmektedir. Uzaklık metriği ise Minkowski seçilerek sınıflandırma yapılmaktadır. Veri kümesinin %70'i eğitim %30'u test için kullanıldığında doğruluk metriğine göre %91.2 başarım elde edilmektedir. Ek olarak, veri kümesinin %80'i eğitim %20'si test için kullanıldığında ise doğruluk metriğine göre %93.7 başarım elde edilmektedir.

(Cavli ve Sen, 2020) dinamik ve statik tekniklerinden yararlanarak hibrit ayırt edici özellikler ortaya çıkartmaktadır. Çıkartılan hibrit özellikler makine öğrenmesi tekniklerine verilerek Android kötücül yazılımların türlerine göre sınıflandırılması sağlanmaktadır. Uygulama izinleri gibi statik özellikler başta olmak üzere DNS ve HTTP protokollerinden çıkartılan ağ özellikleri ile aktivitelerin bigramları öznitelikleri oluşturmaktadır. Çalışmada toplamda statik ve dinamik özelliklerden oluşan 329 öznitelik yer almaktadır. Makine öğrenmesi tekniklerinden KNN, SVM, RF ve DT algoritmaları ile sınıflandırma yapılmaktadır. Genel olarak en iyi sınıflandırma algoritmasının KNN olduğu raporlanmaktadır. Malgenome, UpDroid ve Drebin olmak üzere 3 farklı veri kümesi ile deneyler gerçekleştirilmektedir. Bu veri kümelerinden KNN ile elde edilen başarımlar doğruluk metriğine göre sırasıyla %97.38, %98.04 ve %96.40'dır.

(Kedziora vd, 2019) tersine mühendislik tekniği ile Android uygulamaların Java kaynak kodlarından faydalanarak kötücül yazılım tespiti yapmaktadır. Kaynak kodlardan çıkartılan BroadcastReceiver ve Commands'lar öznitelikleri oluşturmaktadır. Toplamda 696 öznitelik kullanılmaktadır. Öznitelik vektörleri RF, NB, LR, KNN ve SVM sınıflandırma algoritmaları ile eğitilerek uygulamaların

sınıflandırılması gerçekleştirilmektedir. Çalışmada, toplam 1958 uygulama kullanılırken bunlardan 996 tanesi kötücül uygulamaları oluşturmaktadır. Elde edilen en yüksek başarım %80.6662 doğruluk ile RF algoritmasından elde edilmektedir. RF algoritmasının hemen ardından KNN algoritması gelmektedir. Bu sonuç doğruluk metriğine göre %80.3301'dir. KNN algoritmasında, uzaklık metriği Euclidean mesafesi olarak seçilirken k değeri ise 1 seçilerek sınıflandırma yapılmaktadır. Bu algoritmalara karşın, en kötü sonuç ise %76.1217 doğruluk ile NB algoritmasından elde edilmektedir.

(Darus vd, 2019) ilk olarak APK dosyalarını görüntülere dönüştürmektedir. APK dosyalarında yer alan class.dex dosyaları ve bu dosyalar içinde yer alan veri bölümü kısmı ayrı ayrı 8 bitlik gri seviyeli resimlere dönüştürülmektedir. Ardından GIST tekniği ile görüntüden öznitelik çıkarımı yapılmaktadır. class.dex ve veri bölümü kısmından iki farklı görüntü veri kümesi oluşturulmasının nedeni bu iki tekniği birbirleriyle kıyaslamak içindir. Elde edilen görüntülerden çeşitli özellikler çıkartılarak makine öğrenmesi teknikleri ile iyicil ve kötücül uygulamaların sınıflandırılması amaçlanmaktadır. Veri kümesinde 300 iyicil ve 418 kötücül uygulama yer almaktadır. 418 uygulama iki farklı kötücül yazılım ailesinden alınmaktadır. İki farklı kötücül yazılım ailesi ve iyicil yazılımlar sınıflandırılmaya çalışıldığı için üç sınıflı bir problem söz konusudur. XGBoost, KNN ve RF sınıflandırma algoritmaları ile sınıflandırma işlemi gerçekleştirilmektedir. Veri bölümü kullanılarak oluşturulan görüntülerin class.dex kullanılarak oluşturulan görüntülere göre daha iyi sınıflandırıldığı raporlanmaktadır. Veri bölümü kullanılarak oluşturulan görüntülerin sırasıyla XGBoost, KNN ve RF algoritmaları ile sınıflandırılmasından elde edilen sonuçlar doğruluk metriğine göre sırasıyla %70.37, %72.69 ve %74.07'dir. Aynı araştırmacıların yapmış oldukları başka bir çalışmada (Darus vd, 2018) ise görüntü işleme tekniklerinden yararlanılarak iyicil ve kötücül uygulamaların sınıflandırılması amaçlanmaktadır. 300 iyicil ve 183 kötücül uygulamadan oluşan veri kümesi kullanılarak bir önceki çalışmaya benzer biçimde uygulamalar görüntüye dönüştürülmektedir. Ardından görüntü özellikleri çıkartılarak makine öğrenmesi teknikleri ile sınıflandırma işlemi gerçekleştirilmektedir. KNN, RF ve DT algoritmaları ile sınıflandırma işlemi gerçekleştirilmektedir. Bu algoritmalardan elde edilen sonuçlar doğruluk metriğine göre sırasıyla %80.69, %84.14 ve %78.62'dir.

(Abawajy vd, 2021) IoT platformlarındaki Android işletim sistemli cihazlar için kötücül yazılım tespiti geliştirmiştir. Önerdikleri kötücül yazılım tespit sistemi makine öğrenmesine dayanmaktadır. Makine öğrenmesi algoritmalarına öznitelik olarak uygulama izinleri verilmiştir. Google Play Store'dan indirilen 6190 iyicil uygulama ve VirusShare, Drebin ve AndroZoo'dan indirilen 5500 Android kötücül uygulama çalışmada kullanılmıştır. Makine öğrenmesi algoritmalarından KNN, SVM, NB ve LR ile deneyler yapılmıştır. KNN algoritmasında k değeri 3 seçilmiştir. Bu algoritmaların verimliliğini arttırmak için CHI, karşılıklı bilgi, bilgi kazancı, Pearson korelasyon katsayısı ve varyans analizi (ANOVA) öznitelik seçme yöntemleri kullanılmıştır. CHI ve IG tekniklerinin diğerlerinden daha iyi olduğu raporlanmaktadır.

(Lu vd, 2021) 5G ağlarında yaygın bir şekilde Android akıllı cihazlar kullanıldığı için Android kötücül yazılım tespitine odaklanmaktadır. DLAMD adını verdikleri derin öğrenme tabanlı çerçeve ile kötücül yazılımların tespiti gerçekleştirilmektedir. APK dosyalarından çıkartılan izinler ve opcode özniteliklerdendir. DLAMD çerçevesi F-ölçütü metriğine göre %95.69 oranında başarım göstermektedir.

(Tahtaci ve Canbay, 2020) Android paketlerinin çözümlenmiş hali olan smali dosyalarının n-gram özelliklerini kullanarak özellik vektörünü oluşturmaktadır. 1-gram, 2-gram, 3-gram ve 4-gramdan oluşan farklı öznitelik grupları elde edilmektedir. Her bir durum için elde edilen öznitelik sayıları sırasıyla 214, 23004, 394011 ve 2583149'dur. Öznitelik sayısı oldukça fazla olduğundan varyans eşik değeri ve bilgi kazancı ile öznitelik seçimi yapılmıştır. Bazı öznitelikleri bir arada kullanabilmek ve biraz daha boyut indirgeme yapabilmek için temel bileşenler analizi ve stokastik komşuluk vektörü tekniklerine başvurulmaktadır. KNN, NB, LR, DT, SVM ve RF algoritmaları ile sınıflandırma yapılmaktadır. KNN algoritmasında, uzaklık metriği olarak Euclidean ve k değeri 3 seçilmiştir. 1500 iyicil ve 1500 kötücül uygulamadan oluşan veri kümesi kullanılarak deneyler gerçekleştirilmektedir. Çalışmada elde edilen en yüksek başarım SVM kullanılarak elde edilmektedir. Bu sonuç %99.99 olarak raporlanmaktadır. KNN algoritması ile elde edilen sonuç ise %99.83'tür.

(Zhang vd, 2018) yaptıkları çalışmada DeepClassifyDroid adını verdikleri derin öğrenme tabanlı çerçeve ile Android kötücül yazılımları tespit etmeye çalışmaktadır. Görüntü işleme ve doğal dil işlemede iyi sonuçlar veren derin öğrenme tekniği bu alana uygulanarak klasik makine öğrenmesi tekniklerine göre daha iyi sonuçlar elde edilmektedir. Çalışmada statik analizde kullanılan çok sayıda öznitelik derin öğrenme

ağına verilerek model performansı incelenmektedir. Çalışmada elde edilen en yüksek sınıflandırma başarımı ise F-ölçütü metriğine göre 0.974'tür. Ayrıca Lineer SVM, KNN ve NB algoritmaları CNN ağı ile karşılaştırılmaktadır. Bu algoritmaların başarımları sırasıyla 0.951, 0.971 ve 0.908'dir. Bunların yanında önerilen modelin Lineer SVM'den 10 kat, KNN algoritmasından ise 80 kat hızlı çalıştığı vurgulanmaktadır.

(Z. Wang vd, 2016) DroidDeepLearner adını verdikleri çerçeve ile Android işletim sistemi üzerinde çalışan kötücül yazılımların tespit edilmesini sağlamaktadırlar. Çalışmada derin öğrenme mimarisine dayanan derin inanç ağları yapısı kullanılmaktadır. Yanlış pozitif oranını minimize etmek için DBN algoritması tercih edilmektedir. Ayrıca DBN algoritması ile karşılaştırma yapmak için farklı çekirdeklerde destek vektör makinesi kullanılmaktadır. Çalışmada elde edilen en yüksek başarım F-ölçütü metriğine göre 0.9396'dır. Elde edilen en yüksek başarım 2 gizli katmana sahip DBN ile bulunmaktadır. Gizli katman sayısı 10 olduğunda, 0.9101 sınıflandırma başarımı gözlemlenmektedir. Her iki sonuç da destek vektör makinelerinin kullanılmasıyla elde edilen sonuçlardan daha yüksektir.

(S. Hou vd, 2016) Deep4maldroid adında bir çerçeve önermektedirler. Önerilen çerçeve dinamik analiz yaklaşımını temel almaktadır. Çalışmada dinamik analiz özelliklerinden olan Linux çekirdeği sistem çağrıları kullanılmaktadır. Linux çekirdeği sistem çağrıları elde edildikten sonra ağırlıklandırılmış yönlendirilmiş graflar meydana getirilmektedir. En son adımda elde edilen graflar derin öğrenmeye dayalı yığınlaştırılmış otomatik kodlayıcı ağlarına verilerek kötücül yazılımlar ile iyicil yazılımların ayrıştırılması sağlanmaktadır. Bu yaklaşım 4 farklı klasik makine öğrenmesi teknikleri ile karşılaştırılmaktadır. Bunlar SVM, ANN, NB ve DT algoritmalarıdır. Bu 4 algoritma arasından en iyi sonucu SVM vermektedir. SVM ile elde edilen sonuç doğruluk metriğine göre 0.8824'tür. Bu sonuca ek olarak derin öğrenme ile sınıflandırma yapıldığında başarım 0.9368'e çıkmaktadır.

(Zhenlong Yuan vd, 2014) Droid-sec adını verdikleri çerçeve ile iyicil ve kötücül yazılımları sınıflandırmaya çalışmaktadır. Android kötücül yazılım tespiti alanında derin öğrenmenin uygulandığı ilk çalışma olarak ön plana çıkmaktadır. Önerilen çalışmada dinamik ve statik özelliklerden oluşan 200'den fazla öznitelik kullanılmaktadır. Sınıflandırma aşamasında başta derin öğrenme olmak üzere NB, SVM, C4.5, LR ve MLP modelleri tercih edilmektedir. Derin öğrenme tekniğinin diğer

makine öğrenmesi tekniklerine göre daha başarılı olduğu gözlemlenmektedir. Derin öğrenme modeli doğruluk metriğine göre 0.965 sınıflandırma başarımı vermektedir. Bu başarıma en yakın sonuç SVM ile elde edilirken bu sonuç 0.8 olmaktadır. Aynı araştırmacılar bu çalışmaya benzer yaklaşımla Droiddetector çerçevesini önermektedir (Z. Yuan vd, 2016). Droiddetector çerçevesinin bir ön çalışması Droid-sec olarak düşünülebilir. Droiddetector'ün Droid-sec'den farkı daha fazla uygulama kullanılarak uygulama davranışları detaylandırılmaktadır.

(McLaughlin vd, 2017) uygulama dosyalarından çıkartılan baytcode'ları ele almaktadır. Ardından her uygulama için ele alınan baytcode'lar metin biçiminde kaydedilmektedir. Bu sayede metin sınıflandırma tekniklerinin kullanılması sağlanmaktadır. Önerilen yapı, n-gram tabanlı opcode dizgelerini kullanan Android kötücül yazılım tespit sistemlerinden ilham alınarak tasarlanmaktadır. Fakat n-gram yerine ham opcode dizgeleri tercih edilmektedir. CNN'in ham opcode dizgeleri üzerindeki farklılıkları ayrıştırmasının beklendiği ve performansının daha iyi olacağı öngörülmektedir. Böylece hem n-grama dönüşüm için ekstra hesaplama maliyeti olmayacak hem de modelin eğitimi sırasında milyonlarca n-gram sayma ve saklama ihtiyacı ortadan kalkacaktır. Ayrıca mevcut n-gram tabanlı yaklaşımlarda öznitelik arttığı için makine öğrenmesi algoritmaları üzerinde bazı problemler meydana gelmektedir. Önerilen yaklaşımda CNN kullanıldığı için bu problemler ortaya çıkmamaktadır. Çalışmadan elde edilen sonuçlar incelendiğinde, küçük ölçekli veri kümeleri üzerinde önerilen yöntem n-gram tabanlı yaklaşımlara göre oldukça başarılıdır. Büyük ölçekli veri kümelerinde benzer sınıflandırma başarımları elde edilmektedir. Fakat n-gram tekniği gibi ayrıca bir hesaplama yapılmadığı için daha hızlı çalışmaktadır.

(Shifu Hou vd, 2017) yapmış oldukları çalışmada statik analiz tekniğinde sıkça kullanılan API çağrılarını kullanmaktadırlar. API çağrıları iki farklı şekilde uygulanmaktadır. İlk olarak doğrudan API çağrıları ile özellik vektörü oluşturulmaktadır. Bu yaklaşım literatürde var olan yaklaşımdır. İkinci yaklaşımda ise doğrudan API çağrıları yerine Java ile Dalvik sanal makinesi arasında yer alan API çağrı bloklarının kullanılmasını fikri ortaya atılmaktadır. Oluşturulan her iki özellik vektörü derin öğrenme yaklaşımları ve klasik makine öğrenme yaklaşımları ile denenmektedir. Kullanılan derin öğrenme teknikleri DBN ve yığınlaştırılmış otomatik kodlayıcıdır. API çağrı blokları ile elde edilen sınıflandırma başarımının daha yüksek

olduğu gözlemlenmektedir. Ayrıca derin öğrenme tekniklerinin klasik makine öğrenmesi yaklaşımlarına göre daha başarılı sonuçlar verdiği görülmektedir. Derin öğrenme teknikleri birbirleriyle kıyaslandığında DBN'nin sınıflandırma başarımı daha yüksektir.

(Gharib ve Ghorbani, 2017) DNA-droid adını verdikleri çerçeve önermektedir. Bu çerçeve kötücül yazılımların bir türü olan Android fidye yazılımlarının (ramsomware) tespiti için özelleştirilmektedir. Önerilen sistem hibrit analiz yaklaşımıyla çalışmaktadır. Çalışmanın statik analiz kısmında üç farklı durum incelenmektedir. Bu durumlar metin sınıflandırma, görüntü sınıflandırma ve uygulama dosyalarından çıkartılan izinler ile API çağrılarıdır. Fidye yazılımları kullanıcılardan mesaj yoluyla para talep ettiği için metin sınıflandırma yaklaşımlarıyla bu bilgilerin çıkartılması fikri ortaya atılmaktadır. Ayrıca, fidye yazılımları bazı önemli markaların veya kurumların logolarını taklit etmektedir. Bu yüzden görüntü sınıflandırma adımına ihtiyaç duyulmaktadır. Ele alınan üç durum ile uygulamalar iyicil, kötücül veya şüpheli olmak üzere sınıflandırılmaktadır. Şüpheli olarak sınıflandırılan uygulamalar için ayrıca dinamik analiz adımı çalıştırılmaktadır. Sadece şüpheli uygulamalara dinamik analiz uygulanması çerçevenin dezavantajı olarak kabul edilebilir. Çünkü kötücül bir uygulama statik analiz aşamasında kamufle olabilir. Böylece, çerçeve bu uygulamalar için ayrıca denetim yapmayacaktır. Önerilen çerçeve CNN, NB, SVM, RF ve AdaBoost algoritmaları üzerinde denenmektedir. Bu algoritmalar içinde en başarılı sonuç CNN ile elde edilmektedir.

(Duc ve Giang, 2018) çok katmanlı yapay sinir ağlarını kullanarak Android kötücül yazılımların tespit edilmesini sağlamaktadır. 7 farklı statik özellik çıkartılarak 500000'den fazla öznitelik elde edilmektedir. Bu statik özellikler, istenen izin, uygulama bileşeni, amaç filtresi, özellik donanımı, API isteği, kullanılan izin ve URL'dir. Özellik vektörünün boyutu yapay sinir ağlarının çalışmasını zorlaştırmaktadır. Ayrıca özellik vektöründe çok sayıda 0 bulunmaktadır. Bu durumlardan dolayı özellik vektörü seyrek matrise dönüştürülerek sınıflandırma aşamasına geçilmektedir. Sınıflandırma aşamasında elde edilen en yüksek başarım F-ölçütü metriğine göre 0.923'tür.

(Masum ve Shahriar, 2019) iki veri kümesi üzerinde derin öğrenme ile sınıflandırma işlemini gerçekleştirmektedir. Bu veri kümeleri Malgenome-215 ve Drebin-215'dir. Her iki veri kümesinde 215 tane öznitelik bulunmaktadır. Öznitelik

olarak izinler, niyet filtreleri ve API çağrıları kullanılmaktadır. Modelin başarımını ölçmek için kullanılan metrik F-beta'dır. Bu metriğe göre Malgenome-215 üzerinde 0.992 sınıflandırma başarımı elde edilirken Drebin-215 veri kümesinde ise 0.988 başarım elde edilmektedir.

Alzaylaee vd. derin öğrenme tekniğini 7 farklı makine öğrenmesi ile kıyaslamaktadır (Alzaylaee vd, 2020). Çalışmada çok sayıda sistem tasarlanarak birbirleri ile karşılaştırılmaktadır. Göze çarpan karşılaştırmalardan birinde sadece dinamik özelliklerin olduğu bir model, karma bir model ile kıyaslanmaktadır. Karma model içerisinde dinamik özellikler ile statik özelliklerden olan uygulama izinleri yer almaktadır. Bunlar karşılaştırıldığında, dinamik özellikler yerine karma model kullanımının sınıflandırma başarımını dikkate değer bir şekilde arttırdığı gözlemlenmektedir. Dinamik analiz tekniğinde kötü amaçlı davranışları tetikleyecek kod kapsamını sağlamak için test girişi gerekmektedir. Test giriş teknikleri durum bilgisiz, durum bilgili ve iki modelin bir arada olduğu karma yapılardan oluşmaktadır. Çalışmada durum bilgisiz ve durum bilgili girdi teknikleri karşılaştırılmaktadır. Ayrıca dinamik özelliklerin çıkartılması gerçek cihaz üzerinde gerçekleştirilmektedir. Sonuçlar karşılaştırıldığında durum bilgili girdi tekniğinin daha iyi sınıflandırma başarımı verdiği vurgulanmaktadır. Çalışmada en iyi sonuç, durum bilgili girdi tekniğini kullanılarak oluşturulan dinamik özellikler ile uygulama izinlerinin birleşiminden elde edilmektedir. Bu modelin başarımı derin öğrenme ağında F-ölçütü metriğine göre 0.9882'dir. Derin öğrenme modelleri ile klasik makine öğrenmesi teknikleri kıyaslandığında derin öğrenmenin oldukça başarılı olduğu raporlanmaktadır. Klasik makine öğrenmesi tekniklerinden en iyisi ise RF algoritmasıdır.

(Su vd, 2016) DroidDeep adını verdikleri çerçevede statik özelliklerden izinler, API çağrıları ve bileşenler kullanılmaktadır. Bu özelliklerin işlenmesiyle 30 binden fazla öznitelik ortaya çıkmaktadır. Bu özniteliklerin birçoğu hem iyicil hem de kötücül uygulamalarda ortak görüldüğü için sınıflandırma başarımı olumsuz bir şekilde etkilenecektir. Aynı zamanda modelin hesaplama maliyeti artacaktır. Bu nedenlerden dolayı öznitelik seçme adımında DBN tercih edilmektedir. Sınıflandırma işlemi için destek vektör makinesi kullanılmaktadır. Elde edilen en yüksek başarım F-ölçütü metriğine göre 0.975'tir.

(Kim vd, 2019) çok modlu derin öğrenme tekniğini Android kötü amaçlı yazılım tespitine uyarlamaktadırlar. Çok modlu derin öğrenmenin Android kötü amaçlı yazılım tespitine uygulandığı ilk çalışma olarak gösterilmektedir. Çalışmada çok sayıda statik özellik kullanılarak özellik vektörü oluşturulmaktadır. Oluşturulan özellik vektörü gruplara ayrılarak farklı derin öğrenme modellerinin çalıştırılması sağlanmaktadır. Karar verme aşamasında ise modeller bir araya getirilerek uygulamanın sınıflandırılması amaçlanmaktadır. Elde edilen en iyi sonuç doğruluk metriğine göre %98 olurken, F-ölçütü metriğine göre 0.99 olmaktadır.

(Dharmalingam ve Palanisamy, 2021) öznitelik olarak izinleri kullanırken sınıflandırma aşamasında ise derin öğrenme tekniğinden faydalanmaktadır. İyicil uygulamalarda sık görülen izinler ile kötücül uygulamalarda sık görülen izinleri ayrıştırmak için puanlandırma sistemi önerilmektedir. Puanlama sistemi sonucunda ayrım gücü olmayan izinlerin elenmesi TF-IDF tekniği kullanılarak gerçekleştirilmektedir. Hem puanlandırma hem de özellik indirgeme ile elde edilen sınıflandırma başarımının daha iyi olduğu sonucuna varılmaktadır. Sınıflandırma aşamasında derin öğrenmenin yanında SVM ve DT algoritmaları da kullanılmaktadır. Ancak derin öğrenme tekniği bu algoritmalara göre oldukça iyi performans göstermektedir.

(Sharmeen vd, 2020) Android kötücül yazılımın tespiti için iki farklı model önermektedir. Önerilen ilk modelde denetimli yaklaşım kullanılırken ikinci modelde ise yarı-denetimli yaklaşım kullanılmaktadır. İlk modelde derin öğrenme tekniklerinden Restricted Boltzmann Machine (RBM) ile sınıflandırma işlemi gerçekleştirilmektedir. İkinci yaklaşımın denetimsiz öğrenme kısmında TF-IDF ile K-ortalama kümeleme algoritması birlikte kullanılarak etiketi belli olmayan uygulamalar iki gruba ayrıştırılmaktadır. Ardından çok sayıda sınıflandırma algoritması ile örneklerin ayrıştırılması hedeflenmektedir. Çalışmanın öznitelik seçimi aşamasında gömme öznitelik seçme yönteminden Boruta algoritması ile önemli özniteliklerin seçilmesi sağlanmaktadır. İlk önerilen modelde elde edilen en yüksek başarım doğruluk metriğine göre %99.024'tür. Yarı-denetimli yaklaşımının kullanıldığı ikinci modelde ise en yüksek başarım RF ve ExtraTree algoritmaları ile elde edilirken en düşük başarım SVM ile elde edilmektedir.

(W. Li vd, 2018) önerdikleri statik yaklaşımda uygulama izinlerini ve API fonksiyon çağrılarını kullanmaktadır. Uygulamalardan 148 tane izin ve 80 tane API

fonksiyon çağrısı çıkartılmaktadır. Bu özniteliklere ilaveten Kirin güvenlik kuralı (William Enck vd, 2009) ile 9 adet daha öznitelik oluşturularak özellik vektörüne eklenmektedir. Kirin yönteminin uygulanmasının sebebi kötücül uygulamalarda görülen izinler ile iyicil uygulamalarda görülen izinler arasında ayrımın yapılmasını sağlamak içindir. Sınıflandırma işlemi DBN kullanılarak gerçekleştirilmektedir. Çalışmada elde edilen en yüksek başarım duyarlılık metriğine göre %94.2857'dir.

(Liang vd, 2017) dinamik analiz yaklaşımıyla sistem çağrı dizgelerini metin olarak almaktadır. Ardından doğal dil işleme tekniği ve CNN kullanılarak modelin eğitilip iyicil uygulamalar ile kötücül uygulamaların sınıflandırılması yapılmaktadır. Önerilen kötücül yazılım tespit sisteminin en önemli avantajı dinamik analiz aşamasında elde edilen temel sistem çağrı dizgelerinin dışında herhangi bir öznitelik çıkarımı aşamasına ihtiyaç duymamasıdır. Elde edilen sistem çağrı dizgelerinin iyicil uygulamalara özgü veya kötücül uygulamalara özgü olup olmaması önemsenmemektedir. Çünkü bu aşamada doğal dil işleme tekniklerinden kelime gömme tekniği kullanılarak CNN'e girdi verilmektedir. Çalışmada elde edilen en yüksek başarım doğruluk metriğine göre %93.16'dır.

(Shiqi vd, 2018) Android kötücül yazılımların tespiti için ilk olarak API çağrılarını öznitelik olarak kullanmaktadır. Ardından kötücül uygulamaların ayrıştırılmasında etkili olan görüntü dokusu modeli ile API çağrıları birleştirilerek sınıflandırma başarımının arttırılmasını sağlamaktadır. Uygulama dosyaları derlendikten sonra ikili görüntü oluşturularak görüntü dokusu elde edilmektedir. Çalışmada DREBIN veri kümesinden 5560 kötücül uygulama kullanılmaktadır. Test çeşitliliğini artırmak için 1550, 2620, 5825 ve 6965 iyicil uygulama seçilerek 4 farklı veri kümesi oluşturulmaktadır. Çalışmada elde edilen en yüksek başarım API ve görüntü dokularının birlikte kullanılmasıyla elde edilmektedir. DBN kullanılarak elde edilen bu sonuç doğruluk metriğine göre %95.7'dir. Sadece API kullanıldığında elde edilen sonuç ise %93.5'tir.

(Nix ve Zhang, 2017) dinamik analiz yaklaşımıyla çıkartılan API çağrı dizgelerini öznitelik olarak kullanmaktadır. Çalışmada uygulamaların ayrıştırılması için iki farklı deney yapılmaktadır. İlk deneyde 11 farklı türde yer alan iyicil uygulamalar türlerine göre sınıflandırılırken ikinci deneyse ise 8 farklı türde olan kötücül uygulamalar türlerine göre sınıflandırılmaktadır. Çalışmanın sınıflandırma aşamasında derin öğrenme tekniklerinden CNN ve LSTM kullanılmaktadır. Ayrıca

API çağrı dizgelerinden oluşturulan n-gram modeli destek vektör makinesine girdi olarak verilirken kelime çantası tekniği ile oluşturulan model ise NB algoritmasına girdi olarak verilmektedir. Kötücül uygulamaların türlerine göre sınıflandırılmasında elde edilen en yüksek başarım doğruluk metriğine göre %99.4'tür. Bu sonuç CNN ile elde edilmektedir. LSTM, SVM ve NB algoritmalarından elde edilen sonuçlar ise sırasıyla %89.3, %66 ve %82'dir. İyicil uygulamaların türlerine göre sınıflandırılmasında elde edilen en yüksek başarım doğruluk metriğine göre %58.2'dir. Bu sonuç CNN ile elde edilmektedir. LSTM, SVM ve NB algoritmalarından elde edilen sonuçlar ise sırasıyla %27.8, %18.8 ve %13.6'dır.

(Ganesh vd, 2017) uygulamalardan çıkartılan izinleri 12×12'lik görüntüye dönüştürmektedir. Dönüştürülen görüntüler ise CNN'e verilmektedir. Birçok görüntü sınıflandırma problemlerinde kullanılan ve başarılı sonuçlar veren LeNet modeli ile kötücül yazılımların sınıflandırılması yapılmaktadır. LeNet modelinin yanında GoogleNet ve AlexNet modelleri de kullanılmaktadır fakat LeNet kadar iyi sonuç vermemektedir. Çalışmada elde edilen en yüksek başarım doğruluk metriğine göre %93'tür.

(Ding vd, 2018) APK dosyalarından çıkartılan API çağrı dizgelerini iki boyutlu görüntüye dönüştürmektedirler. Ardından elde edilen görüntüler CNN algoritması ile eğitilerek iyicil ve kötücül uygulamaların ayrıştırılması sağlanmaktadır. Ayrıca API çağrı dizgeleri 2-gram şekline dönüştürülerek SVM, KNN ve RF sınıflandırma algoritmaları ile iyicil ve kötücül yazılımların ayrıştırılması sağlanmaktadır. Çalışmada elde edilen en yüksek başarım CNN ile elde edilmektedir. Bu sonuç doğruluk metriğine göre %90'dır. SVM, KNN ve RF sınıflandırma algoritmalarından elde edilen sonuçlar doğruluk metriğine göre sırasıyla 0.88, 0.87 ve 0.89'dur.

(Alshahrani vd, 2018) DDefender adını verdikleri çerçeve ile dinamik ve statik özellikleri çıkartarak Android iyicil ve kötücül yazılımların sınıflandırılmasını amaçlamaktadırlar. Dinamik analiz özelliklerinden sistem çağrıları, Linux sistem bilgisi, ağ trafiği bilgisi ve çalışma zamanında istenilen izinler kullanılmaktadır. Statik analiz özelliklerinden aktivitelerin sayısı, intentler ve receiverlar kullanılmaktadır. Çalışmada iki farklı deney gerçekleştirilmektedir. İlk deneyde veri kümesinin %70'i eğitim %30'u test için bölünürken, ikinci deneyde ise veri kümesinin %80'i eğitim %20'si test için bölünmektedir. Elde edilen en yüksek başarım ikinci deneyde meydana gelmektedir. Bu sonuç doğruluk metriğine göre %95.13'tür.

(X. Xiao ve Yang, 2019) yapmış olduğu çalışmada Dalvik bayt kodlarını görüntülere dönüştüren CNN tabanlı bir yöntem önermektedir. Bu yaklaşımda DEX dosyaları onaltılık biçimde okunarak ardışık grupları sırasıyla RGB kanallarına dönüştürülmektedir. Eğitim ve test aşamalarında Google Play'den indirilen 4406 iyi huylu uygulama ve AMD veri kümesinden alınan 6134 kötü amaçlı yazılım örneği kullanılmaktadır. Veri kümesinin %80'i eğitim %20'si ise test için bölünmektedir. Çalışmada doğruluk metriğine göre %93 başarım elde edilmektedir.

(Lee vd, 2021) yaptıkları çalışmada 5000 iyicil ve 2500 kötücül uygulamadan çıkartılan 1104 statik özniteliği genetik algoritma tabanlı öznitelik seçme yöntemi ile değerlendirmektedirler. Genetik algoritmanın yanında bilgi kazancı metriği de öznitelik seçme işleminde karşılaştırma yapılabilmesi için kullanılmaktadır. 7500 uygulamanın 6000 tanesi eğitim için geriye kalan 1500 tanesi ise test için ayrılmaktadır. Yapılan çalışmada 9 farklı makine öğrenmesi modeli kullanılarak sınıflandırma başarımları hesaplanmaktadır. Öznitelik seçimi yapılmadan elde edilen en yüksek başarım SMO algoritması ile elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.988'dir. Genetik algoritma ile öznitelik seçimi yapıldığında SMO algoritmasından elde edilen başarım F-ölçütü metriğine göre 0.946'dır. Bilgi kazancı ile öznitelik seçimi yapıldığında ise SMO algoritmasından elde edilen başarım F-ölçütü metriğine göre 0.927'dir. Genel olarak genetik algoritmanın daha iyi öznitelik seçimi yaptığı vurgulanmaktadır.

(Bakour ve Ünver, 2021b) VisDroid olarak adlandırılan çerçeve ile kötü amaçlı yazılım örneklerini ailelerine göre sınıflandırmayı amaçlamaktadır. Çalışmada uygulama dosyaları gri seviyeli görüntülere dönüştürülerek elde edilen görüntülerden öznitelik çıkarımı gerçekleştirilmektedir. Ardından geleneksel makine öğrenmesi yaklaşımları ile kötücül uygulamalar ailelerine göre sınıflandırılmaktadır. Geleneksel makine öğrenmesi yaklaşımlarının yanında bazı derin öğrenme teknikleri kullanılarak çeşitli karşılaştırmalar yapılmaktadır. Aynı yazarlar tarafından önerilen bir başka yaklaşım ise DeepVisDroid çerçevesidir (Bakour ve Ünver, 2021a). Bu çalışmada ise derin öğrenme modelleri üzerinde iyileştirmeler yapılarak sınıflandırma başarımının arttırılması hedeflenmektedir.

(Utku, 2022) çalışmasında, ağ trafiği analizinden yararlanılarak Android kötücül yazılım tespit sistemi önerilmektedir. LSTM gibi derin öğrenme tekniklerinin yanında NB, RF ve SVM gibi geleneksel makine öğrenmesi algoritmaları da kötücül yazılım

tespit sisteminin alt yapısında denenmektedir. Çalışmada 3141 kötücül ve 4704 iyicil uygulamadan oluşan veri kümesi kullanılmaktadır. Bu veri kümesine ait 10 tane öznitelik yer almaktadır. Bu öznitelikler ağ trafiği ile ilişkilidir. Elde edilen sonuçlar incelendiğinde en yüksek başarım LSTM tekniği ile elde edilirken bu başarım doğruluk metriğine göre %95 olarak raporlanmıştır.

(Shanmugam vd, 2022) statik analiz tekniğinden yararlanarak uygulama izinleri, API çağrıları ve niyet filtrelerini öznitelik olarak kullanmaktadır. Tüm bu öznitelik gruplarından elde edilen öznitelik sayısı 250'dir. LSTM tekniği elektro arama optimizasyonu algoritması ile birlikte kullanılarak hibrit bir yaklaşım önerilmektedir. Bu hibrit model ile iyicil ve kötücül uygulamaların sınıflandırılması gerçekleştirilmektedir. Çalışmada 14253 iyicil ve 13821 kötücül uygulamadan oluşan bir veri kümesi kullanılmıştır. Veri kümesinin %80'i eğitim %20'si test olarak kullanıldığında doğruluk metriğine göre %97.69 oranında başarım elde edilmektedir.

(Urooj vd, 2022) tersine mühendislik yaklaşımı ile çıkarılan çok sayıda özniteliği makine öğrenmesi algoritmaları ile değerlendirerek uygulamaların iyicil veya kötücül olarak sınıflandırılmasını sağlamaktadırlar. SVM gibi geleneksel makine öğrenmesi algoritmasının yanında topluluk öğrenmesine dayalı AdaBoost algoritması da kullanılmaktadır. Farklı veri kümeleri üzerinde deneyler yapılarak önerilen yöntemin başarımları test edilmiştir. Yapılan deneylerden elde edilen sonuçlar incelendiğinde en yüksek başarımın doğruluk metriğine göre %96.24 olduğu gözlemlenmektedir.

3. FİLTRELEME TABANLI ÖZNİTELİK SEÇME YÖNTEMLERİ KULLANILARAK ÖNERİLEN İZİN TABANLI KÖTÜCÜL YAZILIM TESPİT SİSTEMİ

Bu bölümde var olan öznitelik seçme yöntemleri ile metin sınıflandırma alanında kullanılan bazı öznitelik seçme yöntemlerinin izin tabanlı kötücül yazılım tespitinde kullanımına yer verilecektir.

3.1. Bölüm Motivasyonu

Metin sınıflandırmada kullanılan birçok teknik Android kötücül yazılım tespiti alanında kullanılmaktadır (Çoban ve Özel, 2019; Dharmalingam ve Palanisamy, 2021; Kural vd, 2019; Peynirci vd, 2020; Salah vd, 2020; Suarez-Tangil vd, 2014; Şahın vd, 2018). Metin sınıflandırmada iyi sonuçlar veren ilişki frekansı ağırlıklandırma yöntemi izinler üzerine uygulanarak sınıflandırma başarımının arttırılması sağlanmaktadır (Şahın vd, 2018). Benzer bir yaklaşım (Kural vd, 2019) tarafından yapılmaktadır. Metin sınıflandırmada sıkça kullanılan terim ağırlıklandırma yöntemleri izinler üzerinde uygulanarak makine öğrenmesi algoritmalarının performansları karşılaştırılmaktadır (Kural vd, 2019). Uygulamaların kaynak kodları metin madenciliği ve bilgi getirme teknikleriyle analiz edilerek Android kötücül yazılımlar türlerine ayrıştırılmaktadır (Suarez-Tangil vd, 2014). Metin sınıflandırmada kullanılan IDF yaklaşımından esinlenerek Delta_IDF adı verilen metrik ile öznitelik seçme işlemi gerçekleştirilmektedir (Peynirci vd, 2020). Benzer şekilde (Salah vd, 2020) IDF tekniğinden yararlanarak geliştirdikleri öznitelik seçme yöntemini Android kötücül yazılım tespitinde kullanmaktadırlar. İyicil uygulamalarda sık görülen izinler ile kötücül uygulamalarda sık görülen izinleri ayrıştırmak için puanlandırma sistemi önerilmektedir (Dharmalingam ve Palanisamy, 2021). Puanlandırma sistemi sonucunda ayrım gücü olmayan izinlerin elenmesi TF-IDF tekniği kullanılarak gerçekleştirilmektedir. Metin madenciliğinde özellik vektörünün oluşturulmasını sağlayan kelime çantası ve n-gram teknikleri Android kötücül yazılım tespitine uygulanmaktadır (Çoban ve Özel, 2019). Bu çalışmalar incelendiğinde metin madenciliğinde uygulanan birçok tekniğin Android kötücül yazılım tespiti üzerinde uygulandığı ve dikkate değer başarımlar elde edildiği gözlemlenmektedir.

Android kötücül yazılım tespiti alanında makine öğrenmesi algoritmalarının verimliliğini arttırmak için etkili öznitelik seçme yöntemlerine ihtiyaç vardır (Pan vd,

2020; W. Wang vd, 2019). (W. Wang vd, 2019) tarafından yapılan derleme çalışmasında, ele alınan 234 makaleden 31 tanesinde özellik seçme yöntemi kullanılmaktadır. (Pan vd, 2020) tarafından yapılan derleme çalışmasında ise incelenen çalışmaların %47'sinde herhangi bir öznitelik seçme yönteminin uygulanmadığı geri kalan çalışmaların büyük kısmında ise var olan öznitelik seçme tekniklerinin kullanıldığı vurgulanmaktadır. Bazı çalışmalarda öznitelik sayısının bir milyona çıkabileceği vurgulandığından, öznitelik seçme yönteminin uygulanması araştırmacılar için kaçınılmaz hale gelmektedir (W. Wang vd, 2019). Android kötücül yazılım tespiti alanında öznitelik seçmenin önemli bir konu olması ve metin sınıflandırmada iyi sonuçlar veren bazı öznitelik seçme yöntemlerinin bu alanda kullanılmamış olması bizi bu araştırmayı yapmaya yöneltmektedir.

3.2. Bölüm Katkısı

Bu bölümün ana katkıları şöyle özetlenebilir:

- Filtreleme tabanlı öznitelik seçim yöntemleri kullanılarak yeni bir Android kötü amaçlı yazılım tespit sistemi önerilmiştir. Önerilen yaklaşım, makine öğrenimine dayalı statik Android kötü amaçlı yazılım tespit sistemidir.
- Android kötü amaçlı yazılım tespit sistemlerinde özellik seçimi önemli bir konu olduğundan, mevcut öznitelik seçim yöntemlerine alternatif metin sınıflandırma tabanlı özellik seçim yöntemleri Android kötü amaçlı yazılım tespit sistemine uyarlanmıştır. Böylece tüm izinleri kullanmak yerine en ayırt edici izinler seçilerek sınıflandırma algoritmalarının performansı iyileştirilir.
- Toplam sekiz farklı özellik seçim metriği kullanılmıştır. Her bir metrikten elde edilen izinler ve sınıflandırma sonuçları karşılaştırmalı olarak verilmiştir. Sonuçlar incelendiğinde uyarlanan öznitelik seçme yöntemlerinin mevcut öznitelik seçme yöntemlerine göre daha az öznitelikte daha başarılı olduğu gösterilmiştir.
- Uyarlanmış öznitelik seçim yöntemlerinden elde edilen sonuçlar, IG, CHI ve OR metriklerinden elde edilen sonuçlara göre daha az öznitelikle genellikle daha iyidir.
- Farklı öznitelik seçme yöntemlerinden elde edilen çeşitli boyutlardaki öznitelik vektörlerinin sınıflandırma performansları, çok sayıda makine öğrenmesi tekniği kullanılarak değerlendirilmektedir. Özellikle az özniteliğe

sahip özellik vektörünün MLP ile sınıflandırılması sonucunda hem sınıflandırma başarımı artmakta hem de çalışma süresi azalmaktadır.

- Bazı metrikler kendi aralarında açgözlü yaklaşım ile birleştirilerek çeşitli öznitelik alt kümeleri oluşturulmaktadır. Oluşturulan bu öznitelik alt kümeleri çok sayıda sınıflandırıcı ile test edildiğinde dikkate değer başarımlar elde edilmiştir.

3.3. Öznitelik Seçme Yöntemleri

Var olan özellik vektörünü temsil eden en iyi alt kümenin seçilmesi işlemine öznitelik seçimi denilmektedir. Öznitelik seçme teknikleri 3 farklı kategori altında incelenmektedir (Chandrashekar ve Sahin, 2014). Bunlar filtrelemeli, sarmalamalı ve gömülü yöntemlerdir. Filtreleme tabanlı tekniklerde, tüm öznitelikler arasından k tane en iyi öznitelik seçilir. Geri kalan öznitelikler sınıflandırma adımında kullanılmaz. k tane en iyi özniteliğin bulunması işleminde çeşitli istatistiksel veya bilgi teorisine dayalı teknikler kullanılmaktadır. Sarmalamalı teknikler de çalışma şekli bakımından filtrelemeli tekniklere benzemektedir fakat arama stratejisinde istatistiksel teknikler yerine genetik algoritmalar gibi sezgisel yöntemler ile seçim yapılmaktadır. Filtrelemeli yöntemler sarmalamalı yöntemler ile kıyaslandığında, hesaplama maliyeti bakımından filtrelemeli teknikler daha iyi olurken, en ilişkili alt kümenin bulunması bakımından sarmalamalı yöntemler genellikle daha iyidir. Filtrelemeli ve sarmalamalı teknikler ile öznitelik seçme işlemi ön işlem adımında gerçekleştirilmektedir. Bu iki tekniğin yanında gömülü yöntemler de vardır. Filtrelemeli ve sarmalamalı tekniklerinden farklı olarak öznitelik seçme işlemi makine öğrenmesi algoritmalarının eğitim aşamasında gerçekleştirilmektedir. Eğitim aşamasında oluşturulan modelin başarımını etkileyen en iyi alt kümenin bulunması ile öznitelik seçimi gerçekleştirilmektedir. Hesaplama maliyeti bakımından filtrelemeli yöntemler ve sarmalamalı yöntemlerin arasında kalmaktadır. Buna karşın diğer tekniklere göre daha ilişkili alt kümenin elde edilmesi sağlanmaktadır.

Tezin bu bölümünde gerçekleştirilen Android kötücül yazılım tespit sisteminin altyapısında filtreleme tabanlı 8 farklı öznitelik seçme yöntemi kullanılmaktadır. Bunlardan 4 tanesi çeşitli Android kötücül yazılım tespit çalışmalarında zaten kullanılmaktadır. Bu yöntemlerin matematiksel altyapıları Bölüm 3.3.1'de verilecektir. Geri kalan 4 öznitelik seçme yöntemi ise metin sınıflandırma alanından Android kötücül yazılım tespit sistemine uyarlanan öznitelik seçme yöntemleridir. Bu

yöntemlerin matematiksel altyapıları ise Bölüm 3.3.2'de verilecektir. Bölüm 3.3.1 ve 3.3.2'de değinilecek metriklerde kullanılan notasyonlar Tablo 3.1'de yer almaktadır.

Tablo 3.1. Öznitelik seçme yöntemlerinde kullanılan notasyonlar

Gösterimler	Tanım
p	Uygulama izni
C	Toplam kategori sayısı
$P(c_i)$	Bir uygulamanın c_i sınıfına ait olma olasılığı
$P(p)$	Veri kümesindeki bir uygulamada p izninin kullanılma olasılığı
$P(\bar{p})$	Veri kümesindeki bir uygulamada p izninin kullanılmama olasılığı
$P(c_i\|p)$	p izninin c_i sınıfındaki uygulamalardan birinde kullanılma olasılığı
$P(c_i\|\bar{p})$	p izninin c_i sınıfındaki uygulamalardan hiç birinde kullanılmama olasılığı
a	c_i sınıfında p iznini kullanan uygulamaların sayısı
b	c_i sınıfında p iznini kullanmayan uygulamaların sayısı
c	p iznini kullanan ve c_i sınıfına ait olmayan uygulamaların sayısı
d	p iznini kullanmayan ve c_i sınıfına ait olmayan uygulamaların sayısı
N	Toplam uygulama sayısı yani $a + b + c + d$
n_1	c_i sınıfına ait uygulamaların sayısı
n_2	c_i sınıfına ait olmayan uygulamaların sayısı
d_1	p izninin kullanıldığı uygulama sayısı
d_2	p izninin kullanılmadığı uygulama sayısı

3.3.1. Var Olan Öznitelik Seçme Yöntemleri

Bu bölümde makine öğrenmesi alanında sıkça kullanılan filtreleme tabanlı öznitelik seçme yöntemlerine değinilecektir.

3.3.1.1. Bilgi Kazancı

Bilgi kazancı, bir özniteliğin ilgili sınıf için ne kadar bilgi içerdiğini ölçmeye yarayan metriktir. Bilgi kazancı metriğinin temelinde bilgi teorisine dayanan entropi kullanılmaktadır. Eşitlik 3.1'de IG metriğinin matematiksel gösterimi verilmektedir. Verilen ifadede c_i sınıfındaki p iznine ait IG skoru hesaplanmaktadır. IG değeri ne kadar büyük ise ilgili sınıf ile ilgili öznitelik arasındaki ilişki de o kadar yüksektir. Bu nedenle IG değerleri büyükten küçüğe doğru sıralanır. Kaç tane özniteliğe ihtiyaç varsa en büyük değere sahip o kadar öznitelik seçilerek en ayırt edici özniteliklerin bulanması gerçekleştirilmektedir.

$$IG(p, c_i) = -\sum_{i=1}^{C} P(c_i) log P(c_i) + P(p) \sum_{i=1}^{C} P(c_i|p) \\ log P(c_i|p) + P(\overline{p}) \sum_{i=1}^{C} P(c_i|\overline{p}) log P(c_i|\overline{p}) \quad \textbf{(3.1)}$$

3.3.1.2. Ki-kare Testi

İstatistikte, iki değişken arasında ilişki ve bağımlılık olup olmadığının tespit edilmesinde Ki-kare testi kullanılmaktadır. Öznitelik seçiminde ise öznitelik ile sınıf arasında nasıl bir ilişki olduğunu hesaplamak için bu metriğe başvurulmaktadır. Eşitlik 3.2'de CHI metriğinin matematiksel gösterimi verilmektedir. Verilen ifadede c_i sınıfındaki p iznine ait CHI skoru hesaplanmaktadır. CHI değeri ne kadar büyük ise ilgili sınıf ile ilgili öznitelik arasındaki ilişki de o kadar yüksektir.

$$CHI(p, c_i) = N \frac{(ad - bc)^2}{(a + c)(b + d)(a + b)(c + d)} \quad \textbf{(3.2)}$$

3.3.1.3. Göreceli Olasılıklar Oranı

Göreceli olasılık bir durumun olma olasılığı ile olmama olasılığı arasındaki ilişkiyi göstermektedir. Göreceli olasılıklar oranı ise iki olaya ait göreceli olasılık değerlerinin oranlanmasıyla elde edilmektedir. Eşitlik 3.3'te OR metriğinin matematiksel gösterimi verilmektedir. Verilen ifadede c_i sınıfındaki p iznine ait OR skoru hesaplanmaktadır.

$$OR(p, c_i) = \frac{a \times d}{b \times c} \quad \textbf{(3.3)}$$

3.3.1.4. Ters Doküman Frekansı

Bu metriğin asıl kullanıldığı yer bilgi geri getirme (information retrieval) alanıdır. Fakat çoğu makine öğrenmesi çalışmalarına uyarlanabilmektedir. Bir dokümanda geçen kelimenin doküman için ne kadar önemli olduğunu hesaplamada kullanılır. Eşitlik 3.4'te IDF metriğinin matematiksel gösterimi verilmektedir. Verilen ifadede c_i sınıfındaki p iznine ait IDF skoru hesaplanmaktadır.

$$IDF(p, c_i) = log(\frac{N}{a + c}) \tag{3.4}$$

3.3.2. Uyarlanan Öznitelik Seçme Yöntemleri

Bu bölümde metin sınıflandırma alanında kullanılan ve Andoid kötücül yazılım tespitine uyarlanan filtreleme tabanlı öznitelik seçme yöntemlerine değinilecektir. Metin sınıflandırmada geçen doküman ve terim burada sırasıyla uygulama ve izne dönüştürülmektedir.

3.3.2.1. Doküman Frekansı Eşikleme

Metin sınıflandırmada dokuman frekansı, terimin görüldüğü dokümanların sayısı şeklinde hesaplanmaktadır. Android kötücül yazılım tespitinde ise izni isteyen toplam uygulama sayısıdır. Eşitlik 3.5'te DF metriğinin matematiksel gösterimi verilmektedir. Verilen ifadede c_i sınıfındaki p iznine ait DF skoru hesaplanmaktadır.

$$DF(p, c_i) = a + c \tag{3.5}$$

3.3.2.2. Acc ve Acc2 Metrikleri

(George, 2003) tarafından raporlanmış metin sınıflandırma tabanlı öznitelik seçme metrikleridir. DF metriğinin aksine, Acc metriğinde toplam doküman sayısına değil de sınıflar arasındaki farka bakılmaktadır. Eşitlik 3.6'da Acc metriğinin matematiksel gösterimi verilmektedir.

$$Acc(p, c_i) = a - c \tag{3.6}$$

Eşitlik 3.6'da verilen farkı dengelemek için Eşitlik 3.6, Eşitlik 3.7'ye dönüştürülmektedir. Verilen ifadede c_i sınıfındaki p iznine ait Acc2 skoru hesaplanmaktadır.

$$Acc2(p, c_i) = \frac{a}{n_1} - \frac{c}{n_2} \quad \textbf{(3.7)}$$

3.3.2.3. M2 Metodu

Bu metrik Acc2 metriğinin farklı bir varyasyonu olarak önerilmektedir (Taşcı ve Güngör, 2013). Bu öznitelik seçme yönteminde iki çarpan kullanılmaktadır. Bunun nedeni ikinci çarpandan elde edilen değerler bazı terimler için aynı olabilmektedir. Bu yüzden DF metriği ile çarpım yapılarak aynı değere sahip metriklerin farklı değerler üretmesi sağlanmaktadır. Bu sayede ayrım gücü en iyi terimlerin öznitelik olarak seçilmesi hedeflenmektedir. Eşitlik 3.8'de M2 metriğinin matematiksel gösterimi verilmektedir. Verilen ifadede c_i sınıfındaki p iznine ait M2 skoru hesaplanmaktadır.

$$\mathrm{M2}(p, c_i) = DF(p, c_i)\left[\frac{a}{d_1} - \frac{b}{d_2}\right] \quad \textbf{(3.8)}$$

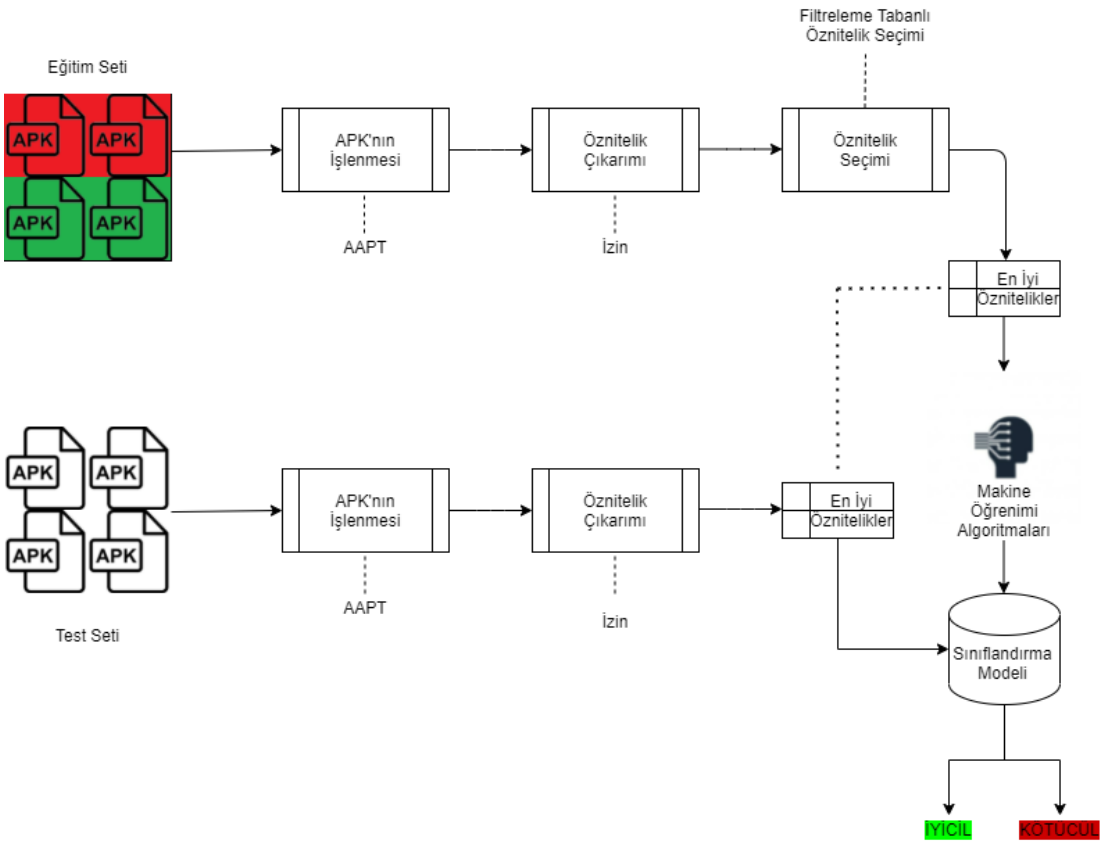
3.3.2.4. İlişki Sıklığı Öznitelik Seçimi

Terim ağırlıklandırma olarak önerilen ve iyi sonuçlar veren ilişki frekansı (Lan vd, 2009), (Durmuş Özkan Şahin ve Kılıç, 2019) tarafından DF metriği ile birleştirilerek öznitelik seçme metriği olarak kullanılmaktadır. Eşitlik 3.9'da RFFS metriğinin matematiksel gösterimi verilmektedir. Verilen ifadede c_i sınıfındaki p iznine ait RFFS skoru hesaplanmaktadır. c değerinin 0 olması durumunda bölme işlemi gerçekleşemez. Bu nedenle, c değerinin 0 olması durumunda $c = 1$ yapılarak öznitelik seçme işlemi gerçekleştirilmektedir.

$$RFFS(p, c_i) = DF(p, c_i)\left[log(2 + \frac{a}{c})\right] \quad \textbf{(3.9)}$$

3.4. Önerilen Kötücül Yazılım Tespit Sisteminin Altyapısı

Bu bölümde önerilen izin tabanlı Android kötücül yazılım tespit sistemi ele alınacaktır. Önerilen tespit sisteminin genel mimarisi Şekil 3.1'de verilmektedir.



Şekil 3.1. Önerilen tespit sisteminin genel mimarisi

Sistemin çalışma şekli şöyledir:

Adım 1: Öncelikle kullanılan veri kümesi eğitim ve test olmak üzere iki gruba ayrılmaktadır.

Adım 2: Eğitim kümesindeki uygulamalara Tablo 3.2'de detaylandırılan algoritma adımları uygulanarak izin listesi oluşturulmaktadır. Böylece özellik çıkarım adımları sayesinde uygulamalarda yer alan izinler elde edilmektedir.

Adım 3: Kullanılan izinler elde edildikten sonra Bölüm 3.3'te anlatılan filtreleme tabanlı öznitelik seçme teknikleri kullanılarak kötücül yazılımları iyicil yazılımlardan ayıran en belirgin özniteliklerin bulunması sağlanmaktadır.

Adım 4: En iyi özniteliklerin seçilmesiyle oluşan özellik vektörü eğitilmek üzere çeşitli makine öğrenmesi algoritmalarına verilerek sınıflandırma modeli oluşturulmaktadır. Böylece eğitim aşaması tamamlanmış olacaktır.

Adım 5: Adım 2'de olduğu gibi test edilecek uygulamalara özellik çıkarım aşaması uygulanır.

Adım 6: Adım 3'te elde edilen ayrım gücü en yüksek izinler test edilecek uygulamalar içerisinde kullanılıyorsa 1, kullanılmıyorsa 0, değeri atanarak test vektörü oluşturulur.

Adım 7: En son adımda ise oluşturulan test vektörü sınıflandırma modeline verilerek türü bilinmeyen uygulamaların türleri belirlenmektedir.

Bu 7 adım ile türü bilinmeyen uygulamalar türlerine ayrıştırılmaktadır. Önerilen sistem göz önünde bulundurulduğunda, tasarımı kolay ve mobil cihaz üzerinde doğrudan çalıştırılabilir biçimdedir.

3.4.1. Deneysel Ayarlamalar

Bu bölümde gerçekleştirilen deneylerde 6000 tane uygulamadan oluşan veri kümesi kullanılmaktadır. Bu uygulamalardan 3000 tanesi kötücül olurken geriye kalan 3000 uygulama ise iyicildir. İyicil uygulamalar birçok Android kullanıcısının uygulama edinmek için kullandığı APKPure'dan indirilmektedir (APKPure, 2021). Kötücül uygulamalar ise VirusShare veri kümesi içerisinden rastgele seçilmektedir (VirusShare, 2021).

APK dosyaları AAPT2 aracı (AAPT2, 2021) ile açılarak AndroidManifest.xml dosyası elde edilmektedir. Her uygulamanın AndroidManifest.xml dosyası tek tek işlenerek uygulama izinlerine erişilmektedir. Elde edilen izinler bir araya gelerek özellik vektörünü oluşturmaktadır. Tüm uygulama dosyaları tarandığında 470 tane eşsiz izin elde edilmektedir. Bu izinler Android işletim sisteminde yer alan doğal (native) ve özel (custom) izinlerden oluşmaktadır. Bu izinler uygulamalar tarafından isteniyorsa 1, istenmiyorsa 0 şeklinde doldurularak veri kümesi yapısal bir hale getirilmektedir. Bu sayede makine öğrenmesi algoritmalarının çalışabileceği yapı oluşturulmuş olacaktır. Tablo 3.2'de detayı verilen algoritma adımları ile uygulamalar işlenmekte ve toplu izin listesi oluşturulmaktadır. Elde edilen izinler ilgili uygulamada yer alıyorsa 1, yer almıyorsa 0 şeklinde ağırlıklandırma yapılarak özellik vektörü oluşturulmaktadır.

Özellik vektörleri oluşturulduktan sonra sınıflandırma algoritmaları bu vektörler üzerinde çıkarımlar yapabilir. Yedi farklı sınıflandırma algoritması kullanılarak öznitelik seçme yöntemleri karşılaştırılmaktadır. Bunlar KNN, NB, SMO, MLP, RF, C4.5 ve LR algoritmalarıdır. Tüm algoritmalar için WEKA paketinden yararlanılmaktadır (Hall vd, 2009). KNN'de k değeri 3 seçilererek sınıflandırma işlemi

gerçekleştirilmektedir. Diğer algoritmalarda WEKA paketinde varsayılan parametreler kullanılmaktadır. Son olarak F-ölçütü metriği ile sınıflandırma başarımları hesap edilerek her bir yöntemin başarımları değerlendirilmektedir.

Tablo 3.2. İzin listesini oluşturan algoritma

Algoritma: İzin listesinin elde edilmesi

Girdi: $Uygulama_1, Uygulama_2, \dots, Uygulama_N$

Çıktı: $izin\_listesi$

Function İ$ZİN_LİSTESİ_OLUŞTUR(Uygulama\ [\,])$

$izin\_listesi\ = \emptyset$ → Boş bir çırpı veri yapısı oluşturulmaktadır

$N_1 = length(Uygulama)$

For $i = 1\ to\ N_1$ **do**

AAPT2'yi çağır

$Uygulama[i]$'yi AAPT2 ile işle

$Uygulama[i]$'nin AndroidManifest.xml dosyasını elde et

AndroidManifest.xml dosyasından izinleri al ve kaydet

$N_2 = length(izinler)$

For $j = 1\ to\ N_2$ **do**

If $izinler[j]$ $izin\_listesi$ *çırpısının bir anahtarı değil* ise **then**

$izin\_listesi \leftarrow izinler[j]$ → İlgili izin çırpıda yoksa, çırpıya anahtar olarak eklenecektir.

End If

End For

End For

End Function

3.5. Elde Edilen Sonuçlar

Bu bölümde deneylerden elde edilen sonuçlar verilecektir. Elde edilen sonuçlar iki farklı şekilde yorumlanacaktır. İlk olarak metriklerden elde edilen en ayırt edici 10 izin karşılaştırmalı olarak değerlendirilecektir. İkinci olarak metriklerden elde edilen izinler çeşitli sınıflandırma algoritmalarına verildiğinde elde edilen sınıflandırma

başarımları yorumlanacaktır. Son olarak bazı metriklerin açgözlü yaklaşımla birleştirilmelerinden oluşan yeni öznitelik alt grupları ile elde edilen sınıflandırma sonuçları değerlendirilecektir. Tablo 3.3'te metriklerden elde edilen en iyi 10 izin verilmektedir.

Tablo 3.3. Metrikler tarafından oluşturulan en iyi 10 izin

Öznitelik Seçme Yöntemleri / **İzinler**	**CHI**	**IG**	**OR**	**IDF**	**DF**	**Acc2**	**M2**	**RFFS**
android.permission.MOUNT_UNMOUNT_FILESYSTEMS	✓	✓	×	×	×	✓	✓	×
android.permission.GET_TASKS	✓	✓	×	×	×	✓	✓	×
android.permission.READ_PHONE_STATE	✓	×	×	✓	✓	✓	✓	✓
android.permission.CHANGE_WIFI_STATE	✓	×	×	×	×	✓	✓	×
android.permission.CHANGE_NETWORK_STATE	✓	✓	×	×	×	✓	✓	×
android.permission.READ_SMS	✓	✓	×	×	×	✓	×	×
android.permission.SEND_SMS	✓	✓	✓	×	×	✓	×	×
android.permission.RECEIVE_SMS	✓	✓	×	×	×	✓	×	×
android.permission.WRITE_SETTINGS	✓	×	×	×	×	✓	✓	×
android.permission.WRITE_SMS	✓	✓	✓	×	×	×	×	×
android.permission.UPDATE_APP_OPS_STATS	×	✓	✓	×	×	×	×	×
android.permission.READ_SETTINGS	×	✓	✓	×	×	×	×	×
android.permission.RECEIVE_MMS	×	✓	✓	×	×	×	×	×
android.permission.ACCESS_MTK_MMHW	×	×	✓	×	×	×	×	×
android.permission.SAMSUNG_TUNTAP	×	×	✓	×	×	×	×	×
android.permission.ACCESS_CACHE_FILESYSTEM	×	×	✓	×	×	×	×	×
android.permission.WRITE_APN_SETTINGS	×	×	✓	×	×	×	×	×
android.permission.READ_INTERNAL_STORAGE	×	×	✓	×	×	×	×	×
android.permission.INTERNET	×	×	×	✓	✓	×	×	✓
android.permission.ACCESS_NETWORK_STATE	×	×	×	✓	✓	×	×	✓

Tablo 3.3. Metrikler tarafından oluşturulan en iyi 10 izin (devamı)

android.permission.WRITE_EXTERNAL_STORAGE	×	×	×	✓	✓	×	✓	✓
android.permission.WAKE_LOCK	×	×	×	✓	✓	×	×	✓
android.permission.ACCESS_WIFI_STATE	×	×	×	✓	✓	×	✓	✓
android.permission.READ_EXTERNAL_STORAGE	×	×	×	✓	✓	×	✓	✓
android.permission.VIBRATE	×	×	×	✓	✓	×	×	✓
android.permission.RECEIVE_BOOT_COMPLETED	×	×	×	✓	✓	×	×	✓
android.permission.ACCESS_COARSE_LOCATION	×	×	×	✓	✓	×	✓	×
android.permission.SYSTEM_ALERT_WINDOW	×	×	×	×	×	✓	×	×
com.android.vending.BILLING	×	×	×	×	×	×	×	✓

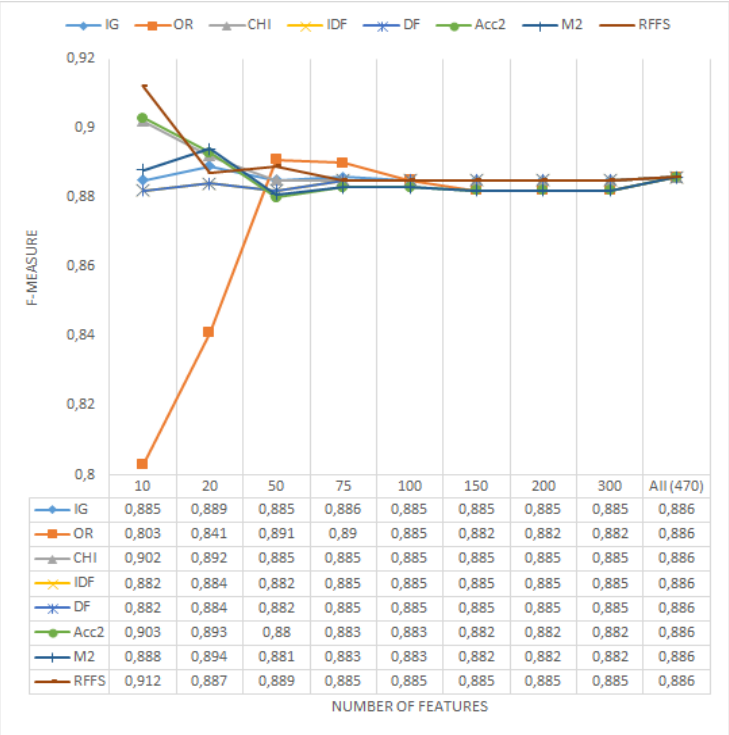
Tablo 3.3'e göre 29 farklı izin metrikler tarafından elde edilmektedir. Bu izinlerden android.permission.READ_PHONE_STATE en çok ortak bulunan izindir. 8 metriğin 6 tanesi tarafından bu izin öznitelik olarak seçilmekteyken sadece IG ve OR metrikleri bu izni en tehlikeli 10 izin arasında bulamamaktadır. Uyarlanan metriklerin tamamı bu izni seçmektedir. Bu izin Kaspersky tarafından tehlikeli Android izinler arasında gösterilmektedir (Kaspersky, 2021a). Bu izni kullanan uygulamalar sesli iletişimle ilişkili hemen hemen her eylemi gerçekleştirme imkânına sahip olmaktadır. Bu izin sayesinde, kullanıcının hangi numarayı ne zaman aradığı bilinecektir. Hatta ücretli numaralar da dâhil olmak üzere kullanıcının haberi olmadan kullanıcının ücreti karşılığında aranabilecektir. Amacımız kötücül uygulamaları iyicil uygulamalardan ayıran en ayırt edici izinleri bulmak olduğu için uyarlanan metriklerin böylesine tehlikeli izni bulmaları dikkate değerdir.

Tehlikeli izin gruplarından bir tanesi de SMS işlemleridir. Bu gruptaki izinlerden olan READ_SMS ve RECEIVE_SMS CHI, IG ve Acc2 metriği tarafından öznitelik olarak seçilirken, SEND_SMS ve WRITE_SMS izinleri CHI, IG ve OR metrikleri tarafından seçilmektedir. RECEIVE_MMS ise IG ve OR tarafından seçilmektedir. Bu tehlikeli izin grupları dışında depolama ve konum işlemlerinin olduğu izinler de vardır. Depolama işlemleri içerisinde yer alan READ_EXTERNAL_STORAGE ve WRITE_EXTERNAL_STORAGE IDF, DF, M2 ve RFFS ile elde edilmektedir.

Konum grubunda yer alan ACCESS_COARSE_LOCATION izni ise IDF, DF ve M2 metriği tarafından seçilmektedir.

CHI metriği tarafından elde edilen izin listesi IG, Acc2 ve M2 ile elde edilen listeye benzer özellikler göstermektedir. CHI ile elde edilen 10 izinden 7 tanesi IG, 9 tanesi Acc2 ve 6 tanesi de M2 metriği listesinde de mevcuttur. IDF ve DF metriği tamamen aynı izinleri seçerken bunlara en yakın izin listesi RFFS tarafından elde edilmektedir. IDF ve DF metriği ile bulunan 10 iznin 9 tanesi RFFS tarafından da bulunmaktadır. En farklı öznitelikler ise OR metriği tarafından elde edilmektedir. OR metriğinin bulduğu sadece 5 izin ortaktır. Bunlardan 2 tanesi hem CHI hem de IG metriğinde yer alırken geri kalan 3 izin ise sadece IG metriğinde yer almaktadır.

Yapılan deneylerden elde edilen sınıflandırma sonuçları Şekil 3.2'den itibaren verilmektedir. Şekil 3.2'de NB algoritmasından elde edilen sonuçlar yer almaktadır. Şekil 3.2'ye göre NB algoritmasından elde edilen en yüksek sınıflandırma başarımı RFFS ile 10 öznitelik seçildiğinde 0.912 olarak bulunmaktadır. Öznitelik sayısı 10 ve 20 olduğunda OR dışındaki metrikler benzer sonuçlar döndürmektedir. Ancak öznitelik sayısı 50 ve daha fazla olduğunda OR metriğinden elde edilen sonuçlar diğer metriklerden elde edilen sonuçlara benzer sonuçlar vermektedir. Bunun nedeni OR metriği 470 tane özelliği temsil eden en iyi alt kümeyi 50 öznitelikte bulmaktadır. Buna karşın diğer metrikler 10 veya 20 öznitelikte NB algoritması ile elde edilebilen en yüksek başarımları bulabilmektedir. Örneğin, CHI metriği ile 10 öznitelikte 0.902 başarım elde edilirken, 20 öznitelikte 0.892 başarım elde edilmektedir. Bu iki sonuç NB algoritmasından CHI ile elde edilmiş en yüksek başarımlardır. CHI metriğindeki benzer durum Acc2 metriğinde de görülmektedir. Acc2 metriği ile 10 öznitelikte 0.903 başarım elde edilirken, 20 öznitelikte 0.893 başarım elde edilmektedir. Bu iki sonuç Acc2 için elde edilen en yüksek başarımlardır.

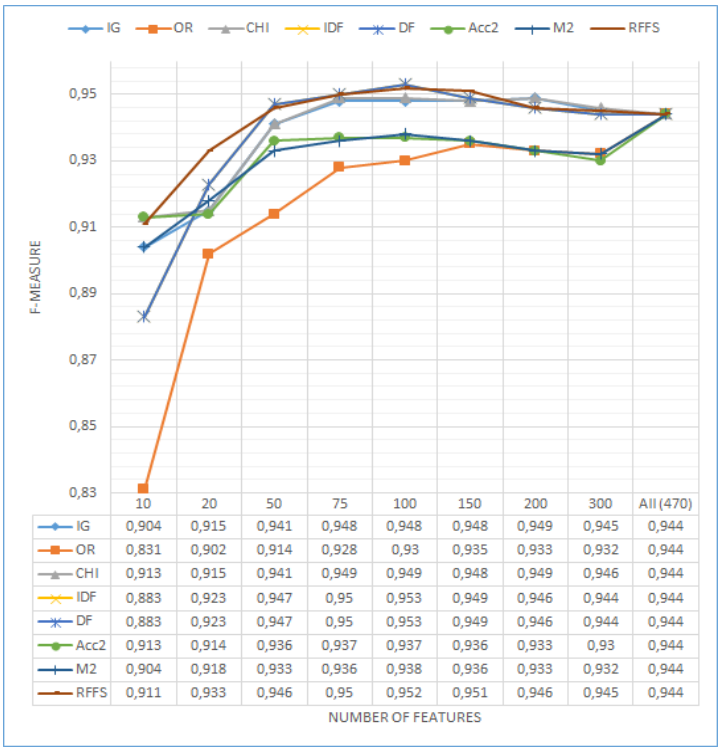


	10	20	50	75	100	150	200	300	All (470)
IG	0,885	0,889	0,885	0,886	0,885	0,885	0,885	0,885	0,886
OR	0,803	0,841	0,891	0,89	0,885	0,882	0,882	0,882	0,886
CHI	0,902	0,892	0,885	0,885	0,885	0,885	0,885	0,885	0,886
IDF	0,882	0,884	0,882	0,885	0,885	0,885	0,885	0,885	0,886
DF	0,882	0,884	0,882	0,885	0,885	0,885	0,885	0,885	0,886
Acc2	0,903	0,893	0,88	0,883	0,883	0,882	0,882	0,882	0,886
M2	0,888	0,894	0,881	0,883	0,883	0,882	0,882	0,882	0,886
RFFS	0,912	0,887	0,889	0,885	0,885	0,885	0,885	0,885	0,886

Şekil 3.2. NB algoritmasından elde edilen sonuçlar

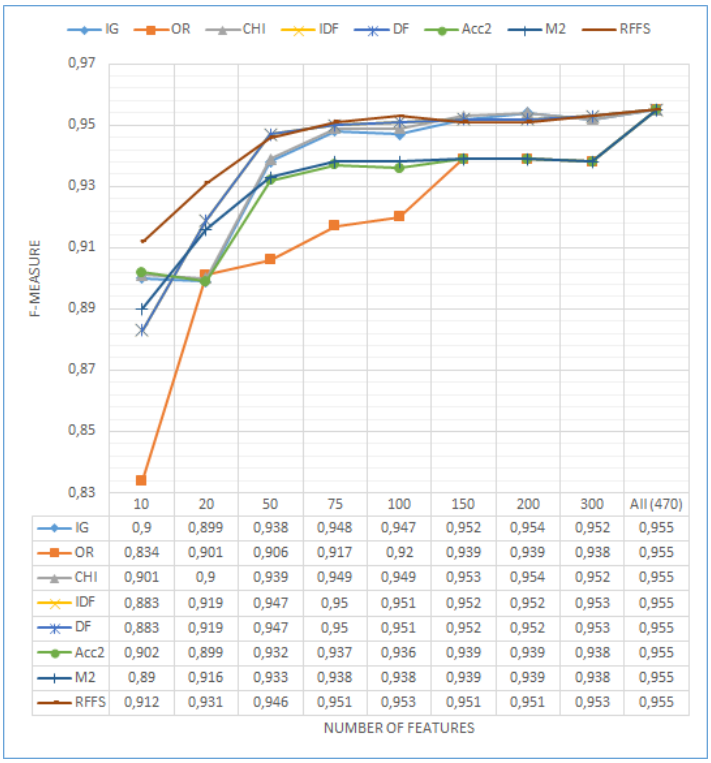
Şekil 3.3'te LR algoritmasından elde edilen sonuçlar verilmektedir. LR algoritmasından elde edilen en yüksek başarım 100 öznitelikte IDF ve DF metrikleri kullanılarak gerçekleştirilmektedir. Bu sonuç 0.953'tür. DF ve IDF metriklerinin ardından en yüksek başarım RFFS metriği ile 0.951 olarak bulunmaktadır. OR metriği dışında diğer metrikler ile 50 öznitelik seçildiğinde, neredeyse bütün özniteliklerin kullanılmasıyla elde edilen sonuçlara yakın sonuçlar bulunmaktadır. Ancak OR metriği diğer metriklerin gerisinde kalmaktadır.

Şekil 3.4'te SMO algoritmasından elde edilen sonuçlar verilmektedir. Bu algoritma ile elde edilen en yüksek başarım 0.955'tir. Bu sonuca herhangi bir öznitelik seçimi uygulanmadan ulaşılmaktadır. Öznitelik seçimi uygulandığında ise en yüksek başarım 200 öznitelikte CHI ve IG metrikleri ile elde edilmektedir. Bu sonuç 0.954'tür. En az öznitelik ile en yüksek başarım uyarlanan RFFS metriği ile elde edilmektedir. RFFS metriği ile 10 öznitelikte 0.912 başarım elde edilirken, 20 öznitelikte 0.931 başarım elde edilmektedir. Bu iki sonuç 10 ve 20 öznitelik kullanılarak SMO algoritmasından elde edilmiş en yüksek başarımlardır.



	10	20	50	75	100	150	200	300	All (470)
IG	0,904	0,915	0,941	0,948	0,948	0,948	0,949	0,945	0,944
OR	0,831	0,902	0,914	0,928	0,93	0,935	0,933	0,932	0,944
CHI	0,913	0,915	0,941	0,949	0,949	0,948	0,949	0,946	0,944
IDF	0,883	0,923	0,947	0,95	0,953	0,949	0,946	0,944	0,944
DF	0,883	0,923	0,947	0,95	0,953	0,949	0,946	0,944	0,944
Acc2	0,913	0,914	0,936	0,937	0,937	0,936	0,933	0,93	0,944
M2	0,904	0,918	0,933	0,936	0,938	0,936	0,933	0,932	0,944
RFFS	0,911	0,933	0,946	0,95	0,952	0,951	0,946	0,945	0,944

Şekil 3.3. LR algoritmasından elde edilen sonuçlar

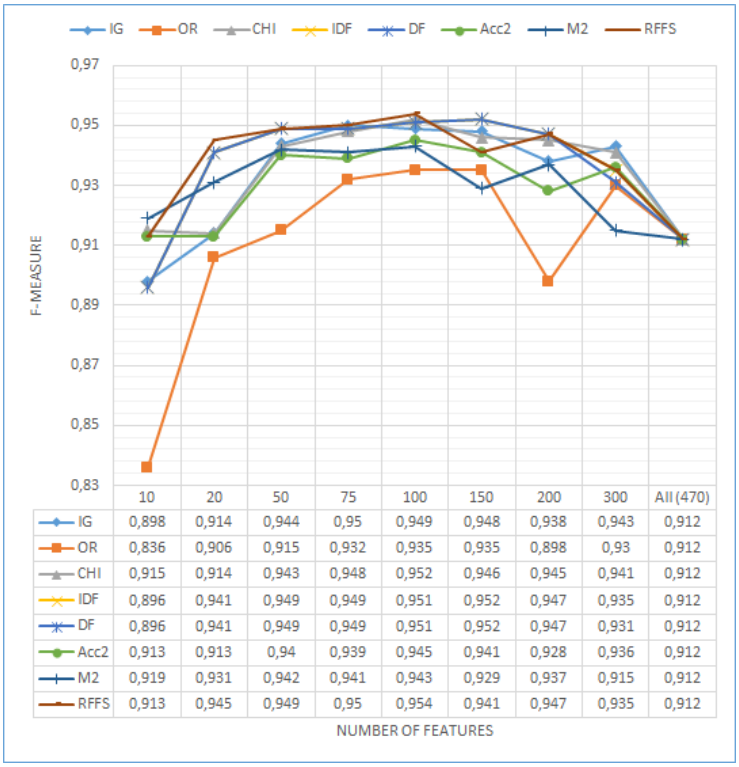


	10	20	50	75	100	150	200	300	All (470)
IG	0,9	0,899	0,938	0,948	0,947	0,952	0,954	0,952	0,955
OR	0,834	0,901	0,906	0,917	0,92	0,939	0,939	0,938	0,955
CHI	0,901	0,9	0,939	0,949	0,949	0,953	0,954	0,952	0,955
IDF	0,883	0,919	0,947	0,95	0,951	0,952	0,952	0,953	0,955
DF	0,883	0,919	0,947	0,95	0,951	0,952	0,952	0,953	0,955
Acc2	0,902	0,899	0,932	0,937	0,936	0,939	0,939	0,938	0,955
M2	0,89	0,916	0,933	0,938	0,938	0,939	0,939	0,938	0,955
RFFS	0,912	0,931	0,946	0,951	0,953	0,951	0,951	0,953	0,955

Şekil 3.4. SMO algoritmasından elde edilen sonuçlar

Şekil 3.5'te MLP algoritmasının sonuçları yer almaktadır. Bütün izinleri kullanmak bu algoritmanın sınıflandırma başarımını olumsuz etkilemektedir.

Öznitelik seçiminin etkisinin en iyi görüldüğü sınıflandırma algoritmasıdır. Öznitelik seçimi uygulanmadığında sınıflandırma başarımı 0.912 olurken 75 öznitelik ile sınıflandırma başarımının 0.95 olduğu görülmektedir. MLP algoritmasından elde edilen en yüksek başarım 0.954'tür. Bu sonuç RFFS metriği ile 100 öznitelik kullanıldığında elde edilmektedir.



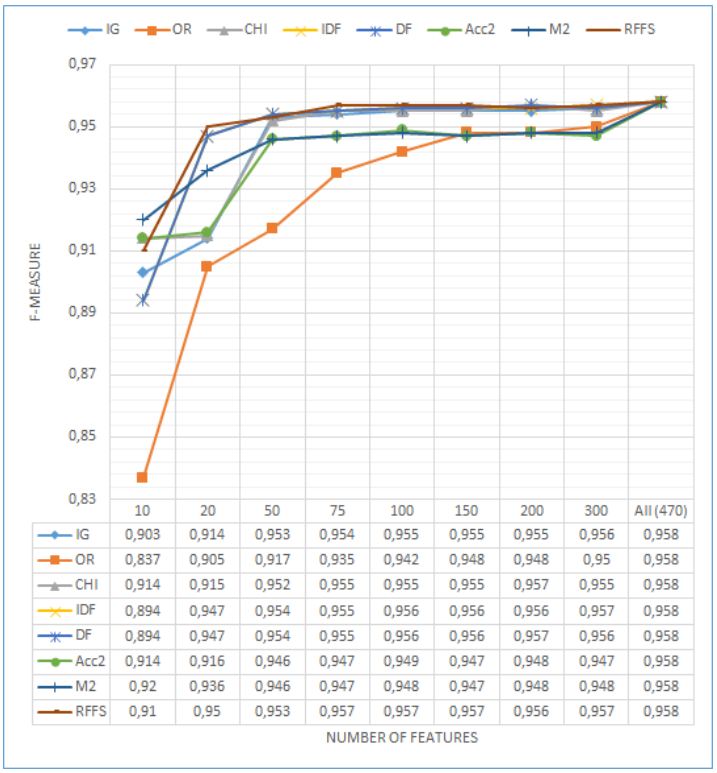
	10	20	50	75	100	150	200	300	All (470)
IG	0,898	0,914	0,944	0,95	0,949	0,948	0,938	0,943	0,912
OR	0,836	0,906	0,915	0,932	0,935	0,935	0,898	0,93	0,912
CHI	0,915	0,914	0,943	0,948	0,952	0,946	0,945	0,941	0,912
IDF	0,896	0,941	0,949	0,949	0,951	0,952	0,947	0,935	0,912
DF	0,896	0,941	0,949	0,949	0,951	0,952	0,947	0,931	0,912
Acc2	0,913	0,913	0,94	0,939	0,945	0,941	0,928	0,936	0,912
M2	0,919	0,931	0,942	0,941	0,943	0,929	0,937	0,915	0,912
RFFS	0,913	0,945	0,949	0,95	0,954	0,941	0,947	0,935	0,912

Şekil 3.5. MLP algoritmasından elde edilen sonuçlar

Şekil 3.6 ve Şekil 3.7'de sırasıyla C4.5 ve RF algoritmalarının sonuçları verilmektedir. Her iki algoritma için en yüksek sınıflandırma başarımı öznitelik seçimi kullanılmadan elde edilmektedir. C4.5 algoritması için elde edilen en yüksek başarım 0.946 olurken RF algoritması için elde edilen en yüksek başarım 0.958'dir. Hem C4.5 algoritmasında hem de RF algoritmasında 10 öznitelik kullanıldığında en iyi sınıflandırma başarımı M2 metriği ile elde edilmektedir. M2 metriği ile 10 öznitelik seçildiğinde, C4.5 algoritmasından elde edilen başarım 0.922 olurken RF algoritmasından elde edilen başarım ise 0.92 olmaktadır. Hem C4.5 algoritmasında hem de RF algoritmasında 20 öznitelik kullanıldığında en iyi sınıflandırma başarımı RFFS metriği ile elde edilmektedir. RFFS metriği ile 20 öznitelik seçildiğinde, C4.5 algoritmasından elde edilen başarım 0.945 olurken RF algoritmasından elde edilen başarım ise 0.95 olmaktadır.

	10	20	50	75	100	150	200	300	All (470)
IG	0,901	0,915	0,939	0,943	0,943	0,943	0,943	0,943	0,946
OR	0,826	0,904	0,916	0,932	0,936	0,938	0,937	0,937	0,946
CHI	0,914	0,916	0,943	0,943	0,943	0,944	0,943	0,943	0,946
IDF	0,897	0,939	0,945	0,945	0,944	0,944	0,944	0,944	0,946
DF	0,897	0,939	0,945	0,945	0,944	0,944	0,944	0,944	0,946
Acc2	0,914	0,917	0,936	0,937	0,938	0,937	0,937	0,937	0,946
M2	0,922	0,937	0,936	0,939	0,938	0,937	0,937	0,937	0,946
RFFS	0,914	0,945	0,945	0,945	0,945	0,944	0,944	0,944	0,946

Şekil 3.6. C4.5 algoritmasından elde edilen sonuçlar

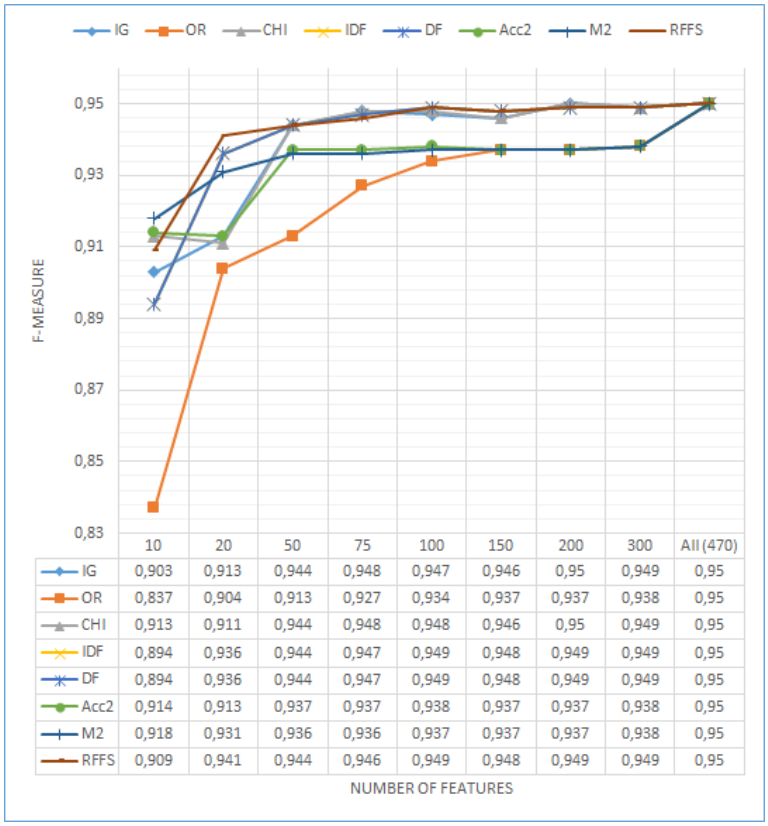


	10	20	50	75	100	150	200	300	All (470)
IG	0,903	0,914	0,953	0,954	0,955	0,955	0,955	0,956	0,958
OR	0,837	0,905	0,917	0,935	0,942	0,948	0,948	0,95	0,958
CHI	0,914	0,915	0,952	0,955	0,955	0,955	0,957	0,955	0,958
IDF	0,894	0,947	0,954	0,955	0,956	0,956	0,956	0,957	0,958
DF	0,894	0,947	0,954	0,955	0,956	0,956	0,957	0,956	0,958
Acc2	0,914	0,916	0,946	0,947	0,949	0,947	0,948	0,947	0,958
M2	0,92	0,936	0,946	0,947	0,948	0,947	0,948	0,948	0,958
RFFS	0,91	0,95	0,953	0,957	0,957	0,957	0,956	0,957	0,958

Şekil 3.7. RF algoritmasından elde edilen sonuçlar

Şekil 3.8'de KNN algoritmasının sonuçları verilmektedir. Öznitelik seçimi uygulanmadan elde edilen sınıflandırma başarımı 0.95 olmaktadır. Buna karşın IG ve CHI ile 200 öznitelik seçildiğinde bu başarım elde edilebilmektedir. Uyarlanan

metrikler göz önüne alınırsa RFFS ile 20 öznitelik seçildiğinde sınıflandırma başarımı 0.941'dir. Bu sonuç diğer metrikler tarafından seçilen 20 öznitelik arasından en iyisidir. Benzer bir şekilde 10 öznitelik kullanıldığında en iyi sınıflandırma başarımı Acc2 ve M2 metrikleri tarafından elde edilmektedir. Acc2 ve M2 metriklerinden elde edilen sonuçlar sırasıyla 0.914 ve 0.918'dir.



	10	20	50	75	100	150	200	300	All (470)
IG	0,903	0,913	0,944	0,948	0,947	0,946	0,95	0,949	0,95
OR	0,837	0,904	0,913	0,927	0,934	0,937	0,937	0,938	0,95
CHI	0,913	0,911	0,944	0,948	0,948	0,946	0,95	0,949	0,95
IDF	0,894	0,936	0,944	0,947	0,949	0,948	0,949	0,949	0,95
DF	0,894	0,936	0,944	0,947	0,949	0,948	0,949	0,949	0,95
Acc2	0,914	0,913	0,937	0,937	0,938	0,937	0,937	0,938	0,95
M2	0,918	0,931	0,936	0,936	0,937	0,937	0,937	0,938	0,95
RFFS	0,909	0,941	0,944	0,946	0,949	0,948	0,949	0,949	0,95

Şekil 3.8. KNN algoritmasından elde edilen sonuçlar

Bu bölümde yapılan çalışmanın ana motivasyonu, metin sınıflandırmada kullanılan çeşitli öznitelik seçme tekniklerini Android kötücül yazılım tespit sistemlerinde verimli bir şekilde kullanabilir miyiz sorusunun cevabını bulmaktır. Şekil 3.2'den Şekil 3.8'e kadar olan sonuçlar genel olarak incelendiğinde uyarlanan öznitelik seçme yöntemlerinin var olan yöntemlere göre az öznitelikte çoğunlukla başarılı olduğu görülmektedir. Bu nedenle araştırmanın ilk amacına ulaştığını düşünebiliriz. Bu sayede Android kötücül yazılım tespiti alanında çalışacak olan araştırmacılar uyarlanan bu öznitelik seçme yöntemlerini kendi sistemlerinde kolayca kullanabileceklerdir.

Öznitelik seçme yöntemleri tarafından seçilen en ayırt edici öznitelikler arasında bazı farklılıklar görülmektedir. Bu durum Tablo 3.3'te verilen izin listesinde

görülmektedir. İzin tablosunda metrikler tarafından seçilen en ayırt edici 10 izin yer almaktadır. 10 izin ele alındığında bile çeşitli farklılıklar gözlemlenmektedir. Bu nedenle farklı öznitelik seçme yöntemlerinden elde edilen en iyi öznitelikler bir araya getirilerek yeni öznitelik alt kümeleri oluşturulmaktadır. Metrikler tarafından seçilen en iyi öznitelikler bir araya getirildiği için açgözlü yaklaşıma dayalı öznitelik alt kümeleri oluşturma şeklinde isimlendirme yapılmaktadır. Bahsedilen yeni öznitelik alt kümeleri ve onların sonuçları şöyledir:

CHI, IG ve IDF metrikleri tarafından seçilen en iyi 10 öznitelik birleştirildiğinde toplam 22 eşsiz öznitelik çıkmaktadır. Bu özniteliklerden elde edilen başarımlar Tablo 3.4'te verilmektedir. 10 tane öznitelik kullanıldığı durumda CHI, IG ve IDF metrikleri arasında en yüksek başarım CHI metriği ile MLP algoritmasından elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.915'tir. Tablo 3.4'teki sonuçlar ile bu sonuç karşılaştırıldığında NB algoritması dışında bütün algoritmalardan elde edilen sonuçlarda dikkate değer bir iyileştirme yapıldığı görülmektedir. 20 tane öznitelik kullanıldığı durumda CHI metriğinden elde edilen en yüksek başarım 0.916 ile C4.5 algoritmasından elde edilirken, IG metriğinden elde edilen en yüksek başarım 0.915 ile LR ve C4.5 algoritmalarından elde edilmektedir. Açgözlü metrik birleştirme yaklaşımı ile IG ve CHI metriklerinden daha başarılı sonuçlar elde edildiği aşikârdır. Tablo 3.4'e göre LR algoritmasından elde edilen başarım 0.936 olurken, C4.5 algoritmasından elde edilen başarım ise 0.944'tür.

Tablo 3.4. Açgözlü yaklaşımından elde edilen 22 iznin kullanımı

	CHI+IG+IDF (22 izin)
Sınıflandırma Algoritması	**F-ölçütü**
NB	0.896
LR	0.936
SMO	0.926
MLP	0.937
KNN	0.940
C4.5	0.944
RF	0.940

CHI, IG ve IDF metrikleri tarafından seçilen en iyi 20 öznitelik birleştirildiğinde toplam 35 eşsiz öznitelik çıkmaktadır. Bu özniteliklerden elde edilen başarımlar Tablo 3.5'te verilmektedir. 20 tane öznitelik kullanıldığı durumda CHI, IG ve IDF metrikleri

arasında en yüksek başarım IDF metriği ile RF algoritmasından elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.947'dir. CHI, IG ve IDF metrikleri açgözlü yaklaşımla birleştirildiğinde RF algoritmasının performansı 0.949'a yükselmektedir. Bu küçük artış diğer algoritmalarda da gözlemlenmektedir. Açgözlü metrik birleştirme yaklaşımı sayesinde IG ve CHI metriklerinden elde edilen sonuçlara göre daha başarılı sonuçlar elde edilmektedir. Metriklerin birleştirilmesiyle elde edilen 35 öznitelik ile sınıflandırma yapıldığında en yüksek başarım RF algoritması ile elde edilmektedir. Elde edilen en düşük başarım ise NB algoritmasından alınmaktadır.

Tablo 3.5. Açgözlü yaklaşımından elde edilen 35 iznin kullanımı

	CHI+IG+IDF (35 izin)
Sınıflandırma Algoritması	**F-ölçütü**
NB	0.888
LR	0.933
SMO	0.929
MLP	0.942
KNN	0.938
C4.5	0.945
RF	0.949

RFFS, M2 ve Acc2 metrikleri tarafından seçilen en iyi 10 öznitelik birleştirildiğinde toplam 20 eşsiz öznitelik çıkmaktadır. Bu özniteliklerden elde edilen başarımlar Tablo 3.6'da verilmektedir. 10 tane öznitelik kullanıldığı durumda RFFS, M2 ve Acc2 metrikleri arasında en yüksek başarım M2 metriği ile C4.5 algoritmasından elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.922'dir. RFFS, M2 ve Acc2 metrikleri tarafından seçilen en iyi 10 izin birleştirilip 20 izin ile sınıflandırılma yapıldığında C4.5 algoritması ile 0.944 başarımı elde edilmektedir. Açgözlü yaklaşım ile bu üç metrik birleşiminin genel olarak iyi sonuçlar verdiği gözlemlenmektedir. Tablo 3.6'ya göre metriklerin birleştirilmesiyle elde edilen en yüksek başarım MLP ve RF algoritmalarından elde edilmektedir. Bu başarım 0.946'dır. Açgözlü yaklaşım ile metrik birleştirilmesi yapılmadığı durumda 20 öznitelik kullanılarak MLP üzerinde elde edilen en yüksek başarım 0.945 olmaktadır. Benzer şekilde açgözlü yaklaşım ile metrik birleştirilmesi yapılmadığı durumda 20 öznitelik kullanılarak NB üzerinde elde edilen en yüksek başarım 0.894'tür. Metrik birleştirme ile NB algoritması üzerinde %2 daha iyi başarım elde edilmektedir.

Tablo 3.6. Açgözlü yaklaşımından elde edilen 20 iznin kullanımı

	RFFS+Acc2+M2 (20 izin)
Sınıflandırma Algoritması	**F-ölçütü**
NB	0.904
LR	0.941
SMO	0.940
MLP	0.946
KNN	0.941
C4.5	0.944
RF	0.946

Tablo 3.7. Açgözlü yaklaşımından elde edilen 32 iznin kullanımı

	RFFS+Acc2+M2 (32 izin)
Sınıflandırma Algoritması	**F-ölçütü**
NB	0.894
LR	0.943
SMO	0.942
MLP	0.947
KNN	0.943
C4.5	0.945
RF	0.952

RFFS, M2 ve Acc2 metrikleri tarafından seçilen en iyi 20 öznitelik birleştirildiğinde toplam 32 eşsiz öznitelik çıkmaktadır. Bu özniteliklerden elde edilen başarımlar Tablo 3.7'de verilmektedir. 20 tane öznitelik kullanıldığı durumda RFFS, M2 ve Acc2 metrikleri arasında en yüksek başarım RFFS metriği ile RF algoritmasından elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.95'tir. RFFS, M2 ve Acc2 metrikleri tarafından seçilen en iyi 20 izin birleştirilip 32 izin ile sınıflandırılma yapıldığında RF algoritması ile 0.952 başarımı elde edilmektedir. Açgözlü yaklaşım ile bu üç metrik birleşiminin genel olarak iyi sonuçlar verdiği gözlemlenmektedir. Metriklerin birleştirilmesiyle elde edilen 32 öznitelik ile sınıflandırma yapıldığında en yüksek başarım RF algoritması ile elde edilmektedir. Elde edilen en düşük başarım ise NB algoritmasından alınmaktadır.

İzin tablosunda gözlemlenen farklılıklardan dolayı metrikler açgözlü yaklaşımla birleştirilip yeni öznitelik alt kümeleri oluşturulmaktadır. CHI, IG ve IDF metriklerinin birleşiminden iki farklı öznitelik alt kümesi ve RFFS, M2 ve Acc2

metriklerinin birleşiminden de iki farklı öznitelik alt kümesi olmak üzere toplamda 4 farklı öznitelik alt kümesi elde edilmektedir. Metriklerin açgözlü yaklaşımla ile birleşiminden ortaya çıkan sonuçlar değerlendirildiğinde sınıflandırma başarımlarında çoğunlukla iyileştirme görülmektedir. Üçlü metrik birleştirme dışında farklı sayıda metrikler açgözlü yaklaşıma benzer şekilde birleştirilerek farklı öznitelik alt kümeleri elde edilebilir. Bu sayede daha farklı sınıflandırma sonuçlarına ulaşılabilir.

3.6. Var Olan Çalışmalar ile Karşılaştırma

Bu alt bölümde, güncel Android kötücül yazılım tespit sistemi çalışmaları ile bu bölümde yapılan deneylerden elde edilen sonuçlar karşılaştırılacaktır. Çalışmada ana motivasyonumuz öznitelik seçimi olduğundan karşılaştırma yaptığımız Android kötücül yazılım tespit sistemlerinin altyapısında özellikle öznitelik seçme yöntemlerinin kullanılmasına dikkat edilerek karşılaştırma yapılmaktadır.

Tablo 3.8'de 16 çalışmanın sonuçları bu bölümde yapılan deneyler ile karşılaştırılmaktadır. Bazı çalışmalarda dinamik veya hibrit analize dayalı Android kötücül yazılım tekniklerinin kullanıldığı görülmektedir. Ancak analiz durumundan ziyade kötücül yazılım tespit sistemlerinin altyapısındaki öznitelik seçim yöntemlerini göz önünde bulundurularak değerlendirme yapılmaktadır. Bu çalışmada olduğu gibi (Abawajy vd, 2021; Bhattacharya ve Goswami, 2018; Dharmalingam ve Palanisamy, 2021; Fatima vd, 2019) çalışmalarında da öznitelik olarak izin kullanılmaktadır. (Fatima vd, 2019) bu bölümde yapılan deneylerde olduğu gibi tam anlamıyla dengeli veri kümesi kullanmaktadır. Diğer izin tabanlı çalışmalarda tam olarak dengeli veri kümesinin kullanıldığı söylenemez. (Fatima vd, 2019) boyut indirgemeyi hesaplama maliyeti fazla olan genetik algoritma ile yapmaktadır. Buna karşın genetik algoritmaya göre daha verimli çalışan filtreleme tabanlı teknikler ile (Fatima vd, 2019)'den daha iyi başarımlara ulaşılmaktadır. Diğer izin tabanlı yaklaşımlar ile kıyaslama yapıldığında, (Abawajy vd, 2021) ve (Dharmalingam ve Palanisamy, 2021)'den daha iyi sınıflandırma sonuçları elde edilmektedir. (Bhattacharya ve Goswami, 2018)'de iki farklı veri kümesi ile sonuç alınmaktadır. Bu bölümde yapılan deneylerden elde edilen sonuçlar bunların bir tanesinden iyi olurken diğerinden kötü olmaktadır.

(Abawajy vd, 2021; Bhattacharya ve Goswami, 2018; Dharmalingam ve Palanisamy, 2021; Fatima vd, 2019) çalışmaları dışında statik analize dayalı 9 çalışma yer almaktadır. Bu 9 çalışmanın sonuçları elde edilen sonuçlar ile karşılaştırıldığında,

bu bölümde yapılan deneylerden elde edilen sonuçlar (Alazab vd, 2020; Deepa vd, 2015; Kouliaridis vd, 2021)'nin sonucundan daha iyidir. Buna karşın diğer çalışmalar elde edilen sonuçlardan daha başarılıdır. Bunun sebebi diğer çalışmalarda izin dışında farklı öznitelik kullanımı tercih edilmesinden kaynaklanabilir. (Jung vd, 2021; Morales-Ortega vd, 2016) çalışmalarının sonuçları ile elde edilen sonuçlar arasında küçük farklılıklar görülmektedir. (Salah vd, 2020)'nin sonuçları elde edilen sonuçlara göre oldukça iyi görünse de aşırı dengesiz veri kümesi kullanılıyor olması göz ardı edilmemelidir. (Peynirci vd, 2020; Xu vd, 2016; Zhao vd, 2015)'nın sonuçları elde edilen sonuçlara kıyasla oldukça başarılır. Özellikle, (Peynirci vd, 2020) ve (Zhao vd, 2015) çalışmalarında dengeli veri kümesi kullanılıyor olması oldukça önemlidir. Sonuçlar genel olarak değerlendirildiğinde hem metin sınıflandırma alanından uyarlanan metriklerin hem de açgözlü arama stratejisine dayanan yaklaşımların az öznitelikte oldukça başarılı olduğu görünmektedir. Genel olarak literatürdeki sonuçlar ile elde edilen sonuçlar kıyaslandığında, uyarlanan metriklerin Android kötücül yazılım tespit sistemlerinde verimli bir şekilde kullanılabileceği gösterilmektedir.

Tablo 3.8. Önceki çalışmalar ile karşılaştırma

Çalışmalar	**Analiz**	**Veri kümesi Boyutu**	**Algoritma**	**Öznitelik Seçme Yöntemi**	**Performans**
(Dharmalingam ve Palanisamy, 2021)	S	1005 İyicil 1592 Kötücül	DNN	İzin Notlandırma Sistemi	0.9422 (Doğruluk)
(Ananya vd, 2020)	D	2475 İyicil 2474 Kötücül	XGBoost	SAILS	0.994 (F-ölçütü)
(Morales-Ortega vd, 2016)	S	1377 İyicil 1377 Kötücül	RF	Relief-f	0.9626 (Doğruluk)
(Peynirci vd, 2020)	S	8900 İyicil 8900 Kötücül	SVM	Delta_IDF	0.997 (Doğruluk)
(Shabtai vd, 2012)	D	40 İyicil 4 Kötücül	DT/J48	Bilgi Kazancı	0.999 (Doğruluk)
(Zhao vd, 2015)	S	3986 İyicil 3986 Kötücül	SVM	FrequenSel	0.975 (Doğruluk)
(Fatima vd, 2019)	S	20000 İyicil 20000 Kötücül	SVM	Genetik Algoritma	0.95 (Doğruluk)
(Xu vd, 2016)	S	12026 İyicil 5264 Kötücül	SVM	Korelasyon Tabanlı Öznitelik Seçme	0.974 (Doğruluk)
(Salah vd, 2020)	S	123453 İyicil 5560 Kötücül	SVM	FF-FA temelli TFIDF	0.99 (Doğruluk)
(Alazab vd, 2020)	S	13719 İyicil 14172 Kötücül	RF	Bilgi Kazancı ve Terim Frekansı	0.943 (F-ölçütü)
(Bhattacharya ve Goswami, 2018)	S	504 İyicil 213 Kötücül	–	Hibrit Topluluk Temelli Kaba Küme Teorisi	0.878 (Doğruluk)

Tablo 3.8. Önceki çalışmalar ile karşılaştırma (devamı)

(Bhattacharya ve Goswami, 2018)	S	2500 İyicil 1150 Kötücül	–	Hibrit Topluluk Temelli Kaba Küme Teorisi	0.9788 (Doğruluk)
(Z. Liu vd, 2021)	H	11000 İyicil 11000 Kötücül	RF	SRBM	0.867-0.870 (F-ölçütü)
(Jung vd, 2021)	S	27041 İyicil 26276 Kötücül	RF	Gini İndeksi	0.9651 (Doğruluk)
(Kouliaridis vd, 2021)	S	1000 İyicil 1000 Kötücül	Topluluk Öğrenmesi	t-SNE	0.9170 (Doğruluk)
(Deepa vd, 2015)	S	758 İyicil 612 Kötücül	Adaboost	Korelasyon Tabanlı Öznitelik Seçme	0.8875 (Doğruluk)
(Abawajy vd, 2021)	S	5500 İyicil 6190 Kötücül	LR	Bilgi Kazancı	≈ 0.9 (Doğruluk)
Önerilen açgözlü yaklaşım	S	3000 İyicil 3000 Kötücül	RF	CHI+IG+IDF (35 izin)	0.949 (F-ölçütü)
Önerilen açgözlü yaklaşım	S	3000 İyicil 3000 Kötücül	RF	RFFS+Acc2+M2 (32 izin)	0.952 (F-ölçütü)
Uyarlanan DF	S	3000 İyicil 3000 Kötücül	RF	DF (20 izin)	0.947 (F-ölçütü)
Uyarlanan Acc2	S	3000 İyicil 3000 Kötücül	C4.5	Acc2 (20 izin)	0.917 (F-ölçütü)
Uyarlanan M2	S	3000 İyicil 3000 Kötücül	C4.5	M2 (20 izin)	0.937 (F-ölçütü)
Uyarlanan RFFS	S	3000 İyicil 3000 Kötücül	RF	RFFS (20 izin)	0.95 (F-ölçütü)

S: Statik, **D:** Dinamik, **H:** Hibrit

4. DOĞRUSAL REGRESYON TABANLI ÖZNİTELİK SEÇME YÖNTEMİ KULLANILARAK ÖNERİLEN İZİN TABANLI KÖTÜCÜL YAZILIM TESPİT SİSTEMİ

Bu bölümde, önerilen doğrusal regresyon tabanlı öznitelik seçme yöntemini kullanarak geliştirilen izin tabanlı kötücül yazılım tespit sistemine değinilecektir. Bölüm 3'te anlatılan izin tabanlı kötücül yazılım tespit sisteminde doğal ve özel izinler öznitelik olarak kullanılmaktadır. Ancak bu bölümde tasarlanan kötücül yazılım tespit sisteminde sadece doğal izinler öznitelik olarak değerlendirilmektedir.

4.1. Bölüm Motivasyonu

Bölüm 3'te verildiği gibi Android kötücül yazılım tespiti üzerine çalışan araştırmacıların çoğunlukla öznitelik seçimi uygulamadığı görülmektedir (Pan vd, 2020; W. Wang vd, 2019). Bölüm 3'te yapılan deneylerde, elde edilen izinlerden en ayırt edici izinlerin %10'u ile %20'si kullanıldığında tüm izinlerin kullanılmasıyla elde edilen başarımlar kadar sınıflandırma başarımının elde edildiği gösterilmektedir. Başka bir ifadeyle, özniteliklerin neredeyse büyük çoğunluğunun sınıflandırma algoritmaları için ayırt edici özelliği gözlemlenememektedir.

Öznitelik seçme aşamasının birçok avantajı bulunmaktadır (Arauzo-Azofra vd, 2011). Bunlardan en önemlisi, öznitelik seçme boyut indirgeme sağladığı için sınıflandırma algoritmalarının eğitim aşamasında geçen sürenin azaltılmasıdır. Ayrıca ideal özelliklerin minimum miktarda belirlenmesi, analizin doğruluğunu artırmak ve model karmaşıklığını azaltmak için çok önemlidir (Feizollah vd, 2015). Mobil cihazların sınırlı donanıma sahip olduğu ve mobil cihaz üzerinde doğrudan çalışabilen gerçek zamanlı kötücül yazılım tespit sistemlerine ihtiyaç olduğu göz önünde bulundurulursa öznitelik seçme aşaması kaçınılmaz olmaktadır. Bu sebeplerden dolayı, öznitelik seçme aşamasının kullanıldığı kötücül yazılım tespit sistemlerine ihtiyaç vardır. Buradan yola çıkarak, geliştirilen Android kötücül yazılım tespit sistemi içerisinde doğrusal regresyon tabanlı öznitelik seçme yöntemi önerilmektedir. Doğrusal regresyon tabanlı öznitelik seçme yönteminin kullanılmasının iki ana sebebi vardır. İlk olarak, doğrusal regresyon yönteminin matematiksel altyapısı karmaşık değildir. İkinci ise, kolayca mobil cihaz üzerine entegre edilebilmektedir. Ayrıca mobil cihazlar gibi sınırlı kaynağa sahip duyargalar üzerinde doğrusal regresyon yöntemi

kullanılmaktadır (Khandoker vd, 2011). Bu sebeple mobil cihaz üzerinde çalışan gerçek zamanlı kötücül yazılım tespit sisteminde doğrusal regresyon kolaylıkla çalıştırılabilecektir.

4.2. Bölüm Katkısı

Bu bölümün ana katkıları şöyle özetlenebilir:

- Doğrusal regresyon tabanlı özellik seçme yöntemi kullanılarak yeni bir Android kötü amaçlı yazılım tespit sistemi önerilmiştir. Önerilen sistem, makine öğrenimine dayalı statik Android kötü amaçlı yazılım tespitidir.
- Android kötü amaçlı yazılım tespit sistemlerinde özellik seçimi önemli olduğundan, bu çalışmada doğrusal regresyon tabanlı özellik seçim yöntemi önerilmiştir. Bildiğimiz kadarıyla, doğrusal regresyon tabanlı öznitelik seçimi Android kötü amaçlı yazılım tespitinde daha önceden kullanılmamıştır.
- Tüm izinleri kullanmak yerine en ayırt edici izinlerin seçilmesi sağlanır böylece sınıflandırma algoritmalarının performansı iyileştirilmiş olur.
- Tüm izinler kullanıldığında en yüksek başarım MLP algoritması ile elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.963'tür. Buna karşın, sadece 27 izin kullanıldığında MLP algoritması ile elde edilen başarım 0.961'dir. Bu iki sonuç arasındaki fark oldukça küçüktür. Bu sonuç iyi öznitelik seçiminin yapıldığını göstermektedir.
- Öznitelik seçme yönteminin yanında, çalışma Android kötü amaçlı yazılım tespiti alanında çalışmalar yapacak araştırmacılara doğrudan kullanabilecekleri çeşitli izin grupları sunmaktadır. Bu izin grupları 7 farklı makine öğrenmesi algoritması altında değerlendirilerek geniş kapsamlı analizler yapılmaktadır.

4.3. Öznitelik Çıkarımı

APK, Android işletim sisteminin uygulamaları dağıtmak ve yüklemek için kullandığı dosya biçimidir. Bir uygulamanın cihaza doğru bir şekilde yüklenebilmesi için gereken tüm öğeleri içerir. Uygulama kaynak kodları, resimler, API çağrıları, uygulama izinleri gibi birçok öğe bu dosya içerisinde yer almaktadır. APK dosyaları sıkıştırılmış dosyalar gibi düşünülebilir. Bu nedenle uygulama dosyalarından gerekli bilgilerin elde edilebilmesi için APK dosyalarının açılması gerekmektedir.

Tablo 4.1. Öznitelik vektörünü oluşturan algoritma

Algoritma: Öznitelik vektörünün oluşturulması

Girdi: $Tüm\ uygulamalar\ ve\ standart\ izinler$

Çıktı: ö$znitelik_vektör$ü → İki boyutlu dizi: Dizinin satır sayısı uygulama sayısını göstermektedir. Sütun sayısı ise izin sayısını göstermektedir.

Function $ÖZNİTELİK\_LİSTESİ\_OLUŞTUR(Uygulama\ [\ ], Standartİzin[\ ])$

ö$znitelik_vektör$ü $=[\ ][\ ]$ → İki boyutlu boş bir dizi oluştur

$N_1 = length(Uygulama)$

For $i = 1\ to\ N_1$ **do**

AAPT2'yi çağır

$Uygulama[i]$'yi AAPT2 ile işle

$Uygulama[i]$'nin AndroidManifest.xml dosyasını elde et

$N_2 = length(Standartİzin)$

For $j = 1\ to\ N_2$ **do**

If $Standartİzin[j]$ $Uygulama[i]$'nin AndroidManifest. xml dosyasında bulunuyor mu **then**

ö$znitelik_listesi[i][j] \leftarrow 1$

Else

ö$znitelik_listesi[i][j] \leftarrow 0$

End If

End For

End For

Return ö$znitelik_vektör$ü

End Function

Çalışmada her uygulamaya ait dosyalar AAPT2 (AAPT2, 2021) aracı kullanılarak açılmaktadır. Öznitelik olarak doğal uygulama izinleri kullanıldığından, AndroidManifest.xml dosyasına erişilerek uygulamaların kullandıkları izinler elde edilmektedir. Tüm izinleri kullanmak yerine Android işletim sisteminin uygulama geliştiricilerine tanıdığı standart izinler (Permissions, 2021) göz önünde bulundurularak özellik vektörünün oluşturulması sağlanmaktadır. APK dosyalarının

işlenmesi ve özellik vektörünün oluşturulması Tablo 4.1'de verilen algoritma ile gerçekleştirilmektedir.

Tablo 4.1'deki algoritmanın çalıştırılmasıyla veri kümesinde yer alan bütün uygulamaların hangi standart izinleri istedikleri veya istemedikleri bilgisine göre özellik vektörü oluşturulmaktadır. Bir uygulama ilgili izini talep ediyorsa 1, talep etmiyorsa 0 şeklinde ağırlıklandırma yapılarak özellik vektörü matematiksel bir yapıya dönüştürülmektedir. Elde edilen özellik vektörü incelendiğinde bazı izinlerin hiçbir uygulama tarafından kullanılmadığı tespit edilmektedir. Bu izinlerin özellik vektöründen elenmesiyle toplam 102 izin değerlendirilmeye alınmaktadır. Tüm uygulamaların ele alınmasıyla Şekil 4.1'de bir parçası verilen işlenmiş veri kümesi meydana gelmektedir. Şekil 4.1'de, 102 izin için 1 iyicil ve 1 kötücül uygulamanın özellik vektörü gösterilmektedir.

```
0,0,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,
1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,malware

1,0,1,1,1,0,1,1,0,0,0,0,0,0,0,1,1,0,0,
0,0,1,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,
0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,
0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,
1,0,benign
```

Şekil 4.1. Elde edilen işlenmiş veri kümesinin bir parçası

4.4. Doğrusal Regresyon Tabanlı Öznitelik Seçme Yöntemi

Doğrusal regresyon, iki ya da daha çok değişken arasındaki ilişkiyi modellemede kullanılan istatistiksel bir metottur (Douglas C. vd, 2012). Bağımlı değişkeni tahmin etmek için kurulan modelde giriş olarak tek bir bağımsız değişken kullanılıyorsa tekli regresyon, birden fazla bağımsız değişken kullanılıyorsa çoklu regresyon olarak adlandırılır. Bu çalışmada uygulama izinleri bağımsız değişkenlere karşılık gelirken, bağımlı değişken ise uygulamaların türünü temsil etmektedir. 102 adet bağımsız değişken kullanıldığından oluşturulan model çoklu regresyona örnektir. Regresyon modelinin oluşturulabilmesi için bağımlı değişkenin nümerik olması gerekmektedir.

Bu nedenle Şekil 4.1'de yer alan uygulama etiketleri "malware" uygulamalar için 0 olurken, "benign" uygulamalar için 1 olmaktadır.

Regresyon analizi ile bağımlı ve bağımsız değişkenler arasındaki ilişkinin varlığı hesaplanmaktadır. n adet bağımsız değişken ile bağımlı değişken arasındaki doğrusal ilişki Eşitlik 4.1'de verildiği gibidir.

$$Y = b_0 + b_1X_1 + b_2X_2 + \cdots + b_nX_n + e \tag{4.1}$$

Eşitlik 4.1'de, Y bağımlı değişkeni gösterirken, X_1, X_2, $X_3 \ldots X_n$ ise bağımsız değişkenleri başka bir ifadeyle uygulama izinlerini göstermektedir. b_1, b_2, ... b_n değerleri modeldeki katsayılar olup, b_0 doğrunun y eksenini kestiği noktayı göstermektedir. e ise hata terimi olarak tanımlanmaktadır. Bu katsayılar en küçük kareler yöntemi ile bulunmaktadır. En küçük kareler yaklaşımı kullanılarak Eşitlik 4.2'de verilen tahmin hatası sıfıra indirgenmeye çalışılmaktadır.

$$SSE = \sum_{i=1}^{n} (y_i - \bar{y_i})^2 \tag{4.2}$$

Eşitlik 4.2'de n toplam veri sayısını, y_i gözlemlenmiş gerçek veriyi, $\bar{y_i}$ ise modelin tahmin ettiği değeri temsil etmektedir. SSE ise tahmin hatalarının karesel toplamıdır. Doğrusal regresyonda SSE değeri, her katsayı farklılaştırıldıktan sonra sıfıra eşitlenmeye veya minimize edilmeye çalışılır. Bu sayede Eşitlik 4.1'de gösterilen çoklu doğrusal regresyon denklemi elde edilir. Eşitlik 4.3'te en küçük kareler yönteminden yararlanılarak 2 bağımsız değişkenden oluşan çoklu doğrusal regresyon modelinin $(y = a + bx_1 + cx_2)$ elde edilmesi gösterilmektedir. Eşitlik 4.3'te a, b ve c katsayıları, x_1 ve x_2 bağımsız değişkenleri, y ise bağımlı değişkeni temsil etmektedir.

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_{1,i} & \sum_{i=1}^{n} x_{2,i} \\ \sum_{i=1}^{n} x_{1,i} & \sum_{i=1}^{n} x_{1,i}^2 & \sum_{i=1}^{n} x_{1,i} \cdot x_{2,i} \\ \sum_{i=1}^{n} x_{2,i} & \sum_{i=1}^{n} x_{1,i} \cdot x_{2,i} & \sum_{i=1}^{n} x_{2,i}^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_{1,i} \cdot y_i \\ \sum_{i=1}^{n} x_{2,i} \cdot y_i \end{bmatrix} \tag{4.3}$$

Tablo 4.2. Elenecek özniteliklerin belirlenmesini sağlayan algoritma

Algoritma: Elenecek özniteliklerin belirlenmesi

Girdi: $Doğrusal\ regresyon\ modeli$

Çıktı: $elenen\_öznitelikler$

Function $ÖZNİTELİK\_SEÇİMİ\_GERÇEKLEŞTİR(RegresyonModeli)$

Doğrusal regresyon modelini bağımsız değişken ve katsayı çiftlerine dönüştür

Elde edilen çiftleri $Model$ adlı bir çırpı veri yapısında sakla → Çırpı veri yapısı kullanılarak doğrusal regresyon modeli, bağımsız değişkenler ve katsayıları şeklinde çiftlere dönüştürülür. Oluşturulan çırpı veri yapısında anahtarlar bağımsız değişkenler iken, anahtarlara karşılık gelen değerler katsayılardır.

$N_1 = length(Model)$

For $i = 1\ to\ N_1$ **do**

If $Model[Model.keys[i]] < 0.1$ **then**

If $Model[Model.keys[i]] > -0.1$ **then**

$elenen\_öznitelikler \leftarrow Model.keys[i]$

End If

End If

End For

Return $elenen\_öznitelikler$

End Function

Her bir bağımsız değişkene karşılık gelen katsayıların hesaplanmasıyla bağımsız değişkenlerin bağımlı değişkenin tahminine ne derece katkı yaptığı belirlenmektedir. Elde edilen veriye göre çok değişkenli doğrusal regresyon modeli oluşturulduğunda katsayıların -1 ile 1 aralığında değerler aldığı gözlemlenmektedir. Şekil 4.1 göz önünde bulundurulursa, işlenmiş veri kümesinin çoğunlukla 0 değerinden oluşan seyrek matris olduğu görülmektedir. Bu durum bazı izinlere ait katsayıların 0 veya 0'a yakın olmasını beraberinde getirmektedir. Buradan yola çıkarak Tablo 4.2'de verilen algoritma vasıtasıyla 0 ve 0'a yakın katsayılara sahip izinlerin aranması sağlanmaktadır. Ardından belirlenen izinlerin elenmesiyle de öznitelik seçme işlemi gerçekleştirilmektedir.

4.5. Önerilen İzin Tabanlı Kötücül Tespit Sisteminin Altyapısı

Android işletim sistemleri için tasarlanan kötücül yazılım tespit sistemi Şekil 4.2'de verilmektedir. Verilen sistemin çalışma adımları şöyledir:

Adım 1: Önerilen kötücül yazılım tespit sisteminin başarımını tam olarak değerlendirebilmek için ilk olarak veri kümesi 10 parçaya ayrılmaktadır. Bu sayede 10 katlı çapraz doğrulama yapma imkanı elde edilmektedir.

Adım 2: APK dosyaları üzerinde Tablo 4.1'de verilen Algoritma çalıştırılarak uygulama izinlerine bağlı özellik vektörü elde edilmektedir.

Adım 3: Özellik vektörünün eğitim için ayrılan parçasına çoklu doğrusal regresyon modeli uygulanmaktadır.

Adım 4: Adım 3'te elde edilen model Tablo 4.2'de detaylandırılan algoritmaya girdi olarak verilerek ayırt edici özelliği düşük öznitelikler belirlenmektedir.

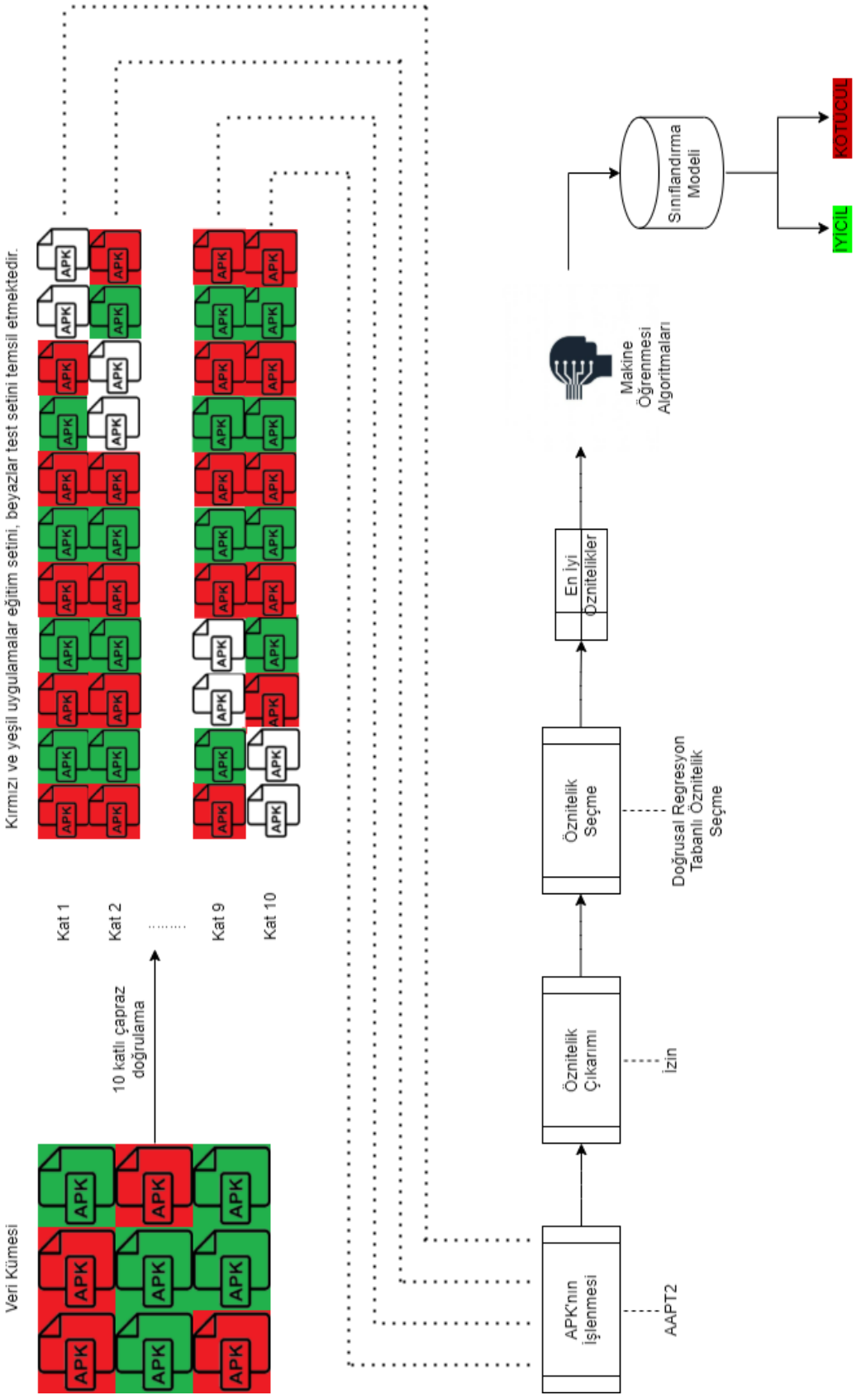
Adım 5: Ayırt edici özelliği düşük olan öznitelikler özellik vektöründen kaldırılarak öznitelik seçme adımı gerçekleştirilmektedir.

Adım 6: Veri kümesinin eğitim parçası çeşitli makine öğrenmesi algoritmalarına verilerek sınıflandırma modeli oluşturulmaktadır.

Adım 7: Veri kümesinin test parçası Adım 6'da oluşturulan modele verilerek etiketi bilinmeyen uygulamaların türleri tahmin edilmektedir.

Adım 8: İlgili çapraz doğrulama parçasının sınıflandırma başarımı hesaplandıktan sonra Adım 2'den Adım 8'e kadar anlatılan tüm işlemleri gerçekleştirmek üzere bir başka çapraz doğrulama parçasına geçilmektedir.

Bu 8 adım gerçekleştirilerek çalışmada sunulan kötücül yazılım tespiti gerçekleştirilmektedir. Önerilen yapının en önemli avantajı makine öğrenmesi algoritmaları için gerekli izin sayısının doğrusal regresyona dayalı öznitelik tekniği ile azaltılmasıdır. Doğrusal regresyon tekniğinin matematiği zor değildir. Bu nedenle kötücül yazılım tespit sistemlerinde kolayca kullanılabilir. Özellikle, genetik algoritma, PSO gibi meta sezgisel algoritmaların yapısı karmaşık ve hesaplama maliyetleri oldukça fazladır. Bu algoritmalara göre önerilen yaklaşımın mobil cihaz



Şekil 4.2. Önerilen kötücül yazılım tespit sisteminin mimarisi

üzerinde çalıştırılacak gerçek zamanlı kötücül yazılım sistemlerine eklenmesi daha elverişlidir.

Mobil kötücül yazılım tespit sistemlerinde genellikle üç adımın olması beklenir. Bunlar tespit doğruluğu, gerçek zamanlı tespit desteği ve ekonomik kaynak tüketimidir. Önerilen tespit sisteminin başarımı %96'lara çıkmaktadır. Önerilen sistem içerisinde boyut azaltma tekniği olduğu için sınıflandırma modeli oluşturulurken bellek daha verimli bir şekilde kullanılacaktır. Ayrıca öznitelik sayısının az olması makine öğrenmesi algoritmalarının çalışma zamanını önemli miktarda azaltacaktır. Bu durumlar göz önünde bulundurularak önerilen kötücül yazılım tespit sisteminin ihtiyaçları karşıladığı görülmektedir.

4.6. Deneysel Ayarlamalar

Bu bölümde gerçekleştirilen deneylerde 2000 tane uygulamadan oluşan veri kümesi kullanılmaktadır. Bu uygulamalardan 1000 tanesi kötücül olurken geriye kalan 1000 uygulama ise iyicildir. İyicil uygulamalar birçok Android kullanıcısının uygulama edinmek için kullandığı APKPure'dan indirilmektedir (APKPure, 2021). Kötücül uygulamalar ise AMD veri kümesi içerisinden rastgele seçilmektedir (AMD, 2019; F. Wei vd, 2017).

Yedi farklı sınıflandırma algoritması kullanılarak öznitelik seçme yöntemleri karşılaştırılmaktadır. Bunlar KNN, NB, SMO, MLP, RF, C4.5 ve LR algoritmalarıdır. Tüm algoritmalar için WEKA paketinden yararlanılmaktadır (Hall vd, 2009). KNN'de k değeri 1 seçilererek sınıflandırma işlemi gerçekleştirilmektedir. Diğer algoritmalarda WEKA paketinde varsayılan parametreler kullanılmaktadır. Son olarak F-ölçütü metriği ile sınıflandırma başarımları hesap edilerek her bir yöntemin başarımları değerlendirilmektedir.

4.7. Elde Edilen Sonuçlar

Bu bölümde, deneylerden elde edilen sonuçlar verilecektir. İlk olarak, her bir kat üzerine önerilen çoklu doğrusal regresyon tabanlı öznitelik seçme modelinin uygulanmasıyla oluşturulan en ayırt edici izinler listelenecektir. İkinci olarak öznitelik seçiminin yapılmadığı durumda elde edilen sınıflandırma başarımları ile öznitelik seçiminin yapılması durumunda elde edilen sınıflandırma başarımları karşılaştırmalı olarak verilecektir. Son olarak her bir kattan elde edilen izin sayısı göz önüne alınarak izinler çoğunluk oylamasına göre birleştirilmektedir.

Tablo 4.3'te, K 10 katlı çapraz doğrulamadaki her bir katı temsil ederken, TSS ise toplam seçim sayısını göstermektedir. Tablo 4.3'te her bir kat üzerinde ayrım gücü en iyi izinler yer almaktadır. Veri kümesi 10 parçaya ayrıldığı için her bir katın eğitim aşamasında kullanılan Android uygulamalar da farklıdır. Bu nedenle eğitim aşamasından önce uygulanan öznitelik seçme yönteminin her kat üzerinde seçtiği izinler arasında farklılıklar görülmektedir. Örneğin kat 1'de 50 izin seçilirken, kat 2'de 57 izin seçilmektedir. Bazı katlarda aynı sayıda izin seçilmesine karşın bu izinler tamamen aynı değildir. Kat 1 - kat 10 ve kat 2 - kat 3 bu duruma örnek olarak verilebilir. Hem kat 1'de hem kat 10'da 50 izin seçilmesine karşın seçilen izinler arasında farklılıklar vardır. Benzer durum kat 2 ve kat 3'te görülmektedir. Bütün durumlar göz önünde bulundurulduğunda, en az sayıda özniteliğin 42 izne sahip kat 4 olduğu görülürken en fazla sayıda özniteliğin 57 izne sahip kat 2 ve kat 3'te olduğu görülmektedir.

Tablo 4.3. Her kat için en iyi ayrım gücüne sahip izinler

10 Katlı Çapraz Doğrulama / **İzinler**	**K 1**	**K 2**	**K 3**	**K 4**	**K 5**	**K 6**	**K 7**	**K 8**	**K 9**	**K10**	**TSS**
android.permission. ACCESS_LOCATION_EXTRA_ COMMANDS	+	+	+	+	+	+	+	+	+	+	10
android.permission. ACCESS_CHECKIN_PROPERTIE S	+	+	+	-	+	+	+	+	+	+	9
android.permission. ACCESS_COARSE_LOCATION	+	+	+	+	+	+	+	+	+	+	10
android.permission. ACCESS_NETWORK_STATE	+	+	+	+	+	+	+	+	+	+	10
android.permission. ACCESS_NOTIFICATION_POLIC Y	+	-	-	+	+	-	+	+	+	+	7
android.permission. ACCESS_WIFI_STATE	+	+	+	+	+	+	+	+	+	+	10
android.permission. ANSWER_PHONE_CALLS	+	+	+	+	+	+	+	-	+	+	9
android.permission. BIND_ACCESSIBILITY_SERVICE	+	+	+	+	+	+	+	+	+	+	10
android.permission. BLUETOOTH	+	+	+	+	+	+	+	+	+	+	10
android.permission. CALL_PHONE	+	+	+	+	+	+	+	+	+	+	10
android.permission. CAMERA	+	+	+	+	+	+	+	+	+	+	10
android.permission .CHANGE_COMPONENT_ENABL ED_ STATE	+	+	+	-	+	+	+	-	+	+	8

Tablo 4.3. Her kat için en iyi ayrım gücüne sahip izinler (devamı)

android.permission. CHANGE_WIFI_MULTICAST_STAT E	+	+	+	-	+	-	+	+	+	-	7
android.permission. CHANGE_WIFI_STATE	+	+	+	+	+	+	+	+	+	+	10
android.permission. FOREGROUND_SERVICE	+	+	+	+	-	-	+	+	+	+	8
android.permission. GET_PACKAGE_SIZE	+	-	+	+	+	+	+	+	+	+	9
com.android.launcher. permission.INSTALL_SHORTCUT	+	+	+	+	+	+	+	+	+	+	10
android.permission. MODIFY_AUDIO_SETTINGS	+	-	+	-	-	-	-	-	-	-	2
android.permission. MODIFY_PHONE_STATE	+	+	+	+	+	-	+	-	+	+	8
android.permission. MOUNT_UNMOUNT_FILESYSTEMS	+	+	+	-	-	-	+	+	+	-	6
android.permission. PACKAGE_USAGE_STATS	+	+	+	+	+	+	+	+	+	+	10
android.permission. PERSISTENT_ACTIVITY	+	-	-	-	-	-	-	-	-	-	1
android.permission. PROCESS_OUTGOING_CALLS	+	+	+	+	+	+	+	+	+	+	10
android.permission. READ_CALENDAR	+	+	+	-	+	-	+	+	+	+	8
android.permission. READ_CALL_LOG	+	+	+	+	+	+	+	+	+	+	10
android.permission. READ_CONTACTS	+	+	+	+	+	+	+	+	+	+	10
android.permission. READ_EXTERNAL_STORAGE	+	+	+	+	+	+	+	+	+	+	10
android.permission. READ_LOGS	+	+	+	+	+	+	+	+	+	+	10
android.permission. READ_PHONE_STATE	+	+	+	+	+	+	+	+	+	+	10
android.permission. READ_SMS	+	+	-	+	+	+	+	-	+	+	8
android.permission. REBOOT	+	+	+	+	+	+	+	-	+	+	9
android.permission. RECEIVE_BOOT_COMPLETED	+	+	+	+	+	+	+	+	+	+	10
android.permission. RECEIVE_SMS	+	+	+	-	+	+	+	+	+	+	9
android.permission. RECORD_AUDIO	+	+	+	+	+	+	+	+	+	+	10
android.permission. REQUEST_INSTALL_PACKAGES	+	-	+	+	+	+	+	+	+	+	9
android.permission. SEND_SMS	+	+	+	+	+	+	+	+	+	+	10
com.android.alarm.permission.SET_ ALARM	+	+	+	+	+	+	+	+	+	+	10
android.permission. SET_DEBUG_APP	+	-	+	+	-	+	+	+	+	+	8
android.permission. SET_PREFERRED_APPLICATIONS	+	-	+	+	+	+	+	-	-	-	6
android.permission. SET_WALLPAPER	+	+	+	-	+	+	+	+	+	+	9

Tablo 4.3. Her kat için en iyi ayrım gücüne sahip izinler (devamı)

android.permission. SYSTEM_ALERT_WINDOW	+	+	+	+	+	+	+	+	+	+	10
android.permission. USE_BIOMETRIC	+	+	+	+	+	+	+	-	+	+	9
android.permission. USE_SIP	+	+	+	-	+	-	-	-	-	-	4
android.permission. VIBRATE	+	+	+	+	+	+	+	+	+	+	10
android.permission. WAKE_LOCK	+	+	+	+	+	+	+	+	+	+	10
android.permission. WRITE_APN_SETTINGS	+	+	+	+	+	+	+	+	+	+	10
android.permission. WRITE_CALENDAR	+	+	+	-	+	-	+	+	+	+	8
android.permission. WRITE_EXTERNAL_STORAGE	+	+	+	+	+	+	+	+	+	+	10
android.permission. WRITE_SETTINGS	+	+	+	+	+	+	+	+	+	+	10
android.permission. WRITE_SYNC_SETTINGS	+	-	-	-	-	-	-	-	+	-	2
android.permission. ACCOUNT_MANAGER	-	+	+	-	+	-	-	+	-	-	4
android.permission. BIND_APPWIDGET	-	+	-	-	-	-	-	-	-	-	1
android.permission. CALL_PRIVILEGED	-	+	-	-	-	-	-	-	-	-	1
android.permission. CHANGE_NETWORK_STATE	-	+	-	-	+	+	-	-	-	-	3
android.permission. DISABLE_KEYGUARD	-	+	-	+	-	-	-	+	+	-	4
android.permission. GET_ACCOUNTS	-	+	+	-	+	+	-	+	-	-	5
android.permission. INSTALL_LOCATION_PROVIDER	-	+	+	-	+	-	-	-	-	-	3
android.permission. INSTALL_PACKAGES	-	+	+	-	+	+	-	+	-	+	6
android.permission. READ_SYNC_SETTINGS	-	+	+	+	+	+	+	+	-	+	8
android.permission. REQUEST_DELETE_PACKAGES	-	+	-	-	-	-	-	-	+	-	2
android.permission. SET_PROCESS_LIMIT	-	+	-	-	-	-	-	-	-	+	2
android.permission. SET_TIME_ZONE	-	+	-	-	-	-	-	+	-	-	2
android.permission. TRANSMIT_IR	-	+	+	-	+	-	-	+	+	+	6
android.permission. USE_FINGERPRINT	-	+	+	-	+	+	-	+	-	+	6
android.permission. WRITE_SECURE_SETTINGS	-	+	+	-	+	-	-	-	-	+	4
android.permission. REORDER_TASKS	-	-	+	-	+	+	-	+	+	+	6
android.permission. STATUS_BAR	-	-	+	+	+	-	-	+	-	-	4
android.permission. WRITE_CALL_LOG	-	-	+	-	-	-	-	-	+	-	2
android.permission. DUMP	-	-	-	+	-	-	-	-	-	-	1

Tablo 4.3. Her kat için en iyi ayrım gücüne sahip izinler (devamı)

android.permission. CLEAR_APP_CACHE	-	-	-	-	+	-	+	+	-	-	3
android.permission. BODY_SENSORS	-	-	-	-	-	-	-	+	+	-	2
android.permission. BROADCAST_STICKY	-	-	-	-	-	-	-	+	-	-	1
android.permission. READ_PHONE_NUMBERS	-	-	-	-	-	-	-	+	-	-	1
com.android.voicemail. permission.WRITE_VOICEMAIL	-	-	-	-	-	-	-	+	-	-	1
android.permission. SET_WALLPAPER_HINTS	-	-	-	-	-	-	-	-	+	-	1

Kat 4'te yer alan eğitim setine doğrusal regresyon yöntemi uygulandığında, 102 iznin katsayısı Şekil 4.3'te verildiği gibi oluşmaktadır. Katsayıları daha iyi yorumlayabilmek için katsayılar küçükten büyüğe doğru sıralanmaktadır. Katsayılardan en küçüğü -1.0262 değeridir. Bu değer android.permission.USE_BIOMETRIC izninin katsayısıdır. Katsayılardan en büyüğü ise 0.7785 değeridir. Bu değer android.permission.SET_PREFERRED_APPLICATIONS izninin katsayısıdır. Katsayısı 0 olan iki izin vardır. Bunlar android.permission.ACCESS_CHECKIN_PROPERTIES ve android.permission.BROADCAST_SMS izinleridir. 37 tane iznin katsayısı 0'dan büyük olurken, 63 tane iznin katsayısı ise 0'dan küçük olmaktadır. Tablo 4.2'de verilen öznitelik eleme algoritmasına göre 60 izin elenmiş olup, kat 4'te 42 izin öznitelik olarak değerlendirmeye alınmaktadır.

```
-1.0262, -0.8425, -0.484, -0.4648, -0.4239, -0.4188, -0.2918, -0.2872, -0.2589,
-0.2441, -0.1909, -0.1692, -0.1645, -0.1582, -0.1571, -0.1498, -0.1269, -0.113,
-0.1046, -0.1043, -0.104, -0.1027, -0.102, -0.0862, -0.0841, -0.0786, -0.0598,
-0.057, -0.0569, -0.0562, -0.0542, -0.0531, -0.0478, -0.0472, -0.0466, -0.0465,
-0.0464, -0.0438, -0.0402, -0.0388, -0.0382, -0.0357, -0.0344, -0.0329, -0.0321,
-0.0298, -0.0295, -0.0294, -0.0288, -0.0283, -0.0275, -0.0229, -0.0192, -0.0186,
-0.0181, -0.0165, -0.0156, -0.0138, -0.0134, -0.0133, -0.0129, -0.0115, -0.0109,
0.0, 0.0, 0.0071, 0.0178, 0.0185, 0.0187, 0.0203, 0.0205, 0.0214, 0.0355, 0.0492,
0.0534, 0.0547, 0.055, 0.0591, 0.0621, 0.0632, 0.0659, 0.0779, 0.0822, 0.1001,
0.1003, 0.1173, 0.1214, 0.1429, 0.1545, 0.1609, 0.1637, 0.1808, 0.1905, 0.2184,
0.2876, 0.3159, 0.3341, 0.3405, 0.3557, 0.3761, 0.5236, 0.7785
```

Şekil 4.3. Kat 4'ten elde edilen doğrusal regresyon yöntemindeki izin katsayıları

Bazı izinler sadece bir kat için öznitelik olarak seçilirken bazı izinlerde bütün katlarda öznitelik olarak seçilmektedir. Sadece bir katta öznitelik olarak seçilen 8 tane izin vardır. Buna karşın tüm katlarda öznitelik olarak seçilen 27 tane izin görülmektedir. Resmi geliştirici sayfasına göre bu 27 iznin bazıları tehlikeli izin

grubunda değerlendirilmektedir (Permissions, 2021). Bu izinler Tablo 4.4'te verilmektedir. Önerilen öznitelik seçme yönteminin tehlikeli izin gruplarından bazılarını bütün katlarda bulması dikkate değerdir. Ayrıca Tablo 4.4'te verilen bu izinlerin yanında 9 katta görülen ANSWER_PHONE_CALLS ve android.permission.RECEIVE_SMS gibi tehlikeli izinlerin çoğu durumlarda seçilmesi oldukça önemlidir. Çünkü bu tehlikeli izinler sınıflandırma algoritmalarının kötücül yazılımlar ile iyicil yazılımlar arasında ayrım yapmasını kolaylaştırabilir. Bu sayede sınıflandırma algoritmalarının başarımları da artar. Ayrıca iyicil ve kötücül uygulamalar arasında ne kadar fazla farklılık ortaya çıkartılırsa kötücül yazılım tespit sisteminin gücü de o kadar fazla olacaktır.

Sadece 1 katta seçilen 8 izin, resmi geliştirici sayfasına göre değerlendirildiğinde (Permissions, 2021), yalnızca android.permission.READ_PHONE_NUMBERS izni tehlikeli olarak geçmektedir. Buna karşın diğer izinler normal, ayrıcalıklı veya gelecekte kaldırılması düşünülen izinlerdendir. android.permission.READ_PHONE_NUMBERS gibi tehlikeli bir iznin çoğu foldlarda öznitelik olarak seçilmemesi önerilen öznitelik seçme yönteminin eksisi olarak düşünülebilir. Ama bunun dışında kalan 7 iznin tehlikeli izinler arasında olmaması önerilen yöntemin artısıdır.

Tablo 4.4. Seçilen bazı tehlikeli izinler

İzinler	**Koruma Seviyesi**
android.permission.ACCESS_COARSE_LOCATION	tehlikeli
android.permission.CALL_PHONE	tehlikeli
android.permission.CAMERA	tehlikeli
android.permission.PROCESS_OUTGOING_CALLS	tehlikeli
android.permission.READ_CALL_LOG	tehlikeli
android.permission.READ_CONTACTS	tehlikeli
android.permission.READ_EXTERNAL_STORAGE	tehlikeli
android.permission.READ_PHONE_STATE	tehlikeli
android.permission.RECORD_AUDIO	tehlikeli
android.permission.SEND_SMS	tehlikeli
android.permission.WRITE_EXTERNAL_STORAGE	tehlikeli

Detaylı sınıflandırma sonuçlarının elde edilebilmesi için her bir kat üzerinde farklı sınıflandırma algoritmaları işletilerek herbir sınıflandırma algoritmasının başarımları ayrı ayrı kaydedilmektedir. Ardından her bir sınıflandırma algoritmasına ait sonuçların ortalaması alınarak raporlama işlemi gerçekleştirilmektedir. Örneğin,

kat 4'e ait test kısmının RF algoritması ile sınıflandırmasından oluşan karmaşıklık matrisi Tablo 4.5'te verilmektedir. Elde edilen sonuçlar 102 izin kullanılarak gerçekleştirilmektedir. Buna karşın, 42 izinli kat 4'e ait test kısmının RF algoritması ile sınıflandırmasından oluşan karmaşıklık matrisi Tablo 4.6'da verilmektedir.

Tablo 4.5'e göre, gerçekte iyicil olan 100 uygulamanın tamamı sınıflandırma sonucunda iyicil olarak sınıflandırılmaktadır. Buna karşın gerçekte kötücül olan 100 uygulamanın 96 tanesi kötücül olarak sınıflandırılırken, 4 tanesi ise iyicil olarak sınıflandırılmaktadır. Bu bilgilere göre kesinlik değeri $\frac{100}{100+4} = 0.9615$, duyarlılık değeri ise $\frac{100}{100+0} = 1$ olarak hesap edilmektedir. Kesinlik ve duyarlılık değerlerinin harmonik ortalaması F-ölçütü değeri olup, bu değer $\frac{2*0.9615*1}{0.9615+1} \cong 0.98$ olarak hesap edilmektedir. Bu hesap iyicil sınıfı için yapılmaktadır. Ayrıca karmaşıklık matrisi tersine çevrilip kötücül uygulamalar içinde aynı adımlar tekrarlanarak bir başka F-ölçütü sonucu daha ortaya çıkmaktadır. Elde edilen bu iki değerin aritmetik ortalaması alınarak ilgili kat üzerinde elde edilen başarım ortaya çıkmaktadır.

Tablo 4.5. Test setinin RF ile sınıflandırılmasından elde edilen sonuçlar (Kat 4, 102 izin)

		Tahmini Sınıf	
		İyicil	**Kötücül**
Gerçek Sınıf	**İyicil**	100	0
	Kötücül	4	96

Tablo 4.6'ya göre, gerçekte iyicil olan 100 uygulamanın tamamı sınıflandırma sonucunda iyicil olarak sınıflandırılmaktadır. Buna karşın gerçekte kötücül olan 100 uygulamanın 97 tanesi kötücül olarak sınıflandırılırken, 3 tanesi ise iyicil olarak sınıflandırılmaktadır. Bu bilgilere göre kesinlik değeri $\frac{100}{100+3} = 0.9708$, duyarlılık değeri ise $\frac{100}{100+0} = 1$ olarak hesap edilmektedir. Kesinlik ve duyarlılık değerlerinin harmonik ortalaması F-ölçütü değeri olup, bu değer $\frac{2*0.9708*1}{0.9708+1} \cong 0.985$ olarak hesap edilmektedir. Tablo 4.5'te yapılan hesaplamada değinildiği gibi karmaşıklık matrisi tersine çevrilerek bir başka F-ölçütü değeri hesaplanmaktadır. Ardından bu iki değerin ortalaması alınarak ilgili katın nihai sınıflandırma başarımı elde edilmektedir.

Tablo 4.6. Test setinin RF ile sınıflandırılmasından elde edilen sonuçlar (Kat 4, 42 izin)

		Tahmini Sınıf	
		İyicil	**Kötücül**
Gerçek Sınıf	**İyicil**	100	0
	Kötücül	3	97

Elde edilen öznitelikler değerlendirildikten sonra bu özniteliklerin çeşitli makine öğrenmesi algoritmalarıyla kullanılması ile elde edilen sınıflandırma başarımları verilecektir. Tablo 4.7'de öznitelik seçimi yapılmadan elde edilen ortalama başarımlar ile önerilen öznitelik seçme işleminin uygulandığı ortalama başarımlar karşılaştırılmaktadır.

Tablo 4.7. Öznitelik seçmenin sınıflandırma başarımına etkisi

	Öznitelik seçimi olmadan (102 izin)	**Öznitelik seçimi uygulanarak (42-57 izin)**
Sınıflandırma Algoritması	**F-ölçütü**	**F-ölçütü**
NB	0.9305	0.9226
LR	0.954	0.9565
SMO	0.9655	0.9625
MLP	0.963	0.9605
KNN	0.9485	0.9515
C4.5	0.956	0.958
RF	0.9625	0.9645

Tablo 4.7'ye göre, herhangi bir öznitelik seçme işleminin uygulanmadığı durumda elde edilen en yüksek başarım, 0.9655 olarak SMO algoritması ile elde edilmektedir. Bu algoritmaya en yakın sonuç MLP ve RF algoritmalarının sonuçlarıdır. Bu sonuçlar sırasıyla 0.963 ve 0.9625'tir. Tüm sınıflandırıcılar arasında en kötü sonuç NB algoritmasından elde edilmektedir. Öznitelik seçimi yapıldığında ve öznitelik seçimi yapılmadığında NB algoritması diğer algoritmalara göre daha kötü sonuç vermektedir. Öznitelik seçimi yapıldığında en yüksek başarım RF algoritmasından elde edilmektedir. Bu sonuç 0.9645'tir.

Özellik seçiminin sınıflandırma algoritmaları üzerinde etkisi incelenecek olursa RF, C4.5, KNN ve LR algoritmalarının başarımları artarken, MLP, NB ve SMO algoritmalarının başarımları azalmaktadır. Özellik seçimi ile boyut düşürüldüğü için öğrenme sürecinin kısaltılması sağlanır. Öğrenme süresinin kısaltılmasının yanında, orijinal veri kümesinden elde edilen başarımı çok düşürmeyecek şekilde bir öznitelik seçim yöntemi gerekmektedir. Ayrıca anlamsız ve gereksiz öznitelikler gibi başarımı olumsuz etkileyen özelliklerin silinmesi ile sonuç modelinin doğruluğunun artması amaçlanmaktadır. Bu sayede, oluşturulan modelin verimliliği, uygulanabilirliği ve anlaşılabilirliği sağlanacaktır. Bu durumlar göz önünde bulundurulursa önerilen öznitelik seçme yöntemi ile sınıflandırma başarımını olumsuz etkileyen özniteliklerin

kaldırıldığı, birçok sınıflandırıcı tarafından görülmektedir. Bu duruma RF, C4.5, KNN ve LR algoritmaları örnek olarak verilebilir. Çünkü bu algoritmaların başarımları öznitelik seçme yöntemi kullanılmasıyla artmaktadır. Özellik seçiminin uygulanmasıyla MLP, NB ve SMO algoritmalarının başarımlarında azda olsa düşmeler görülmektedir. Özellikle MLP ve SMO gibi hesaplama maliyeti yüksek olan sınıflandırma algoritmalarında eğitim sürecinin oldukça kısaltılması ve başarımın orijinal veri kümesine göre çok düşmemesi önerilen yöntem için dikkate değerdir.

Tablo 4.7'de yer alan sonuçlara ek olarak çalışmanın sonuçlarını genişletmek için bazı deneyler yapılmaktadır. Bu deneylerden ilki kat 4'te seçilen 42 iznin bütün katlara uygulanmasıdır. 42 iznin bütün katlara uygulanmasıyla elde edilen ortalama sonuçlar Tablo 4.8'de verilmektedir. İkinci deney ise 8 ve üzeri katlarda seçilen 43 iznin bütün katlara uygulanmasıdır. Üçüncü ve dördüncü deneyler ise sırasıyla 9 ve üzeri katlarda seçilen 35 iznin ve sadece 10 katta seçilen 27 iznin bütün katlara uygulanmasıdır. Bu izin gruplarının makine öğrenmesi algoritmalarına verilmesi ile elde edilen ortalama sınıflandırma başarımları Tablo 4.9'da verilmektedir.

Tablo 4.8. Kat 4'te elde edilen 42 iznin tüm katlara uygulanması

	Öznitelik seçimi uygulanarak (42 izin)
Sınıflandırma Algoritması	**F-ölçütü**
NB	0.928
LR	0.958
SMO	0.963
MLP	0.959
KNN	0.954
C4.5	0.956
RF	0.962

Tablo 4.8 incelendiğinde, 42 iznin kullanımıyla elde edilen sonuçlar Tablo 4.7'de verilen sonuçlara oldukça benzer görünmektedir. Tablo 4.8'deki sonuçlar Tablo 4.7'de verilen öznitelik seçimi yapılmadan elde edilen sonuçlar ile kıyaslandığında, LR ve KNN algoritmalarının başarımları artmaktadır. Buna karşın NB, SMO, MLP ve RF algoritmaların başarımları azalmaktadır. C4.5 algoritmasının başarımında ise değişiklik gözlenmemektedir. Bu sonuçlar önerilen öznitelik seçme yönteminin genele uygulanabileceğini göstermektedir.

Tablo 4.9'daki deneylerin yapılma sebebi, en çok seçilen izinlerin bir araya gelmesiyle meydana gelen özellik vektörünün sınıflandırma performansına etkisini görebilmektir. 27 izinden oluşan özellik vektörü ile sınıflandırma yapıldığında, en yüksek başarım MLP algoritmasından elde edilmektedir. Bu sonuç 0.961'dir. Bu sonuca en yakın olan 0.96 değerine SMO ve RF algoritmaları kullanılarak ulaşılmaktadır. Sadece 9 katta görülen 8 iznin 27 izne eklenmesiyle oluşturulan özellik vektörü ile sınıflandırma yapıldığında, bu 8 izin NB, KNN ve C4.5 algoritmalarının başarımını etkilememektedir. Bu algoritmaların aksine, bu 8 izin SMO ve LR algoritmalarının başarımını arttırırken, MLP ve RF algoritmalarının başarımını azaltmaktadır. Hem sadece 9 katta görülen 8 iznin hem de 8 katta görülen 8 iznin 27 izne eklenmesiyle oluşturulan özellik vektörü ile sınıflandırma yapıldığında, bu 16 izin RF ve C4.5 algoritmalarının başarımını etkilememektedir. Bu algoritmaların aksine, bu 16 izin NB, KNN, SMO ve LR algoritmalarının başarımını arttırırken, sadece MLP algoritmasının başarımını azaltmaktadır. Tablo 4.9'da oluşturulan üç farklı modelin sonuçları Tablo 4.7'deki öznitelik seçimi yapılmadan elde edilen sonuçlar ile kıyaslandığında, çoğunluk oylamasına dayalı bu üç modelin oldukça başarılı sonuçlar verdiği dikkate değerdir.

Tablo 4.9. Tüm katlar için bazı izin gruplarının kullanımı

	8 ve üzeri katlarda görülen izinler (43 izin)	**9 ve üzeri katlarda görülen izinler (35 izin)**	**Tüm katlarda görülen izinler (27 izin)**
Sınıflandırma Algoritması	**F-ölçütü**	**F-ölçütü**	**F-ölçütü**
NB	0.928	0.923	0.923
LR	0.96	0.961	0.958
SMO	0.964	0.961	0.96
MLP	0.957	0.959	0.961
KNN	0.954	0.953	0.953
C4.5	0.956	0.956	0.956
RF	0.96	0.959	0.96

4.8. Var Olan Çalışmalar ile Karşılaştırma

Bu bölümde önerilen Android kötücül yazılım tespit sistemi ile var olan bazı çalışmaların karşılaştırması yapılacaktır. Tablo 4.10'da Android kötücül yazılım tespitinde öznitelik seçimi yapan bazı çalışmalar ile önerilen çalışmanın karşılaştırılması yer almaktadır. En az sayıda kullanılan izinle en yüksek elde edilen başarı 0.961 olduğu için bu değer ile var olan çalışmaların kıyaslaması yapılmaktadır.

Doğrusal regresyon tabanlı öznitelik seçme yöntemi 13 çalışma ile kıyaslandığında, önerilen yöntemin 9 çalışmanın sonucundan daha iyi olduğu görülmektedir. (Bhattacharya vd, 2019) çalışmasında iki farklı veri kümesi denenmektedir. Bu sonuçlardan bir tanesi bu bölümde yapılan deney sonuçlarından iyiyken, diğer sonuç ise iyi değildir. Önerilen yöntemin (Fereidooni vd, 2016; Salah vd, 2020) çalışmalarına göre daha kötü sonuç vermesinin sebebi daha az sayıda statik özellik kullanıyor olmasından kaynaklanabilmektedir. Çünkü (Fereidooni vd, 2016; Salah vd, 2020) çalışmalarında uygulama izinleri dışında birçok özellikten faydalanılmaktadır.

Tablo 4.10. Önceki çalışmalar ile karşılaştırma

Çalışma	**Analiz**	**Veri kümesi Boyutu**	**Algoritma**	**Öznitelik Seçme Yöntemi**	**Performans**
(Sanz vd, 2013)	S	1811 İyicil 249 Kötücül	RF	İzin Karşılaştırması	0.8641 (Doğruluk)
(Suleiman Y Yerima vd, 2014; Suleiman Y Yerima vd, 2013)	S	1000 İyicil 1000 Kötücül	Bayesian	Karşılıklı Bilgi	0.92-0.94 (Doğruluk)
(Pehlivan vd, 2014)	S	2338 İyicil 1446 Kötücül	RF	Cfs Subset	0.949 (Doğruluk)
(Fereidooni vd, 2016)	S	11187 İyicil 18677 Kötücül	XGboost	Rastgele Karar Ağacı	0.973 (Gerçek Pozitif Oranı)
(Altaher, 2016)	S	100 İyicil 100 Kötücül	RF	Bilgi Kazancı	0.89 (Gerçek Pozitif Oranı)
(Abdullah vd, 2017)	S	850 İyicil 1505 Kötücül	RF	Bilgi Kazancı	0.946 (Gerçek Pozitif Oranı)
(Fatima vd, 2019)	S	20000 İyicil 20000 Kötücül	SVM	Genetik Algoritma	0.95 (Doğruluk)
(Yildiz ve Doğru, 2019)	S	621 İyicil 1119 Kötücül	SVM	Genetik Algoritma	0.981 (F-ölçütü)
(Salah vd, 2020)	S	123453 İyicil 5560 Kötücül	SVM	FF-FA temelli TFIDF	0.99 (Doğruluk)
(Alazab vd, 2020)	S	13719 İyicil 14172 Kötücül	RF	Bilgi Kazancı ve Terim Frekansı	0.943 (F-ölçütü)
(Bhattacharya vd, 2019)	S	504 İyicil 213 Kötücül	–	Kaba Küme Teorisi ve PSO	0.9118 (F-ölçütü)
(Bhattacharya vd, 2019)	S	2500 İyicil 1150 Kötücül	–	Kaba Küme Teorisi ve PSO	0.9785 (F-ölçütü)
(Bai vd, 2020)	S	5666 İyicil 5560 Kötücül	CatBoost	Filtreleme Tabanlı Hızlı Korelasyon	0.9529 (Doğruluk)
Önerilen yöntem	S	1000 İyicil 1000 Kötücül	MLP	Doğrusal Regresyon	0.961 (F-ölçütü)

S: Statik

(Yildiz ve Doğru, 2019) çalışmasında öznitelik seçme yöntemi olarak genetik algoritma kullanılmaktadır. Genetik algoritma ile çok sayıda öznitelik varyasyonları oluşturulup sınıflandırma algoritmalarının en yüksek performansa ulaşmasını sağlayan en iyi varyasyonun bulunması gerçekleştirilir. Hesaplama maliyeti oldukça fazla olan bir algoritmadır. Her ne kadar (Yildiz ve Doğru, 2019) çalışması doğrusal regresyon tabanlı öznitelik seçme yöntemine göre başarılı olsa da gerçek zamanlı kötücül yazılım tespitinde kullanılabilirliği zor görünmektedir. Buna karşın önerilen sistemin hesaplama maliyeti genetik algoritmaya göre düşük olduğundan gerçek zamanlı kötücül yazılım sistemlerine verimli bir şekilde uyarlanabilir.

Tablo 4.10 dikkate alındığında, araştırmacılar genellikle dengeli olmayan veri kümeleri ile deneyleri gerçekleştirmektedirler. Karşılaştırılan 10 çalışmadan yalnızca 3 tanesinde dengeli veri kümesi tercih edilmektedir. Önerilen doğrusal regresyon tabanlı öznitelik seçme yöntemi de dengeli veri kümesi üzerinde çalıştırılmaktadır. Dengeli veri kümesi kullanan tüm çalışmalar arasında en iyi sınıflandırma sonucu bu çalışmada elde edilen 0.961'dir. (Suleiman Y Yerima vd, 2014; Suleiman Y Yerima vd, 2013) çalışmalarında, bu çalışmada olduğu gibi 1000 iyicil ve 1000 kötücül yazılımdan oluşan veri kümesi kullanılmaktadır. Bu nedenle bu çalışmanın sonuçları (Suleiman Y Yerima vd, 2014; Suleiman Y Yerima vd, 2013) çalışmaları ile tam olarak kıyaslanabilir. (Suleiman Y Yerima vd, 2014; Suleiman Y Yerima vd, 2013) çalışmalarında en etkili öznitelik sayısının 15 ile 20 arasında olduğu belirtilmektedir. 15 ile 20 arasında öznitelik kullanıldığında elde edilen başarım 0.92 ile 0.94 aralığındadır. Bu çalışmada ise 27 öznitelik kullanılarak 0.961 başarım elde edilmektedir. Bu sonuçlar göz önünde bulundurulduğunda, önerilen doğrusal regresyon tabanlı öznitelik seçme yönteminin Android kötücül yazılım tespitinde etkin ve başarılı sonuçlar verdiği görülmektedir.

4.9. Yöntemlerin Karşılaştırılması

Bu bölümde, ele alınan öznitelik seçme yöntemleri ile istatistiksel boyut indirgeme yöntemlerinin karşılaştırılması yapılacaktır. Bölüm 3'te kullanılan veri kümesi ele alınarak yöntemlerin karşılaştırılmasına değinilecektir. Tez kapsamında yapılan bir başka çalışmada, PCA ve LDA teknikleri ile çeşitli veri kümeleri üzerinde boyut indirgemeler yapılarak boyutu indirgenmiş veri kümelerinin sınıflandırma başarımları kıyaslanmıştır (Durmuş Özkan Şahin vd, 2021c). Bu alt bölümde ise aynı

veri kümesi altında filtreleme tabanlı öznitelik seçme yöntemleri, doğrusal regresyona dayalı öznitelik seçme yöntemi, PCA ve LDA tekniklerinin kıyaslanması yapılacaktır.

Bölüm 3'te de kullanılan veri kümesinde 3000 iyicil ve 3000 kötücül uygulama yer almaktadır. Bu uygulamalardan 470 resmi ve özel izin çıkartılmaktadır. Tablo 4.11, herhangi bir boyut indirgeme olmadan bu veri kümesinden elde edilen sonuçları göstermektedir. Tablo 4.11'e göre en başarılı sınıflandırma algoritması RF iken en kötü sınıflandırma algoritması NB'dir. En yüksek başarım F-ölçütü metriğine göre 0.958 olurken, en kötü başarım ise F-ölçütü metriğine göre 0.886 olmaktadır. SMO algoritmasına 470 özellik verildiğinde, sınıflandırma performansı F-ölçütü metriğine göre 0.955'tir. SMO algoritmasından sonra en iyi sonuç KNN ile elde edilmektedir. Bu sonuç F-ölçütü metriğine göre 0.948'dir. KNN algoritmasında k değeri 5 seçilerek deneyler gerçekleştirilmektedir.

Tablo 4.11. Herhangi bir boyut indirgeme olmadan elde edilen sonuçlar

	470 izin
Sınıflandırma Algoritması	**F-ölçütü**
NB	0.886
LR	0.944
SMO	0.955
MLP	0.912
KNN	0.948
C4.5	0.946
RF	0.958

Bu veri kümesi üzerine PCA uygulandığında, PCA'dan elde edilen 191 bileşen varyansın %90'ını, PCA'dan elde edilen 141 bileşen ise varyansın %80'ini içermektedir. Tablo 4.12 ve Tablo 4.13, PCA ile boyutu indirgenmiş veri kümelerinden elde edilen sonuçları göstermektedir. PCA ile boyut indirgemesi yapıldığında, NB algoritması dışında tüm algoritmalarda başarılı sonuçlar elde edilmektedir. 141 bileşen kullanıldığında, sınıflandırma performansı RF algoritması için F-ölçütü metriğine göre 0.967 olmaktadır. RF algoritmasından sonra en iyi sonuç LR ve MLP'den elde edilmektedir. RF, LR ve MLP sonuçları Tablo 4.12'de yer alan en yüksek başarımlardır. 191 bileşen kullanıldığında ise, sınıflandırma performansı RF ve LR için F-ölçütü metriğine göre 0.967 olmaktadır. Bu sonuçlar Tablo 4.13'te yer alan en yüksek başarımlardır.

Tablo 4.12. 141 bileşenden elde edilen sonuçlar

	141 bileşen
Sınıflandırma Algoritması	**F-ölçütü**
PCA + NB	0.793
PCA + LR	0.966
PCA + SMO	0.945
PCA + MLP	0.963
PCA + KNN	0.945
PCA + C4.5	0.941
PCA + RF	0.967

Tablo 4.13. 191 bileşenden elde edilen sonuçlar

	191 bileşen
Sınıflandırma Algoritması	**F-ölçütü**
PCA + NB	0.595
PCA + LR	0.967
PCA + SMO	0.953
PCA + MLP	0.963
PCA + KNN	0.94
PCA + C4.5	0.94
PCA + RF	0.966

Tablo 4.14, LDA ile boyutu indirgenmiş veri kümelerinden elde edilen sonuçları göstermektedir. Tablo IV'te tüm sınıflandırma algoritmaları %98'in üzerinde başarım göstermektedir. LDA ile boyut küçültme sonrası elde edilen sonuçlar Tablo 4.11 ile karşılaştırıldığında tüm algoritmaların sınıflandırma performansları artmaktadır. Özellikle NB ve MLP algoritmalarının performansı önemli ölçüde artmaktadır. Herhangi bir boyut indirgeme kullanılmadan NB algoritması ile sınıflandırma yapıldığında 0.886 başarım elde edilmektedir. NB, LDA ile boyutu indirgenmiş veriye uygulandığında performans 0.989'a yükselmektedir. Benzer bir durum MLP'de görülmektedir. MLP algoritması, tüm öznitelikler kullanıldığında 0.912 sonuç vermektedir. LDA ile MLP algoritması kullanıldığında sınıflandırma performansı 0.988'e çıkmaktadır. Diğer algoritmalar göz önünde bulundurulduğunda da sınıflandırma performanslarının arttığı görülmektedir.

Tablo 4.14. LDA ile boyut küçültme sonrası elde edilen sonuçlar

Sınıflandırma Algoritması	**F-ölçütü**
LDA + NB	0.989
LDA + LR	0.989
LDA + SMO	0.989
LDA + MLP	0.988
LDA + KNN	0.988
LDA + C4.5	0.988
LDA + RF	0.983

Tablo 4.15'te ise doğrusal regresyona dayalı öznitelik seçme yönteminden elde edilen sonuçlar yer almaktadır. Doğrusal regresyona dayalı öznitelik seçme yöntemi ile öznitelik sayısı 470'ten 223'e indirilmektedir. 223 izin kullanılarak sınıflandırma yapıldığında NB, LR, SMO ve MLP algoritmalarının başarımları Tablo 4.11'de yer alan sonuçlar ile kıyaslandığında artmaktadır. KNN, C4.5 ve RF algoritmalarının başarımları ise değişmemektedir.

Tablo 4.15. Doğrusal regresyona dayalı öznitelik seçme yönteminden elde edilen sonuçlar

	223 izin
Sınıflandırma Algoritması	**F-ölçütü**
NB	0.894
LR	0.950
SMO	0.956
MLP	0.929
KNN	0.948
C4.5	0.946
RF	0.958

Tüm yöntemlerden elde edilen sonuçlar göz önünde bulundurularak kıyaslama yapılacak olursa LDA yöntemi diğer yöntemlere göre daha iyi sonuçlar vermektedir. LDA yöntemindeki temel problem sınıf etiketi kullanılarak boyut indirgeme yapılmaktadır. Ayrıca izinlerden oluşan öznitelik vektörlerine dönüşüm uygulanarak farklı yapıya dönüşmektedir. Bu nedenle gerçek zamanlı sistemlerde LDA tekniğini doğrudan kullanabilmek oldukça güçtür. Filtreleme tabanlı öznitelik seçme yöntemlerinde ve doğrusal regresyona dayalı öznitelik seçme yönteminde de sınıf etiketi kullanılarak öznitelik seçimi gerçekleştirilmektedir. Ancak bu yöntemlerde ayırt edici gücü en iyi olan izinlerin seçimi gerçekleştirildiğinden herhangi bir dönüşüm söz konusu değildir. Bu nedenle gerçek zamanlı sistemlere doğrudan

uygulayabilmek oldukça kolaydır. Filtreleme tabanlı öznitelik seçme yöntemleri ile doğrusal regresyona dayalı öznitelik seçme yöntemi kıyaslandığında filtreleme tabanlı yöntemlerin daha iyi sonuçlar verdiği gözlemlenmektedir. Örneğin, doğrusal regresyon tekniğine dayalı öznitelik seçme yöntemiyle elde edilen sınıflandırma başarımları filtreleme tabanlı öznitelik seçme yöntemleri kullanıldığında 50-75 izin ile elde edilebilmektedir. PCA yöntemi LDA, filtreleme tabanlı öznitelik seçme yöntemleri ve doğrusal regresyona dayalı öznitelik seçme yöntemine göre oldukça farklı yapıdadır. PCA tekniğinde sınıf etiketine ihtiyaç duyulmadan boyut indirgeme gerçekleştirilmektedir. Bu yönüyle denetimsiz boyut indirgeme tekniği olarak ele alınmaktadır. Denetimsiz bir boyut indirgeme tekniği olmasına karşın NB algoritması dışında iyi sonuçlar elde edilmektedir. Yöntemler genel olarak kıyaslandığında, filtreleme tabanlı öznitelik seçme yöntemleri ile daha az öznitelik kullanılarak daha iyi sınıflandırma başarımlarına ulaşılmaktadır.

5. LINREGDROID: DOĞRUSAL REGRESYON MODELİNE DAYALI SINIFLANDIRICILAR KULLANARAK ANDROİD KÖTÜCÜL YAZILIMLARIN TESPİTİ

Bu bölümde, doğrusal regresyon tekniğine bazı kurallar ekleyerek kötücül yazılımların tespit edilmesi gerçekleştirilmektedir. Bölüm 4'te doğrusal regresyon sonucunda elde edilen regresyon katsayılardan yararlanılarak öznitelik seçme yaklaşımı önerilmişti. Bu bölümde ise doğrusal regresyon ile sınıflandırma yapılmaktadır. Genel olarak doğrusal regresyon yaklaşımı ile sınıf etiketi değilde sınıf değeri tahmini yapılır. Bu nedenle doğrudan sınıflandırma yapabilmek için sınıf değeri sonucuna bazı kurallar ekleyerek sınıflandırma yapılabilmesi bu bölümde sağlanmaktadır. Kötücül yazılımlar ile iyicil yazılımların ayrıştırılmasında doğrusal regresyon tekniği kullanıldığından genel olarak bu izin tabanlı kötücül yazılım tespit mimarisi LINREGDROID olarak adlandırılmaktadır. Ayrıca iyi sonuçlar veren bazı sınıflandırma algoritmaları torbalama tekniğine göre birlikte kullanılmaktadır.

5.1. Bölüm Motivasyonu

(D. Ö Şahin vd, 2020) çalışmasında lineer regresyonun izin tabanlı Android kötücül yazılım tespitinde nasıl sonuç verdiği raporlanmaktadır. Çalışmada, sınıflandırma işlemi yapılmadan regresyon tekniklerinin ürettiği tahmin değerlerinin hata oranları karşılaştırılmaktadır. Doğrusal regresyon tekniği çok katmanlı algılayıcı, destek vektör makinesine dayalı regresyon ve toplamsal regresyon gibi iyi sonuçlar veren yöntemler ile kıyaslandığında daha az hata oranı ile ön plana çıkmaktadır. İyi bilinen tekniklere göre daha az hata ürettiği için lineer regresyona dayalı bir sınıflandırıcı izin tabanlı kötücül yazılım tespit sisteminde nasıl sonuç verir sorusunun araştırılması bu çalışmanın ana motivasyonunu oluşturmaktadır.

Doğrusal regresyon tekniklerini sınıflandırıcıya evirip kullanan birçok çalışma mevcuttur. (Polat, 2015) çalışmasında, doğrusal regresyon tekniğinden elde edilen sınıflandırıcı ile UCI Makine Öğrenimi Deposunda yer alan iris, statlog (heart) ve balance scale veri kümeleri sınıflandırılmaktadır. Doğrusal regresyon tekniği KNN ile kıyaslandığında daha yüksek başarımlar elde edilmektedir. (Khashei vd, 2012) çalışmasında, yapay sinir ağları ve çok katmanlı doğrusal regresyonun birlikte kullanılmasıyla karma bir sınıflandırma algoritması önerilmektedir. Önerilen teknik Fisher iris, Forensic glass, Japanese credit ve Pima Indian gibi farklı problemlerin

olduğu veri kümelerinde test edilmektedir. Doğrusal regresyon yüz tanıma veya sınıflandırma problemlerinde de sıkça kullanılmaktadır (Hao ve Wang, 2012; Naseem vd, 2010; Seal vd, 2015; Tang vd, 2019). Genel olarak doğrusal regresyon modelinin birçok örüntü tanıma ve makine öğrenmesi problemlerinde kullanıldığı görülmektedir. Ancak, makine öğrenmesine dayalı Android kötücül yazılım tespiti bağlamındaki önemli derleme çalışmaları incelendiğinde (Jusoh vd, 2021; Kouliaridis ve Kambourakis, 2021; Senanayake vd, 2021), doğrusal regresyon modeline dayalı kötücül yazılım tespit sistemine rastlanılmamaktadır. Tezin bu bölümünde ise doğrusal regresyon modelinden yararlanılarak kural tabanlı iki farklı sınıflandırma algoritması ile kötücül yazılım tespiti yapılmaktadır. Önerilen sınıflandırma modellerinin iki önemli avantajı bulunmaktadır. Bunlardan birincisi, önerilen modeller KNN ve NB algoritmalarından daha başarılıdır. İkincisi ise, sadece doğrusal regresyon denklemine ihtiyaç duyularak basit bir karar vericinin elde edilebilmesidir. Bu sayede mobil cihazlar üzerinde doğrudan çalışabilen bir sınıflandırıcı kullanılabilecektir. Mobil cihazların kaynak tüketimi ile batarya tüketimi doğrudan ilişkilidir. Başka bir ifadeyle, kaynak tüketimi arttıkça mobil cihazlar daha çok enerji harcamaktadırlar. Bu yüzden, önerilen sınıflandırıcı oldukça basit olduğundan mobil cihazların kaynak tüketimi olumsuz etkilenmeyecektir. Sonuç olarak, önerilen tespit sistemi mobil cihazı zorlamadan çalışacaktır.

5.2. Bölüm Katkısı

Bu bölümün ana katkıları şöyle özetlenebilir:

- Bildiğimiz kadarıyla, bu çalışma Android kötü amaçlı uygulamaları tespit etmek için doğrusal regresyon modeli kullanan Android kötü amaçlı yazılım tespiti alanındaki ilk kapsamlı çalışmadır.
- İzinlere dayalı Android kötü amaçlı yazılım tespiti için genel bir çerçeve önerilmiştir. Bu çerçevenin temelinde çoklu doğrusal regresyona dayalı kural tabanlı sınıflandırma algoritmaları yer almaktadır.
- Doğrusal regresyon sonucunda üretilen denklemler göz önünde bulundurularak iki farklı kural tabanlı sınıflandırıcı oluşturulmaktadır. İlk kuraldan elde edilen kötücül yazılım tespit sistemi LINREGDROID1 ikinci kuraldan elde edilen kötücül yazılım tespit sistemi ise LINREGDROID2 ile adlandırılmaktadır.

- Elde edilen sınıflandırma algoritmaları 10 kat çapraz doğrulama tekniği kullanılarak KNN, NB, SVM, karar ağaçları ve torbalama tekniğinin olduğu karar ağaçları modelleri ile kıyaslanmaktadır. Önerilen sınıflandırıcılar KNN ve NB tekniklerine göre oldukça başarılıdır. SVM, karar ağaçları gibi iyi sonuçlar veren sınıflandırma algoritmaları ile kıyaslandığında ise genel olarak birbirlerine yakın sonuçlar elde edilmektedir.
- En başarılı sınıflandırma algoritmaları çoğunluk oylamasına dayalı torbalama tekniği ile birlikte kullanılarak sınıflandırma algoritmalarının başarımının arttırılması sağlanmaktadır.
- Doğrusal regresyonda, en küçük kareler yöntemine göre denklem ve katsayılar oluşturulmaktadır. En küçük kareler yönteminin yanında katsayılara rastgele değer verildiğinde elde edilen denklemin nasıl sonuç vereceği araştırılmaktadır.
- Deneyler dört farklı veri kümesi üzerinde farklı yapılarda olan sınıflandırma algoritmaları kullanılarak iki farklı değerlendirme metriği ile gerçekleştirilmektedir.

5.3. Önerilen Sınıflandırma Algoritmaları

Bu alt bölümde, doğrusal regresyon tabanlı sınıflandırma algoritmaları ile topluluk öğrenmesine dayalı sınıflandırma algoritmalarına değinilecektir.

5.3.1. Doğrusal Regresyon Tabanlı Sınıflandırıcılar

Bölüm 4'te doğrusal regresyon tekniğine değinilmişti. Bu alt bölümde doğrusal regresyon tekniği biraz daha detaylandırılarak sınıflandırıcı gibi nasıl kullanıldığı anlatılacaktır. Doğrusal regresyon tekniği, tahminleme probleminlerini çözmede sıkça kullanılan bir yöntemdir. Aynı sınıftaki verilerin aynı lineer alt uzaya ait olacağı ve lineer bir denklemle gösterilebileceği teorisine dayanmaktadır (Tang vd, 2019). Veri kümesi ile örnekler arasında bu doğrusal ilişki bulunursa doğrusal regresyon tekniği aracılığıyla başka örneklere ait tahminler yapmak mümkün olmaktadır. Eşitlik 5.1'de basit doğrusal regresyon modeli gösterilmektedir.

$$y = \beta_0 + \beta_1 X + \varepsilon \quad \textbf{(5.1)}$$

Eşitlik 5.1'de y bağımlı, X ise bağımsız değişken olarak adlandırılmaktadır. Burada, doğrunun y eksenini kestiği nokta β_0 iken, β_1 ise regresyon katsayısını

göstermektedir. Son olarak, ε ise elde edilen tahminin hatasını temsil etmektedir. Eşitlik 5.1 sadece X bağımsız değişkenini bulundurduğundan basit doğrusal regresyon olarak tanımlanmaktadır. Denklemi etkileyen birden fazla bağımsız değişken olması durumunda çoklu doğrusal regresyon olarak adlandırılmaktadır. Çoklu regresyon modeli ise Eşitlik 5.2'de verilmektedir. Eşitlik 5.2 göz önünde bulundurulduğunda $X_1, X_2, \dots, X_n$'den oluşan çok sayıda bağımsız değişken yer almaktadır.

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \quad \textbf{(5.2)}$$

Bu çalışmada ele alınan problem düşünüldüğünde, öznitelikler yani izinler bağımsız değişkeni temsil ederken, y ise bir uygulamanın sınıfını temsil etmektedir. Çok sayıda uygulama izni öznitelik olarak kullanıldığından, çoklu doğrusal regresyon modeline ihtiyaç duyulmaktadır. Şekil 5.1'de, bir uygulamanın sınıfı iyicil veya kötücül olarak gösterilmektedir. Ancak doğrusal regresyonda denklem sistemleri çözüleceğinden iyicil yerine 1, kötücül yerine ise 0 değeri kullanılarak işlemler gerçekleştirilmektedir.

```
1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,
0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,
0,0,0,0,1,0,0,0,benign

0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,
1,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,0,
0,0,1,0,1,1,1,0,1,1,0,0,0,1,1,1,0,
0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,1,1,
0,0,1,1,1,1,1,0,malicious
```

Şekil 5.1. Elde edilen işlenmiş veri kümesinin bir parçası

N tane uygulama ve bu uygulamalardan elde edilen M tane izinden $(p_1, p_2, \dots, p_M)$ oluşan bir veri kümesi olduğunu farz edelim. İzinler ile uygulamalar arasında doğrusal ilişki olduğunda Eşitlik 5.3'te gösterildiği gibi denklem sistemi oluşturulabilmektedir.

$$
\begin{aligned}
y'_1 &= \beta_0 + \beta_1 p_{1,1} + \beta_2 p_{1,2} + \cdots + \beta_M p_{1,M} \\
y'_2 &= \beta_0 + \beta_1 p_{2,1} + \beta_2 p_{2,2} + \cdots + \beta_M p_{2,M} \\
y'_3 &= \beta_0 + \beta_1 p_{3,1} + \beta_2 p_{3,2} + \cdots + \beta_M p_{3,M} \\
&\vdots \\
y'_N &= \beta_0 + \beta_1 p_{N,1} + \beta_2 p_{N,2} + \cdots + \beta_M p_{N,M}
\end{aligned}
\qquad \textbf{(5.3)}
$$

Eşitlik 5.3'te $(y'_1, y'_2, \dots, y'_N)$, izinlerin $(p_1, p_2, \dots, p_M)$ doğrusal kombinasyonlarının sonucunu temsil etmektedir. β_i, izinlerin $(y'_1, y'_2, \dots, y'_N)$ üzerindeki etkisini göstermektedir. Eşitlik 5.3'te, her denklem için uygun $\beta_i (1 \leq i \leq M)$ parametresinin bulunması amaçlanmaktadır. Gerçek sınıf değerleri $(y_1, y_2, \dots, y_N)$ ise yaklaşık olarak $(y'_1, y'_2, \dots, y'_N)$ değerlerine eşit olacaktır.

Doğrusal regresyon modelinin kalitesini ölçmek için genellikle ortalama kare hatası kullanılmaktadır. Ortalama kare hatası ne kadar küçük olursa, doğrusal regresyon modelinin üreteceği değer de gerçek değere o kadar yakın olacaktır. Bu yüzden, kaliteli regresyon modeli elde etmek için, modelin ortalama kare hatasını olabildiğince küçük yapmak gerekmektedir. En uygun β_i parametresinin bulunması ile kaliteli regresyon modelleri oluşturulmaktadır. Eşitlik 5.4'te hatanın nasıl hesaplandığı gösterilmektedir.

$$
\begin{aligned}
SSE &= \sum_{j=1}^{N} (y_j - y'_j)^2 \\
&= \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2
\end{aligned}
\qquad \textbf{(5.4)}
$$

Eşitlik 5.4'te elde edilen SSE fonksiyonunu minimize etmek için, bu fonksiyonun $\beta_i (1 \leq i \leq M)$ bilinmeyenlerinin her birine göre kısmi türevleri alınmalıdır. Hatanın minimuma indirilmesi amaçlandığından kısmı türev almanın sonucu sıfıra eşitlenmektedir. Eşitlik 5.5, kısmı türev alma işlemini göstermektedir.

$$\begin{aligned}
\frac{\partial SSE}{\partial \beta_0} &= \frac{\partial \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2}{\partial \beta_0} = 0 \\
\frac{\partial SSE}{\partial \beta_1} &= \frac{\partial \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2}{\partial \beta_1} = 0 \\
&\vdots \\
\frac{\partial SSE}{\partial \beta_M} &= \frac{\partial \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2}{\partial \beta_M} = 0
\end{aligned} \qquad \textbf{(5.5)}$$

Eşitlik 5.5'te $\beta_i (1 \leq i \leq M)$ bilinmeyenlerinin her birine göre kısmi türevler uygulandığında Eşitlik 5.6 elde edilmektedir. Eşitlik 5.6'da gösterilen A matrisi ve Y vektörü veri kümesi üzerinden doğrudan elde edilebilmektedir. A ve Y bilindiğine göre $A^{-1}Y$ işleminin sonucu ile β vektörü bulunmaktadır. Elde edilen β vektörünün her bir elemanı sırasıyla $\beta_i (1 \leq i \leq M)$ bilinmeyenlerine karşılık gelmektedir.

$$\beta_0 N + \beta_1 \sum_{i=1}^{N} p_{i,1} + \beta_2 \sum_{i=1}^{N} p_{i,2} + \cdots + \beta_M \sum_{i=1}^{N} p_{i,M} = \sum_{i=1}^{N} y_i$$

$$\beta_0 \sum_{i=1}^{N} p_{i,1} + \beta_1 \sum_{i=1}^{N} p_{i,1}^2 + \beta_2 \sum_{i=1}^{N} p_{i,1}p_{i,2} + \cdots + \beta_M \sum_{i=1}^{N} p_{i,1}p_{i,M} = \sum_{i=1}^{N} p_{i,1}y_i$$

$$\beta_0 \sum_{i=1}^{N} p_{i,2} + \beta_1 \sum_{i=1}^{N} p_{i,1}p_{i,2} + \beta_2 \sum_{i=1}^{N} p_{i,2}^2 + \cdots + \beta_M \sum_{i=1}^{N} p_{i,2}p_{i,M} = \sum_{i=1}^{N} p_{i,2}y_i$$

$$\vdots$$

$$\beta_0 \sum_{i=1}^{N} p_{i,M} + \beta_1 \sum_{i=1}^{N} p_{i,1}p_{i,M} + \beta_2 \sum_{i=1}^{N} p_{i,2}p_{i,M} + \cdots + \beta_M \sum_{i=1}^{N} p_{i,M}^2 = \sum_{i=1}^{N} p_{i,M}y_i$$

$$\Downarrow \qquad \textbf{(5.6)}$$

$$\underbrace{\begin{bmatrix} N & \sum_{i=1}^{N} p_{i,1} & \sum_{i=1}^{N} p_{i,2} & \cdots & \sum_{i=1}^{N} p_{i,M} \\ \sum_{i=1}^{N} p_{i,1} & \sum_{i=1}^{N} p_{i,1}^2 & \sum_{i=1}^{N} p_{i,1}p_{i,2} & \cdots & \sum_{i=1}^{N} p_{i,1}p_{i,M} \\ \sum_{i=1}^{N} p_{i,2} & \sum_{i=1}^{N} p_{i,1}p_{i,2} & \sum_{i=1}^{N} p_{i,2}^2 & \cdots & \sum_{i=1}^{N} p_{i,2}p_{i,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} p_{i,M} & \sum_{i=1}^{N} p_{i,1}p_{i,M} & \sum_{i=1}^{N} p_{i,2}p_{i,M} & \cdots & \sum_{i=1}^{N} p_{i,M}^2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}}_{\beta} = \underbrace{\begin{bmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} p_{i,1}y_i \\ \sum_{i=1}^{N} p_{i,2}y_i \\ \vdots \\ \sum_{i=1}^{N} p_{i,M}y_i \end{bmatrix}}_{Y}$$

Eşitlik 5.2'de yer alan regresyon katsayılarının hesaplanması sonucunda doğrusal regresyon modeli elde edilecektir. Bu modele Şekil 5.1'de gösterildiği gibi uygulamalardan elde edilen özellik vektörleri verildiğinde özellik vektörüne ait olan uygulamanın sınıf değeri belirlenmektedir. Bu hesaplama sonucunda sınıf etiketi değil de ilgili uygulamaya ait sınıf değeri ortaya çıkmaktadır. Tez kapsamında sınıflandırma problemi ele alındığından, elde edilen sınıf değerlerine Tablo 5.1'de verilen algoritma ile Tablo 5.2'de verilen algoritma ayrı ayrı uygulanarak iki farklı sonuç oluşmaktadır. Bu sonuçlardan ilki LINREGDROID1 iken ikincisi ise LINREGDROID2 olarak adlandırılmaktadır. Hem Tablo 5.1'deki ve hem de Tablo 5.2'deki algoritmalar doğrusal regresyon denkleminin sonucunu basit kurallara göre işleyerek uygulamala-

Tablo 5.1. LINREGDROID1 ile sınıf etiketlerini belirleyen algoritma

Algoritma: LINREGDROID1 ile sınıf etiketlerini belirleme

Girdi: $TestData[\,][\,], \beta[\,]$ → sırasıyla veri kümesini ve regresyon katsayılarını temsil eder.

Çıktı: $SınıfEtiketi$ → test edilen her uygulamanın tahmini sınıf etiketlerini temsil eder

Function $SINIFLANDIR(TestData[][], \beta[])$

$N_1 = uygulama\ sayısı$

$N_2 = izin\ sayısı$

$sonuc[\,] = \emptyset$

For $i = 1\ to\ N_1$ **do**

$toplam = 0$

For $j = 1\ to\ N_2$ **do**

$toplam \leftarrow toplam + TestData[i][j] * \beta[j]$

End For

$sonuc[i] = \beta[0] + toplam$

End For

$SınıfEtiketi[\,] = \emptyset$

For $i = 1\ to\ N_1$ **do**

If $sonuc[i] >= 0.5$ **then**

$SınıfEtiketi[i] \leftarrow 1$

Else

$SınıfEtiketi[i] \leftarrow 0$

End If

End For

Return $SınıfEtiketi[\,]$

End Function

rın sınıflandırılmasını sağlamaktadırlar. Tablo 5.1'de verilen algoritmada, doğrusal regresyon sonucunda elde edilen sınıf değerlerinin 0.5'ten büyük veya eşit olması durumunda sınıf etiketine "1" değeri başka bir ifade ile iyicil etiketi atanmaktadır. Aksi durumlarda ise uygulamanın sınıf etiketine "0" yani kötücül etiketi atanmaktadır. Benzer bir kural ise Tablo 5.2'de verilen algoritmada yer almaktadır. Bu algoritmada

ise doğrusal regresyon sonucunda elde edilen sınıf değerlerinin 0'a mı yoksa 1' e mi daha yakın olduğu tespit edilmektedir. Eğer sınıf değeri 0'a daha yakınsa ilgili uygulamanın etiketine "0" yani kötücül etiketi atanmaktadır. Aksi takdirde uygulama "1" yani iyicil ile etiketlendirilmektedir.

Tablo 5.2. LINREGDROID2 ile sınıf etiketlerini belirleyen algoritma

Algoritma: LINREGDROID2 ile sınıf etiketlerini belirleme

Girdi: $TestData[\,][\,], \beta[\,]$ → sırasıyla veri kümesini ve regresyon katsayılarını temsil eder.

Çıktı: $SınıfEtiketi$ → test edilen her uygulamanın tahmini sınıf etiketlerini temsil eder

Function $SINIFLANDIR(TestData[][], \beta[])$

$N_1 = uygulama\ sayısı$

$N_2 = izin\ sayısı$

$sonuc[\,] = \emptyset$

For $i = 1\ to\ N_1$ **do**

$toplam = 0$

For $j = 1\ to\ N_2$ **do**

$toplam \leftarrow toplam + TestData[i][j] * \beta[j]$

End For

$sonuc[i] = \beta[0] + toplam$

End For

$SınıfEtiketi[\,] = \emptyset$

For $i = 1\ to\ N_1$ **do**

If $abs(0 - sonuc[i]) < abs(1 - sonuc[i])$) **then**

$SınıfEtiketi[i] \leftarrow 1$

Else

$SınıfEtiketi[i] \leftarrow 0$

End If

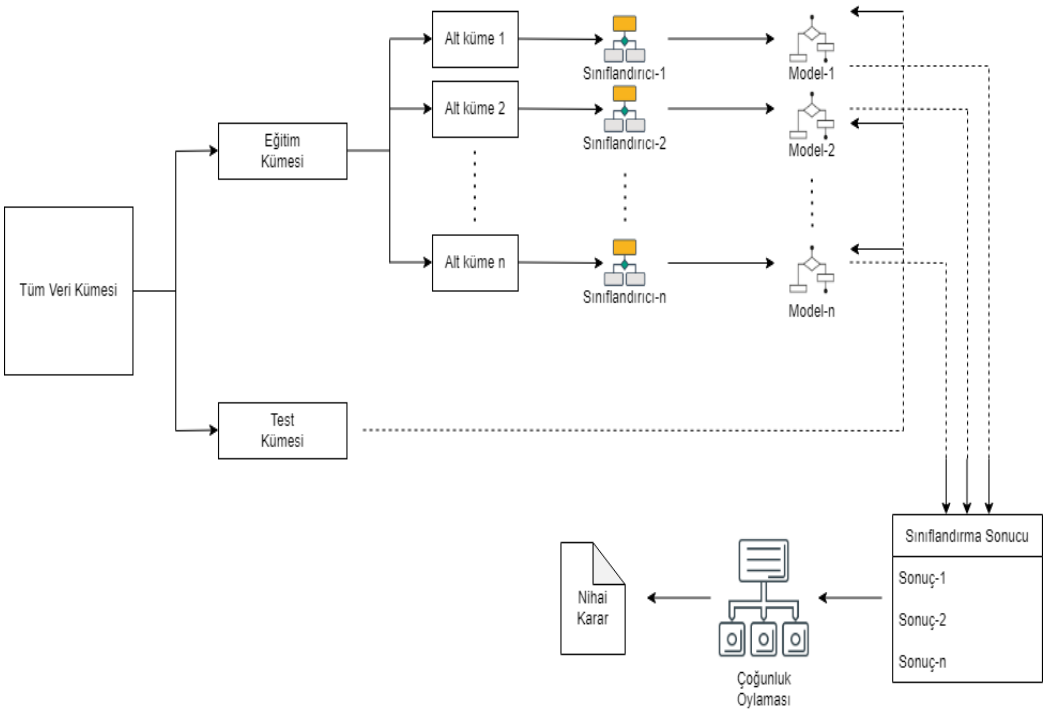
End For

Return $SınıfEtiketi[\,]$

End Function

5.3.2. En İyi Sınıflandırma Algoritmalarının Torbalanması

Topluluk öğrenmesine dayalı modeller genellikle iki farklı şekilde oluşturulmaktadır. Bunlardan birincisi torbalama yöntemi iken ikincisi ise arttırma yöntemidir. Bu yöntemlerin birbirlerine göre avantaj ve dezavantajları (Dietterich, 2000) tarafından ayrıntılı bir şekilde analiz edilmektedir. Bu çalışmada torbalama teknikleri kullanılarak topluluk öğrenmesine dayalı sınıflandırma modelleri oluşturulmaktadır.



Şekil 5.2. Torbalama modellerinin genel çerçevesi

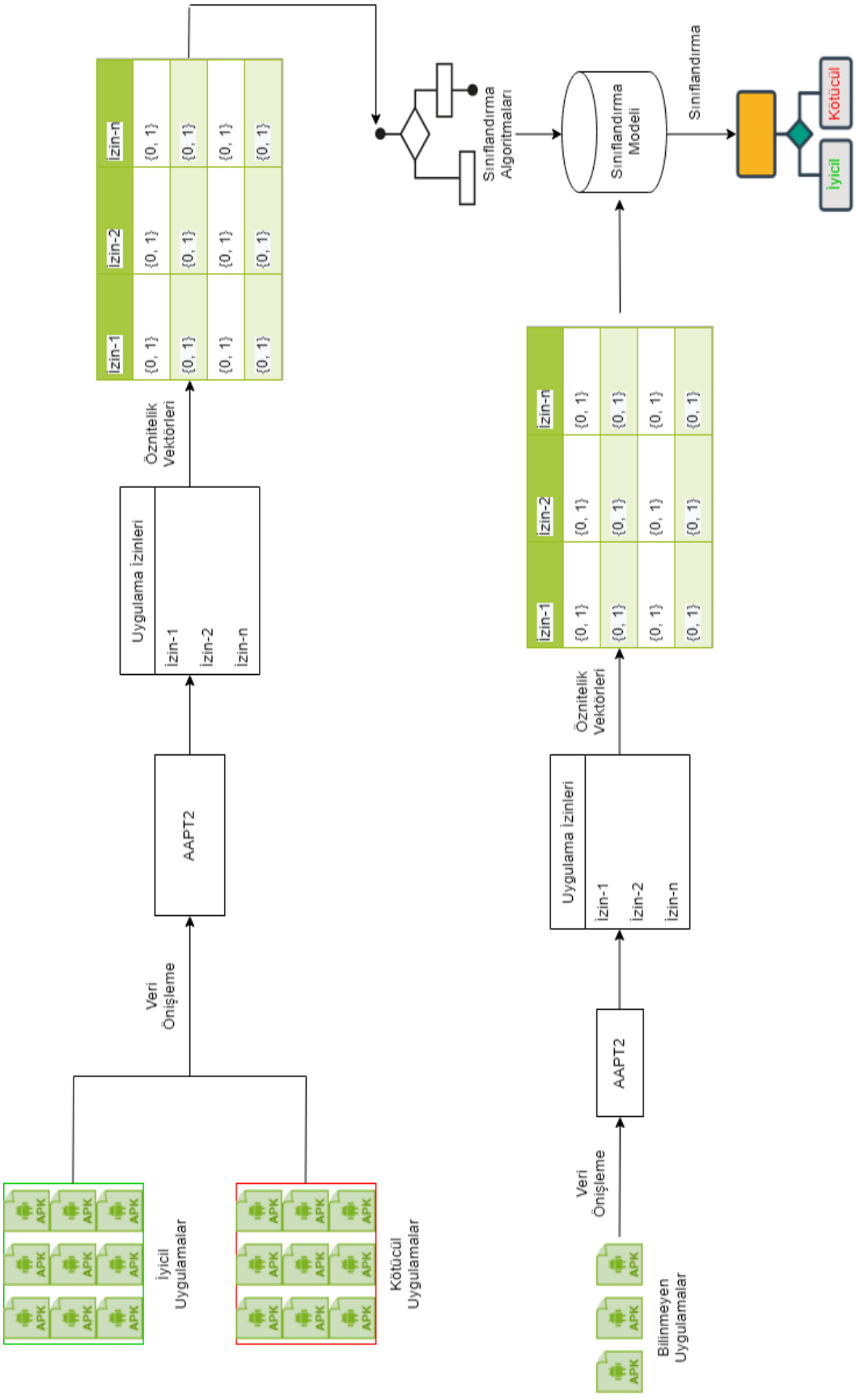
Torbalama tekniğine dayalı modeller genel olarak Şekil 5.2'de gösterildiği gibi oluşturulmaktadır. Şekil 5.2'de görüldüğü gibi eğitim için kullanılan veri kümesinden n sayıda rastgele alt veri kümeleri oluşturulmaktadır. Bu n tane alt veri kümelerinin her biri üzerinde sınıflandırıcılar eğitilirse n adet farklı model ortaya çıkacaktır. En son durumda test kümesinde yer alan bir örnek bu n tane model ile test edildiğinde n adet sınıflandırma sonucu hesaplanmaktadır. Test edilen örneğin sınıfı ise çoğunluk oylamasına göre belirlenmektedir. Örneğin 2 sınıflı ($etiket1$, $etiket2$) bir problem olduğu varsayalım. Test edilen bir örnek k tane model tarafından $etiket1$, l tane model tarafından $etiket2$ olarak sınıflandırılsın ($k + l = n$ olmak üzere). Eğer k değeri l değerinden büyükse test edilen örnek, $etiket1$ olarak sınıflandırılacaktır. Aksi takdirde test edilen örnek $etiket2$ olarak sınıflandırılacaktır. Test verisinde yer alan

bütün örneklere aynı adımlar uygulanarak test verisindeki örneklerin sınıfları tahmin edilmektedir.

Tezin bu bölümünde torbalama tekniğine dayalı iki farlı topluluk öğrenmesi modeli oluşturulmaktadır. Oluşturulan ilk modelde veri kümesinin eğitim kısmı rastgele 5 alt kümeye bölünmektedir. Ardından oluşturulan her bir alt parçaya doğrusal regresyon modeli uygulanmaktadır. Sonuç olarak 5 farklı model meydana gelmektedir. Test aşamasında yer alan her bir uygulama bu modellerden geçirilip çoğunluk oylaması ile uygulamaların türleri tahmin edilmektedir. Bu yöntem Topluluk-1 olarak adlandırılmaktadır. Topluluk-1'in altyapısında Tablo 5.2'de detayı verilen karar verici yer almaktadır. Oluşturulan ikinci topluluk öğrenmesi modeli ise Topluluk-2 olarak adlandırılmaktadır. Burada veri kümesinin eğitim kısmı rastgele 5 alt kümeye bölünmektedir. Ardından oluşturulan parçalardan ikisine doğrusal-SVM, ikisine DT ve birine doğrusal regresyon modeli uygulanmaktadır. İlk olarak, test aşamasında yer alan her bir uygulama bu 5 model ile değerlendirilmektedir. Daha sonra çoğunluk oylaması ile uygulamaların türleri tahmin edilmektedir. Hem Topluluk-1, hem de Topluluk-2 oluşturulurken alt küme sayısının tek olmasına dikkat edilmiştir. Bunun nedeni çoğunluk oylamasında eşitlik durumunun oluşmaması içindir.

5.4. Önerilen İzin Tabanlı Kötücül Yazılım Tespit Sisteminin Altyapısı

Kötücül yazılımların sınıflandırılmasını sağlayan izin tabanlı kötücül yazılım tespit sistemi Şekil 5.3'te verilmektedir. Şekil 5.3'te yer alan işlemler adım adım uygulanarak kötücül yazılımların iyicil yazılımlardan ayrıştırılması sağlanmaktadır. İlk olarak veri kümeleri oluşturulmaktadır. Bu çalışmada 4 farklı veri kümesi kullanılmaktadır. İlk veri kümesi olan M0DROID, (Damshenas vd, 2015) çalışmasının yazarlarından bir tanesi olan Ali Dehghantanha tarafından paylaşılmaktadır. Bu veri kümesinde, 200 tane iyicil, 200 tane kötücül uygulama bulunmaktadır. M0DROID kümesinden 76 doğal izin elde edilmekte ve öznitelik olarak kullanılmaktadır. İkinci veri kümesi ise AMD'dir. Bu veri kümesinde 1000 tane kötücül, 1000 tane iyicil uygulama yer almaktadır. Bu veri kümesinde yer alan kötücül uygulamalar (AMD, 2019; F. Wei vd, 2017) tarafından sağlanılmaktadır. İyicil uygulamalar ise APKPure uygulama deposundan indirilmektedir (APKPure, 2021). Bu veri kümesinden 102 doğal izin çıkartılarak öznitelik olarak kullanılmaktadır. Üçüncü veri kümesi ise (Urcuqui-López ve Cadavid, 2016) çalışmasının yazarlarından olan Lopez tarafından paylaşılmaktadır (Kaggle, 2021). Bu veri kümesinde toplamda 558 uygulama bulun-



Şekil 5.3. Önerilen kötücül yazılım tespit sisteminin mimarisi

maktadır. Bu uygulamaların yarısı iyicil iken geri kalan yarısı ise kötücüldür. Bu veri kümesinde doğal ve özel izinlerden oluşan 330 öznitelik yer almaktadır. Son olarak, dördüncü veri kümesi ise (Arslan, 2021) çalışmasından temin edilmektedir. Bu veri kümesinde toplamda 7622 uygulama yer almaktadır. Bu uygulamalardan 6661 tanesi kötücül uygulama iken 961 tanesi ise iyicil uygulamadır. Bu veri kümesinde doğal ve özel izinlerden oluşan 349 öznitelik yer almaktadır.

Bu bölümde yapılan deneylerde 10 katlı çapraz doğrulama tekniği kullanılmaktadır. İlk olarak veri kümesi 10 parçaya bölünmektedir. Bu parçalardan 9 tanesi eğitim 1 tanesi ise test için kullanılmaktadır. Her bir iterasyonda test için ayrılan parçalar değiştirilerek veri kümesi üzerinde yer alan bütün uygulamaların test edilmesi sağlanmaktadır. Bu işlem 10 kez tekrarlanarak ortalama başarım hesaplanmaktadır.

Veri kümelerinin oluşturulmasının ardından uygulamalar üzerinde ön işlem adımı uygulanarak uygulamalardan izinler çıkartılmaktadır. Bu aşamadan sonra her bir uygulama özellik vektörüne dönüştürülmektedir. Özellik vektörünün elde edilmesi makine öğrenmesi algoritmaları için oldukça önemlidir. Çünkü bu algoritmalara özgü özellik vektörleri girdi olarak verilmezlerse bu algoritmalar hesaplama yapamazlar. Uygun özellik vektörlerinin sınıflandırma algoritmalarına verilmesiyle sınıflandırma modelleri oluşturulmaktadır. Test için ayrılan uygulamalar üzerinde de ön işlem adımları uygulanıp özellik vektörlerine dönüştürülmektedir. Bu vektörlerin sınıflandırma modellerine verilmesiyle uygulamaların türleri tahmin edilmektedir.

Bu çalışmada önerilen doğrusal regresyona dayalı sınıflandırma algoritmalarını karşılaştırmak için temelde 5 farklı sınıflandırma algoritması kullanılmaktadır. Bunlar KNN, NB, SVM, DT ve çok sayıda karar ağaçlarının bir araya getirildiği TreeBagger (Bagging-DT) algoritmalarıdır. Ayrıca, önerilen torbalama tekniğine dayalı algoritma birleştirme yöntemleri içinde de bu algoritmalardan bazıları tercih edilmektedir. Bu algoritmalar için MATLAB R2016'dan faydalanılmaktadır. NB ve SVM algoritmalarında farklı parametreler denenerek bu algoritmalardan elde edilen sonuçlar genişletilmektedir. Çalışmada kullanılan algoritmalar ve onların parametreleri Tablo 5.3'te detaylandırılmaktadır. Tablo 5.3'e göre, DT algoritmasında ön tanımlı parametreler kullanılmaktadır. KNN algoritmasında k değeri 1 seçilerek sınıflandırma gerçekleştirilmektedir. NB algoritmasında iki farklı dağılım kullanılarak sınıflandırma yapılmaktadır. Bunlardan birincisi çok terimli dağılım (multinomial distribution (mn)) iken ikincisi ise çok değişkenli çok terimli (multivariate

multinomial distribution (mvmn)) dağılımdır. SVM algoritmasında ise iki farklı çekirdek fonksiyonu kullanılmaktadır. Bunlar doğrusal ve radyal tabanlı fonksiyonlardır. Son olarak, Bagging-DT algoritması ise toplamda 5 tane ağaç ile gerçekleştirilmektedir. Son olarak, tüm sınıflandırma algoritmalarının başarımları F-ölçütü ve doğruluk metrikleri ile değerlendirilerek karşılaştırılmaktadır.

Tablo 5.3. Kullanılan algoritmalar ve onların parametreleri

Sınıflandırma Algoritmaları	**MATLAB'da Algoritmaların Fonksiyon Adı**	**Algoritma Parametreleri**
KNN	fitcknn	'NumNeighbors', 1
mn-NB	naivebayes.fit	'Distribution', 'mn'
mvmn-NB	naivebayes.fit	'Distribution', 'mvmn'
linear-SVM	fitcsvm	'KernelFunction', 'Linear'
rbf-SVM	fitcsvm	'KernelFunction', 'rbf'
DT	fitctree	Ön tanımlı parametreler
Bagging-DT	TreeBagger	'number_of_trees', 5 'OOBPredictorImportance', 'On'

5.5. Elde Edilen Sonuçlar

Bu bölümde veri kümelerinden elde edilen sonuçlar yorumlanacaktır. Tablo 5.4'te AMD veri kümesinden elde edilen sonuçlar yer almaktadır. Bu sonuçlar 10 katlı çapraz doğrulamanın ortalamasıdır. AMD veri kümesi üzerinde, LINREGDROID1 ve LINREGDROID2 hem doğruluk metriğine göre hem de F-ölçütü metriğine göre 0.9560 başarım vermektedir. KNN algoritması ile elde edilen sonuç doğruluk metriğine göre %93.6 olurken F-ölçütü metriğine göre 0.9359'dur. LINREGDROID1 ve LINREGDROID2 KNN algoritmasına göre %2 oranında iyileştirme sağlamaktadır. mn-NB ve mvmn-NB sınıflandırıcılarından ise F-ölçütü metriğine göre sırasıyla 0.9001 ve 0.9320 başarımları elde edilmektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar NB algoritmasına göre %2 ile %5 oranında daha yüksek başarım göstermektedir. linear-SVM ve rbf-SVM yöntemleri ise F-ölçütü metriğine göre sırasıyla 0.9655 ve 0.9278 başarımlarını vermektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar rbf-SVM modelinden %3 oranında daha başarılıdır. Ancak bu modeller linear-SVM modeli ile kıyaslandığında %1 oranında daha az başarım göstermektedir. LINREGDROID1, LINREGDROID2 ve DT modelleri AMD veri kümesi üzerinde aynı sonuçları göstermektedir. Var olan Bagging-DT ile

Topluluk-1 ve Topluluk-2 modelleri üzerinde adil kıyaslama yapmak için eğitim seti rastgele beş parçaya bölünerek torbalama teknikleri karşılaştırılmaktadır. AMD veri kümesi üzerinde Bagging-DT, Topluluk-1 ve Topluluk-2 neredeyse aynı başarımları göstermektedir. Bütün sonuçlar göz önünde bulundurulduğunda elde edilen en yüksek başarım Topluluk-2 modelinden alınmaktadır. Bu sonuç hem doğruluk metriğine göre hem de F-ölçütü metriğine göre 0.9695'tir.

Tablo 5.4. AMD veri kümesinden elde edilen sonuçlar

Sınıflandırma Algoritması	**Doğruluk (%)**	**F-ölçütü**
LINREGDROID1	95.6	0.956
LINREGDROID2	95.6	0.956
KNN	93.6	0.9359
mvmn-NB	93.2	0.932
mn-NB	90.05	0.9001
rbf-SVM	92.8	0.9278
linear-SVM	96.55	0.9655
DT	95.6	0.956
Bagging-DT	96.5	0.965
Topluluk-1	96.4	0.964
Topluluk-2	96.95	0.9695

Tablo 5.5'te Lopez'in veri kümesinden elde edilen sonuçlar verilmektedir. Bu veri kümesinde, uygulama sayısı göz önüne alındığında oldukça fazla izin bulunmaktadır. 558 uygulamaya karşın 330 izin yer almaktadır. Bu durum genel olarak iyi doğrusal regresyon modelinin oluşturulmasını zorlaştırmaktadır. Bu nedenle sınıflandırılması zor bir veri kümesidir. LINREGDROID1 ve LINREGDROID2 hem doğruluk metriğine göre hem de F-ölçütü metriğine göre Lopez'in veri kümesinde 0.9187 başarım vermektedir. KNN algoritması ile elde edilen sonuç doğruluk metriğine göre %83.75 olurken F-ölçütü metriğine göre 0.8359'dur. LINREGDROID1 ve LINREGDROID2 KNN algoritmasına göre %8 oranında iyileştirme sağlamaktadır. mn-NB ve mvmn-NB sınıflandırıcılarından ise F-ölçütü metriğine göre sırasıyla 0.8553 ve 0.8811 başarımları elde edilmektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar NB algoritmasına göre %3 ile %6 oranında daha yüksek başarım göstermektedir. linear-SVM ve rbf-SVM yöntemleri ise F-ölçütü metriğine göre sırasıyla 0.9375 ve 0.9123 başarımlarını vermektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar rbf-SVM modeli ile benzer sonuçları göstermektedir. Ancak bu modeller linear-SVM modeli ile kıyaslandığında %2 oranında daha az

başarım göstermektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar DT modeli ile kıyaslandığında ise %1 oranında daha az başarım göstermektedir. Bu veri kümesi üzerinde Bagging-DT, Topluluk-1 ve Topluluk-2 torbalama teknikleri ana sınıflandırıcılardan düşük sonuçlar vermektedir. Örneğin, DT modeli ile elde edilen sonuç F-ölçütü metriğine göre 0.925 olurken Bagging-DT ile elde edilen sonuç ise F-ölçütü metriğine göre 0.9150'dir. Benzer durum Topluluk-1 ve Topluluk-2'nin sonuçlarında da görülmektedir. Bütün sonuçlar göz önünde bulundurulduğunda elde edilen en yüksek başarım linear-SVM modelindendir. Bu sonuç hem doğruluk metriğine göre hem de F-ölçütü metriğine göre 0.9375'tir.

Tablo 5.5. Lopez'in veri kümesinden elde edilen sonuçlar

Sınıflandırma Algoritması	**Doğruluk (%)**	**F-ölçütü**
LINREGDROID1	91.87	0.9187
LINREGDROID2	91.87	0.9187
KNN	83.75	0.8359
mvmn-NB	88.12	0.8811
mn-NB	85.62	0.8553
rbf-SVM	91.25	0.9123
linear-SVM	93.75	0.9375
DT	92.5	0.925
Bagging-DT	91.5	0.915
Topluluk-1	91.25	0.9123
Topluluk-2	92.5	0.925

Tablo 5.6'da M0DROID veri kümesinden elde edilen sonuçlar yer almaktadır. M0DROID veri kümesi üzerinde, LINREGDROID1 ve LINREGDROID2 doğruluk metriğine göre %82.942 başarım verirken, F-ölçütü metriğine göre ise 0.8287 başarım vermektedir. KNN algoritması ile elde edilen sonuç doğruluk metriğine göre %82.69 olurken F-ölçütü metriğine göre 0.8258'dir. Hem LINREGDROID1 ve LINREGDROID2 hem de KNN benzer sonuçlar vermektedir. mn-NB ve mvmn-NB sınıflandırıcılarından ise F-ölçütü metriğine göre sırasıyla 0.7733 ve 0.7765 başarımları elde edilmektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar NB algoritmasına göre %5 oranında daha yüksek başarım göstermektedir. linear-SVM ve rbf-SVM yöntemleri ise F-ölçütü metriğine göre sırasıyla 0.8619 ve 0.8673 başarımlarını vermektedir. AMD ve Lopez'in veri kümelerinin aksine rbf çekirdek fonksiyonu bu veri kümesinde daha başarılı sonuç üretmektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar hem rbf-SVM modelinden hem de

linear-SVM modelinden daha düşük sonuç vermektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar DT modeli ile kıyaslandığında ise %4 oranında daha az başarım göstermektedir. Bu veri kümesi üzerinde Bagging-DT, Topluluk-1 ve Topluluk-2 torbalama teknikleri ana sınıflandırıcılardan daha yüksek sonuçlar vermektedir. Örneğin, DT modeli ile elde edilen sonuç F-ölçütü metriğine göre 0.8619 olurken Bagging-DT ile elde edilen sonuç ise f-ölçütü metriğine göre 0.8712'dir. Benzer durum Topluluk-1 ve Topluluk-2'nin sonuçlarında da görülmektedir. LINREGDROID2 modeli ile elde edilen sonuç F-ölçütü metriğine göre 0.8287 olurken Topluluk-1 ile elde edilen sonuç ise F-ölçütü metriğine göre 0.8348'dir. Bütün sonuçlar göz önünde bulundurulduğunda elde edilen en yüksek başarım Topluluk-2 modelinden alınmaktadır. Bu sonuç doğruluk metriğine göre %89.22 iken F-ölçütü metriğine göre 0.8915'tir.

Tablo 5.6. M0DROID veri kümesinden elde edilen sonuçlar

Sınıflandırma Algoritması	**Doğruluk (%)**	**F-ölçütü**
LINREGDROID1	82.942	0.8287
LINREGDROID2	82.942	0.8287
KNN	82.69	0.8258
mvmn-NB	77.923	0.7765
mn-NB	77.429	0.7733
rbf-SVM	86.962	0.8673
linear-SVM	86.212	0.8619
DT	86.212	0.8619
Bagging-DT	87.205	0.8712
Topluluk-1	83.74	0.8348
Topluluk-2	89.22	0.8915

Tablo 5.7'de Arslan'ın veri kümesinden elde edilen sonuçlar yer almaktadır. Diğer veri kümelerinin aksine bu veri kümesi üzerinde doğruluk ve F-ölçütü metriklerinin oldukça farklı çıkmasının sebebi bu veri kümesinin dengeli olmayan bir veri kümesi olmasındandır. Bu veri kümesi üzerinde, LINREGDROID1 ve LINREGDROID2 doğruluk metriğine göre %96.69 başarım verirken, F-ölçütü metriğine göre ise 0.9172 başarım vermektedir. KNN algoritması ile elde edilen sonuç doğruluk metriğine göre %96.54 olurken F-ölçütü metriğine göre 0.9126'dır. mn-NB ve mvmn-NB sınıflandırıcılarından ise F-ölçütü metriğine göre sırasıyla 0.8667 ve 0.8571 başarımları elde edilmektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar NB algoritmasına göre %6 oranında daha yüksek başarım göstermektedir.

linear-SVM ve rbf-SVM yöntemleri ise F-ölçütü metriğine göre sırasıyla 0.9470 ve 0.8617 başarımlarını vermektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar rbf-SVM modelinden %5 oranında daha başarılıdır. Ancak bu modeller linear-SVM modeli ile kıyaslandığında %3 oranında daha az başarım göstermektedir. Önerilen doğrusal regresyon modeline dayalı yaklaşımlar DT modeli ile kıyaslandığında ise %3 oranında daha az başarım göstermektedir. Bu veri kümesi üzerinde Bagging-DT haricinde Topluluk-1 ve Topluluk-2 torbalama teknikleri ana sınıflandırıcılardan daha yüksek sonuçlar vermektedir. Ancak Bagging-DT daha düşük başarım vermektedir. Örneğin, DT modeli ile elde edilen sonuç F-ölçütü metriğine göre 0.9443 olurken Bagging-DT ile elde edilen sonuç ise F-ölçütü metriğine göre 0.9249'dur. Buna karşın LINREGDROID2 modeli ile elde edilen sonuç F-ölçütü metriğine göre 0.9172 olurken Topluluk-1 ile elde edilen sonuç ise F-ölçütü metriğine göre 0.9229'dur. Bütün sonuçlar göz önünde bulundurulduğunda elde edilen en yüksek başarım Topluluk-2 modelinden alınmaktadır. Bu sonuç doğruluk metriğine göre %98.53 iken F-ölçütü metriğine göre 0.9662'dir.

Tablo 5.7. Arslan'ın veri kümesinden elde edilen sonuçlar

Sınıflandırma Algoritması	**Doğruluk (%)**	**F-ölçütü**
LINREGDROID1	96.69	0.9172
LINREGDROID2	96.69	0.9172
KNN	96.54	0.9126
mvmn-NB	93.96	0.8571
mn-NB	94.74	0.8667
rbf-SVM	94.86	0.8617
linear-SVM	97.76	0.9470
DT	97.63	0.9443
Bagging-DT	97.01	0.9249
Topluluk-1	96.91	0.9229
Topluluk-2	98.53	0.9662

Veri kümelerinden elde edilen sonuçlara göre oluşturulan doğrusal regresyon modeline dayalı sınıflandırıcıların genel olarak iyi sonuçlar verdiği görülmektedir. Ayrıca, izin tabanlı kötücül yazılım tespitinde, aynı sınıftaki verilerin aynı lineer alt uzaya ait olacağı ve lineer bir denklemle ifade edilebileceği gösterilmektedir. Veri kümesi ile örnekler arasında doğrusal ilişki bulunduğundan doğrusal regresyon tekniği aracılığıyla başka örneklere ait tahminler yapılabilmesi sağlanmaktadır. Son olarak, elde edilen torbalama tekniklerinin de iyi sonuçlar verdiği göz ardı edilmemelidir.

Torbalama tekniklerinin oluşturulmasında, veri kümeleri nispeten küçük olduğundan veri kümelerinin eğitim kısımları rastgele 5 parçaya bölünerek işlemler yapılmaktadır. Daha büyük veri kümelerinde daha fazla alt kümeler oluşturularak daha yüksek başarımların elde edilebilmesi mümkündür. Ayrıca, bu çalışmada, regresyon katsayılarına rastgele değerler vererek farklı regresyon modelleri oluşturulmaktadır. Rastgele oluşturulan modeller hakkında elde edilen bulgular Yorum-1'de yer almaktadır.

Yorum-1

Bu çalışmada elde edilen regresyon katsayıları genel olarak -1 ile 1 arasında değişmektedir. Regresyon katsayılarına -1 ile 1 arasında rastgele değerler vererek 10000 tane regresyon modeli oluşturulmaktadır. Ancak rastgele modellerin hata oranları gerçek modele göre daha fazladır. Örneğin AMD veri kümesi üzerinde yapılan deneylerde gerçek regresyon modelinin Pearson korelasyon katsayısı 0.8836'dır. Rastgele oluşturulan modellerden en iyisinin sonucu Pearson korelasyon katsayısına göre 0.8694'tür. Bu rastgele modellerden sadece 3429 tanesinin Pearson korelasyon katsayısı 0.80'in üzerindedir. Kaba kuvvet ile arama yerine akıllı arama stratejileri geliştirilerek daha iyi modeller ortaya çıkartılabilir.

5.6. Var Olan Çalışmalar ile Karşılaştırma

Bu alt bölümde, elde edilen sonuçlar literatürdeki bazı sonuçlar ile kıyaslanacaktır. Tablo 5.8'de var olan çalışmaların sonuçları ile bu çalışmada elde edilen sonuçlar kıyaslanmaktadır. Kıyaslama yapılırken sadece statik analiz göz önünde bulundurulmamaktadır aynı zamanda dinamik ve hibrit bazı çalışmalardan elde edilen sonuçlara da yer verilmektedir. Var olan çalışmalarda raporlanan en yüksek başarımlar ve bu başarımların elde edildiği sınıflandırma algoritmaları ile karşılaştırmalar yapılmaktadır. Bu çalışmada, izin tabanlı bir Android kötücül yazılım tespit sistemi önerildiğinden ilk olarak izin tabanlı modeller kendi aralarında değerlendirilecektir. Ardından genel bir kıyaslama yapılacaktır.

Tablo 5.8'e göre sadece izinleri öznitelik olarak kullanan 5 çalışma yer almaktadır. Bu çalışmalardan elde edilen en yüksek başarım F-ölçütü metriğine göre 0.9820 olarak (Arslan, 2021) çalışmasından elde edilmektedir. Aynı veri kümesini kullanarak Topluluk-2 tekniği ile F-ölçütü metriğine göre 0.9662 sonucu bu çalışmada

Tablo 5.8. Önceki çalışmalar ile karşılaştırma

Çalışma	Kullanılan Öznitelikler	Veri Kümesi Boyutu	Sınıflandırma Algoritması	Sınıflandırma Başarımı
(Arslan, 2021)	İzinler	6661 Kötücül 961 İyicil	DNN	0.9820 (F-ölçütü)
(J. Li vd, 2018)	İzinler	5494 Kötücül 5494 İyicil	SVM	95.63 (doğruluk %)
(Milosevic vd, 2017)	İzinler	200 Kötücül 200 İyicil	Topluluk öğrenmesi	0.894 (F-ölçütü)
(Milosevic vd, 2017)	Kaynak kod analizi	200 Kötücül 200 İyicil	Topluluk öğrenmesi	0.9560 (F-ölçütü)
(Luo vd, 2017)	Sistem çağrı fonksiyonları	100 Kötücül 100 İyicil	J48	0.86 (F-ölçütü)
(Kurniawan vd, 2015)	İnternet trafiği Batarya seviyesi Batarya sıcaklığı	200 Kötücül 200 İyicil	RF	85.6 (doğruluk %)
(Tao vd, 2018)	İzinler API çağrıları	15336 Kötücül 31185 İyicil	RF	0.9824 (F-ölçütü)
(Arslan vd, 2019)	İzinler	6660 Kötücül 1074 İyicil	Rastgele Ağaç	92 (doğruluk %)
(Feizollah vd, 2017)	İzinler Niyet filtreleri	5560 Kötücül 1846 İyicil	Bayes Ağları	95.5 (Gerçek Pozitif Oranı %)
(Sayfullina vd, 2015)	4 dosya grubu altında 13 statik özellik	10000 Kötücül 10000 İyicil	Normalleştirilmiş Bernoulli NB	82.10 (Gerçek Pozitif Oranı %)
(Wen, 2013)	İzinler	139 Kötücül 231 İyicil	SVM	89.68 (doğruluk %)
(Nissim vd, 2016)	İzinler Classes.dex dosyası özellikler	10000 Kötücül 30000 İyicil	SVM	98.8 (doğruluk %)
(Ma vd, 2020)	API çağrı dizgeleri	9616 Kötücül 11982 İyicil	Çift Yönlü LSTM	97.22 (doğruluk %)
(Pektaş ve Acarman, 2020)	API çağrı grafikleri	33139 Kötücül 25000 İyicil	CNN	98.86 (doğruluk %)
AMD'den sonuçlar	İzinler	1000 Kötücül 1000 İyicil	LINREGDROID Topluluk-1 Topluluk-2	0.956 (F-ölçütü) 0.964 (F-ölçütü) 0.9695 (F-ölçütü)
Lopez'in veri kümesinden sonuçlar	İzinler	279 Kötücül 279 İyicil	LINREGDROID Topluluk-1 Topluluk-2	0.9187 (F-ölçütü) 0.9123 (F-ölçütü) 0.925 (F-ölçütü)
M0DROID'den sonuçlar	İzinler	200 Kötücül 200 İyicil	LINREGDROID Topluluk-1 Topluluk-2	0.8287 (F-ölçütü) 0.8348 (F-ölçütü) 0.8915 (F-ölçütü)
Arslan'ın veri kümesinden sonuçlar	İzinler	6661 Kötücül 961 İyicil	LINREGDROID Topluluk-1 Topluluk-2	0.9172 (F-ölçütü) 0.9229 (F-ölçütü) 0.9662 (F-ölçütü)

elde edilmektedir. Bizim sonucumuz yaklaşık olarak %2 oranında daha düşüktür. Ancak DNN tekniğinin hesaplama maliyeti oldukça fazladır. Ayrıca çok sayıda parametre olduğundan ağın oluşturulması oldukça karmaşıktır. Bu veri kümesine

benzer dağılım (Arslan vd, 2019) çalışmasında kullanılmaktadır. (Arslan vd, 2019) çalışmasından elde edilen sonuç doğruluk metriğine göre %92'dir. Bu çalışmada ise benzer dağılımda veri kümesi kullanıldığında, doğruluk metriğine göre Topluluk-2 ile %98.53 başarım elde edilmektedir. LINREGDROID ile sınıflandırma yapıldığında ise doğruluk metriğine göre %96.69 başarım elde edilmektedir. (Arslan vd, 2019) çalışmasından elde edilen sonuçlara göre %4 ile %6 arasında iyileştirme yapılmaktadır. (J. Li vd, 2018) tarafından yapılan çalışmada ise doğruluk metriğine göre %95.63 başarım elde edilmektedir. AMD veri kümesini kullanarak bu başarıma benzer sonuçlar bu çalışmadan elde edilmektedir. İzin tabanlı kötücül yazılım tespit sistemlerinin küçük veri kümeleri üzerindeki sonuçları incelendiğinde, (Milosevic vd, 2017) çalışmasında F-ölçütü metriğine göre 0.894 başarım elde edilmektedir. (Wen, 2013) çalışmasında ise doğruluk metriğine göre %89.68 başarım elde edilmektedir. (Milosevic vd, 2017)'de M0DROID veri kümesi kullanılmaktadır. Bu veri kümesini kullanarak F-ölçütü metriğine göre 0.8915 başarım elde edilmektedir. (Milosevic vd, 2017; Wen, 2013) ve bizim çalışmamızda her ne kadar izin tabanlı yaklaşım kullanılsa da sınıflandırma yaklaşımlarında farklı yapılar sunulmaktadır. Ancak bu üç çalışmanın sonuçları birbirlerine oldukça yakın değerdedir. Bu çalışmada kullanılan Lopez'in veri kümesi de küçük boyuttadır. Bu veri kümesi üzerinde elde edilen başarımlar diğer küçük veri kümelerinden elde edilen sonuçlara göre daha iyidir. Bunun nedeni iyi huylu ve kötü amaçlı uygulamaların bu veri setinde daha kolay ayrıştırılabilir olmasından kaynaklanabilir.

Uygulama izinleri ile birlikte API çağrıları veya niyet filtreleri gibi başka öznitelikler kullanıldığında genelde başarımın arttığı görülmektedir (Feizollah vd, 2017; Nissim vd, 2016; Tao vd, 2018). (Sayfullina vd, 2015) çalışmasında 4 farklı dosya değerlendirilerek çok sayıda statik özellik çıkartılmaktadır. Ancak (Sayfullina vd, 2015) çalışmasındaki başarım (Feizollah vd, 2017; Nissim vd, 2016; Tao vd, 2018) çalışmalarından elde edilen sonuçlar kadar yüksek değildir. Bunun sebebi NB algoritmasından kaynaklı olabilir. Çünkü NB algoritması ile elde edilen sonuçlar genelde diğer sınıflandırma algoritmaları kadar yüksek değildir. Küçük veri kümeleri üzerindeki dinamik analiz yaklaşımlarının sonuçları değerlendirildiğinde, (Luo vd, 2017) çalışmasında F-ölçütü metriğine göre 0.86 başarım elde edilmektedir. (Kurniawan vd, 2015) çalışmasında ise doğruluk metriğine göre %85.6 başarım elde edilmektedir. Tablo 6.8 genel olarak değerlendirildiğinde derin öğrenme tekniklerinin

başarımlarının oldukça iyi olduğu gözlemlenmektedir (Arslan, 2021; Ma vd, 2020; Pektaş ve Acarman, 2020). Bu çalışmada yapılan deneylerin sonuçları incelendiğinde, önerilen yöntemlerin literatürdeki sonuçlar kadar başarılı olduğu görülmektedir.

Sonuçlar genel olarak incelendiğinde, araştırmacılar genelde dengeli olmayan veri kümesi üzerinde deneylerini gerçekleştirmektedir. Veri kümesinin dağılımı başarımı etkileyen önemli faktörlerdendir. Bu çalışmada yapılan deneylerde dengeli veri kümesinin kullanılmasına özen gösterilmektedir. Başarımı etkileyen önemli bir faktör de öznitelik çıkarımıdır. İyicil ve kötücül uygulamalar arasında daha ayırt edici öznitelikler keşfedildikçe daha yüksek sınıflandırma başarımları elde edilebilir. Bunun gibi farklılıklardan dolayı elde edilen sonuçlar arasında farklılıklar görülmektedir. Örneğin, (Milosevic vd, 2017) çalışmasında M0DROID veri kümesi kullanılarak deneyler gerçekleştirilmektedir. Benzer şekilde bu çalışmada da M0DROID veri kümesi ile deneyler gerçekleştirilmektedir. M0DROID veri kümesinden izinler çıkartıldığında her iki çalışmadan elde edilen sonuçlar neredeyse aynıdır. Ancak izin yerine uygulama kaynak kodları kullanıldığında daha iyi başarımın elde edildiği gösterilmektedir (Milosevic vd, 2017). Son olarak, veri kümelerinin dağılımları aynı olsa bile kötücül yazılımların özellikleri iyicil yazılımların özelliklerine benzeyebilir. Bu durumda sınıflandırma algoritmalarının başarımında da farklılıklar meydana gelebilir.

6. KNN ALGORİTMASINDAKİ k DEĞERLERİNİN VE MESAFE METRİKLERİNİN SINIFLANDIRMA BAŞARIMINA OLAN ETKİSİNİN İNCELENMESİ

Bu bölümde, KNN algoritmasında kullanılan çeşitli k değerlerinin ve mesafe metriklerinin izin tabanlı kötücül yazılım tespit sisteminin başarımını nasıl etkilediği detaylı bir şekilde incelenecektir.

6.1. Bölüm Motivasyonu

Makine öğrenimi çalışmalarında parametre seçimi oldukça önemli konulardan bir tanesidir. Parametre seçimi sınıflandırma başarımını ciddi bir şekilde etkilemektedir. Son yıllarda makine öğrenimine dayalı Android kötücül yazılım tespitinde dikkate değer bir artış söz konusudur. KNN algoritması da araştırmacıların sıkça tercih ettiği uygulaması kolay ve iyi sınıflandırma performansı veren algoritmalardan bir tanesidir. KNN algoritmasında k değeri ve uzaklık metriği sınıflandırma başarımını doğrudan etkilemektedir. Ancak yapılan çalışmaların çoğunda araştırmacıların çoğunlukla hiper parametre seçimi yapmadan algoritmaların ön tanımlı ayarları ile sınıflandırma yaptıkları vurgulanmaktadır (Baldini ve Geneiatakis, 2019). Literatür incelemesi sonucunda, (Baldini ve Geneiatakis, 2019) çalışması Android kötücül yazılım tespitinde KNN algoritmasının parametre seçimine odaklanan ilk çalışma olarak dikkat çekmektedir. Bu çalışmadan farklı olarak bu bölümde yapılan deneylerde tam anlamıyla dengeli ve farklı boyutlardaki veri kümeleri kullanılmaktadır. Ayrıca öznitelik olarak sadece izinler kullanılarak sonuçlar elde edilmektedir.

KNN algoritmasını kullanarak yapılan Android kötücül yazılım tespit sistemleri ile ilgili karşılaştırma Tablo 6.1'de yapılmaktadır. Bu çalışmalar göz önünde bulundurulduğunda farklı k değerlerinin bazı araştırmacılar tarafından denendiğin görülmektedir. Ancak farklı uzaklık metriklerini kullanan sadece bir çalışma ön plana çıkmaktadır. Tablo 6.1'de yer alan izin tabanlı çalışmalardan yalnızca 3 tanesinde k parametresine farklı değerler verilerek sınıflandırma gerçekleştirilmektedir. Tablo 6.1'e göre, izin tabanlı çalışmalarda KNN algoritmasının parametrelerini detaylı şekilde inceleyen çalışma sınırlıdır. Bu nedenle, farklı veri kümeleri üzerinden uygulama izinleri çıkartılarak KNN algoritmasının parametreleri üzerine geniş kapsamlı deneyler yapılmaktadır.

Tablo 6.1. KNN algoritmasını kullanan bazı çalışmalar

Çalışma	İzin tabanlı mı?	Farklı k değerleri kullanılmış mı?	Farklı uzaklık metrikleri kullanılmış mı?
(Dini vd, 2012)	×	×	×
(Utku ve Dogru, 2017)	✓	✓	×
(Arslan ve Yurttakal, 2020)	✓	✓	×
(Cavli ve Sen, 2020)	×	×	×
(Kedziora vd, 2019)	×	×	×
(Baldini ve Geneiatakis, 2019)	×	✓	✓
(Darus vd, 2018)	×	×	×
(Darus vd, 2019)	×	×	×
(Lashkari vd, 2018)	×	×	×
(Abawajy vd, 2021)	✓	×	×
(Borja vd, 2013)	✓	✓	×
(W. Wang vd, 2018)	×	×	×
(Tahtaci ve Canbay, 2020)	×	×	×
(Sahin vd, 2021)	✓	✓	✓

6.2. Bölüm Katkısı

Bu bölümün ana katkıları şöyle özetlenebilir:

- KNN makine öğrenmesi tekniğini kullanan izin tabanlı Android kötücül yazılım tespiti için genel bir değerlendirme yapılarak en iyi parametrelerin bulunması sağlanmaktadır.
- 3 farklı veri kümesi kullanılarak geniş kapsamlı deneyler yapılmaktadır.
- 1'den 9'a kadar 5 farklı k değeri ve 5 farklı uzaklık metriği kullanılarak sınıflandırma işlemi gerçekleştirilmektedir.
- Son olarak elde edilen sınıflandırma başarımları sıkça kullanılan doğruluk ve F-ölçütü metriklerine göre elde edilip yorumlanmaktadır.

6.3. Deneysel Ayarlamalar

Bu bölümde yapılan deneyler 3 farklı veri kümesi üzerinde gerçekleştirilmektedir. Bu veri kümeleri sırasıyla, AMD (AMD, 2019; F. Wei vd, 2017), M0DROID (Damshenas vd, 2015) ve Lopez'in çalışmasında yer alan veri kümeleridir (Urcuqui-López ve Cadavid, 2016). AMD veri kümesi içerisinden rastgele alınan 1000 kötücül yazılıma ek olarak APKPure sitesinden (APKPure, 2021) indirilen 1000 iyicil örnek ile ilk veri kümesi oluşturulmaktadır. İkinci veri kümesi ise

(Damshenas vd, 2015) çalışmasının yazarları tarafından paylaşılmıştır. Bu veri kümesi içerisinde 200 iyicil ve 200 kötücül uygulama yer almaktadır. Son veri kümesi ise toplamda 558 uygulamadan oluşmaktadır. Bu uygulamalardan 279 tanesi kötücül olurken geri kalan uygulamalar ise iyicildir. AMD ve M0DROID veri kümelerinde uygulamalar ham halde bulunmaktadır. Bu nedenle, bir önceki bölümde yer alan Tablo 4.1'de detayı verilen algoritma ile bu veri kümelerinde yer alan uygulamaların izinleri çıkartılmaktadır. Bununla birlikte, AMD veri kümesinde toplam 102 doğal izin elde edilirken, M0DROID veri kümesinde ise 76 doğal izin elde edilmektedir. Lopez'in çalışmasında kullanılan veri kümesi ise doğrudan kullanıma hazır olup herhangi bir ön işlem aşamasına ihtiyaç duyulmadan kullanılmaktadır (Kaggle, 2021). Bu veri kümesinde ise doğal ve özel izinlerin yer aldığı toplam 330 izin bulunmaktadır.

AMD ve M0DROID veri kümeleri üzerinde sonuçlar alınırken 10 çapraz doğrulama tekniği kullanılmaktadır. Lopez'in çalışmasından alınan veriler eğitim ve test işlemleri için bölündüğünden bölünen parçalar göz önünde bulundurularak sınıflandırma işlemi gerçekleştirilmektedir. Bu veri kümesinde yer alan 398 uygulama eğitim için kullanılırken, 160 uygulama da test işlemi için kullanılmaktadır. KNN algoritması için WEKA kütüphanesi kullanılmaktadır. KNN algoritmasında sadece k parametresi ve uzaklık metrikleri değiştirilmektedir. Bunun haricinde diğer parametre ve ayarlar üzerinde herhangi bir değişiklik gerçekleştirilmemektedir.

6.4. Elde Edilen Sonuçlar

Bu bölümde, deneylerden elde edilen sonuçlar verilip yorumlanacaktır. Tablo 6.2'de AMD veri kümesinden elde edilen sonuçlar yer almaktadır. Tablo 6.2'de elde edilen en yüksek sınıflandırma başarımı F-ölçütü metriğine göre 0.945'tir. Bu sonuç k değeri 1 veya 3 seçildiğinde elde edilmektedir. Euclidean, Manhattan ve Minkowski uzaklık metrikleri kullanıldığında aynı sınıflandırma başarımları elde edilmektedir. Bunun 2 nedeni olabilir. Bunlardan birincisi bu uzaklık metriklerinin aynı altyapıya dayanmasıdır. İkincisi ise bu üç uzaklık metriğine göre en yakın k tane örneğin aynı sayıda seçim yapılması olarak değerlendirilebilir. k değeri 1 seçildiğinde doğruluk metriğine göre %94.5 oranında başarım elde edilirken, k değeri 3 seçildiğinde doğruluk metriğine göre %94.55 oranında başarım elde edilmektedir. k değerinin 3 seçilmesi bu veri kümesi için daha iyi sınıflandırma başarımı vermektedir. Filtered uzaklık metriğine göre sınıflandırma yapıldığında elde edilen en yüksek başarım F-ölçütü metriğine göre 0.93 olurken doğruluk metriğine göre %93.05'tir. Bu sonuç k

değeri 1 seçildiğinde elde edilmektedir. Bütün uzaklık metrikleri karşılaştırıldığında en kötü sınıflandırma başarımı Chebyshev metriği ile elde edilmektedir. Chebyshev uzaklık metriği ile elde edilen en yüksek başarım F-ölçütü metriğine göre 0.586'dır. Buna karşın en kötü sınıflandırma sonucu ise 0.405 olup bu sonuç k değeri 9 seçildiğinde elde edilmektedir.

Tablo 6.2. AMD veri kümesinden elde edilen sonuçlar

Uzaklık Metrikleri	$k = 1$		$k = 3$		$k = 5$		$k = 7$		$k = 9$	
	F	**D %**	**F**	**D %**	**F**	**D %**	**F**	**D %**	**F**	**D %**
Chebyshev	0.586	63.6	0.474	56.7	0.436	54.7	0.41	53.35	0.405	53.15
Euclidean	0.945	94.5	0.945	94.55	0.941	94.15	0.937	93.75	0.935	93.5
Filtered	0.93	93.05	0.924	92.45	0.917	91.7	0.911	91.15	0.907	90.7
Manhattan	0.945	94.5	0.945	94.55	0.941	94.15	0.937	93.75	0.935	93.5
Minkowski	0.945	94.5	0.945	94.55	0.941	94.15	0.937	93.75	0.935	93.5

F: F-ölçütü, **D:** Doğruluk

Tablo 6.3'te M0DROID veri kümesinden elde edilen sonuçlar yer almaktadır. Tablo 6.3'te elde edilen en yüksek sınıflandırma başarımı F-ölçütü metriğine göre 0.859'dur. Bu sonuç Euclidean, Manhattan, Minkowski uzaklık metriklerinden bir tanesi kullanıldığında ve k değeri 5 seçildiğinde elde edilmektedir. k değeri 1 ve 3 seçildiğinde, elde edilen en yüksek sınıflandırma başarımları F- ölçütü metriğine göre sırasıyla 0.842 ve 0.852'dir. k değeri 7 ve 9 olduğunda sınıflandırma başarımında dikkate değer azalmalar görülmektedir. Örneğin uzaklık metriği olarak Filtered kullanılıp k değeri de 1 olarak seçildiğinde sınıflandırma başarımı doğruluk metriğine göre %81.2'dir. Buna karşın k değeri için sırasıyla 7 ve 9 değerleri seçildiğinde elde edilen sınıflandırma başarımı doğruluk metriğine göre %76.69 ve %75.93'tür. Filtered metriğinde gözlemlenen bu durum Chebyshev metriğinde de görülmektedir. Bu uzaklık metriğe göre k değeri 7 veya 9 seçildiğinde, sınıflandırma başarımı %10 oranında düşmektedir. Bütün uzaklık metrikleri karşılaştırıldığında en kötü sınıflandırma başarımı Chebyshev metriği ile elde edilmektedir. Chebyshev uzaklık metriği ile elde edilen en yüksek başarım F-ölçütü metriğine göre 0.803'tür. Buna karşın en kötü sınıflandırma sonucu ise 0.703 olup bu sonuç k değeri 7 veya 9 seçildiğinde elde edilmektedir.

Tablo 6.3. M0DROID veri kümesinden elde edilen sonuçlar

Uzaklık Metrikleri	$k = 1$		$k = 3$		$k = 5$		$k = 7$		$k = 9$	
	F	D %	F	D %	F	D %	F	D %	F	D %
Chebyshev	0.803	80.45	0.748	75.43	0.706	71.67	0.703	71.42	0.703	71.42
Euclidean	0.842	84.21	0.852	85.21	0.859	85.96	0.837	83.7	0.834	83.45
Filtered	0.812	81.2	0.812	81.2	0.776	77.69	0.766	76.69	0.758	75.93
Manhattan	0.842	84.21	0.852	85.21	0.859	85.96	0.837	83.7	0.834	83.45
Minkowski	0.842	84.21	0.852	85.21	0.859	85.96	0.837	83.7	0.834	83.45

F: F-ölçütü, **D:** Doğruluk

Tablo 6.4'te Lopez'in veri kümesinden elde edilen sonuçlar yer almaktadır. Tablo 6.4'te elde edilen en yüksek sınıflandırma başarımı F-ölçütü metriğine göre 0.937'dir. Bu sonuç Chebyshev metriği haricindeki uzaklık metriklerinden bir tanesi kullanıldığında ve *k* değeri 1 seçildiğinde elde edilmektedir. Uzaklık metriği olarak Chebyshev kullanıldığında ve *k* değeri 1 seçildiğinde elde edilen sınıflandırma başarımı F-ölçütü metriğine göre 0.847'dir. Diğer veri kümelerinde olduğu gibi Chebyshev metriği ile sınıflandırma yapıldığında ve *k* değeri 3, 5, 7, 9 seçildiğinde sınıflandırma başarımında dikkate değer bir azalma gözlemlenmektedir. Euclidean, Manhattan, Minkowski metrikleri ele alındığında bu veri kümesi için en iyi sonuç *k* değerinin 1 seçilmesi ile elde edilmektedir. Buna karşın *k* değeri 3 seçildiğinde elde edilen sınıflandırma başarımı doğruluk metriğine göre %91.25'dir. Ek olarak, *k* değeri için 5, 7 ve 9 değerleri seçildiğinde elde edilen sınıflandırma başarımı ise doğruluk metriğine göre %90.62'dir. Diğer uzaklık metriklerinde olduğu gibi *k* değerinin 1 seçilmesi bu veri kümesi için en iyi sınıflandırma sonucunu vermektedir.

Tablo 6.4. Lopez'in veri kümesinden elde edilen sonuçlar

Uzaklık Metrikleri	$k = 1$		$k = 3$		$k = 5$		$k = 7$		$k = 9$	
	F	D %	F	D %	F	D %	F	D %	F	D %
Chebyshev	0.847	85	0.695	71.87	0.619	66.25	0.554	61.87	0.554	61.87
Euclidean	0.937	93.75	0.912	91.25	0.906	90.62	0.906	90.62	0.906	90.62
Filtered	0.937	93.75	0.894	89.37	0.881	88.12	0.894	89.37	0.868	86.87
Manhattan	0.937	93.75	0.912	91.25	0.906	90.62	0.906	90.62	0.906	90.62
Minkowski	0.937	93.75	0.913	91.25	0.906	90.62	0.906	90.62	0.906	90.62

F: F-ölçütü, **D:** Doğruluk

6.5. Var Olan Çalışmalar ile Karşılaştırma

Bu alt bölümde, güncel Android kötücül yazılım tespit sistemi çalışmaları ile bu bölümde yapılan deneylerden elde edilen sonuçlar karşılaştırılacaktır. Tablo 6.5'te KNN algoritmasının kullanıldığı Android kötücül yazılım çalışmalarından bazıları verilmektedir. (Dini vd, 2012) ve (Cavli ve Sen, 2020) çalışmaları dışında

araştırmacıların çoğunluğu kötücül yazılımlar ile iyicil yazılımların ayrıştırmaları üzerinde çalışmaktadır. (Lashkari vd, 2018) çalışmasında ise kötücül yazılımların ailelerine göre sınıflandırılması gibi birçok farklı senaryo denenmektedir. Bu bölümde yapılan deneylerde, genel olarak KNN algoritmasının Android kötücül yazılım tespitinde kullanımına odaklanıldığı için diğer çalışmaların sonuçları da verilmektedir. Tablo 6.5'e göre, KNN algoritmasından elde edilen en yüksek başarımın (Tahtaci ve Canbay, 2020) ve (Baldini ve Geneiatakis, 2019) çalışmalarından elde edildiği görülmektedir. Bu çalışmanın ardından en yüksek başarım (Cavli ve Sen, 2020) tarafından elde edilmektedir. KNN algoritmasından elde edilen en yüksek üçüncü sonuç (W. Wang vd, 2018) çalışmasından elde edilmektedir. Dördüncü en yüksek sonuç ise (Borja vd, 2013) çalışmasından elde edilmektedir. İzinlere ek olarak farklı statik özellikler (Baldini ve Geneiatakis, 2019; Borja vd, 2013; W. Wang vd, 2018) çalışmalarında ele alındığından, bu bölümde yapılan deneylerden daha iyi sonuçlar elde edilmiş olabilir. Ayrıca, (Tahtaci ve Canbay, 2020) smali dosyalarının n-gram özelliklerini kullanarak deneyleri gerçeklemektedir. En yüksek beşinci başarım ise bu bölümde AMD veri kümesinin kullanılmasıyla yapılan deneylerden elde edilmektedir. İyicil ve kötücül yazılımların sınıflandırılmasına odaklanan çalışmalar incelendiğinde bu bölümden elde edilen sonuçların genel olarak iyi olduğu görülmektedir.

(Abawajy vd, 2021; Arslan ve Yurttakal, 2020; Dini vd, 2012; Lashkari vd, 2018; Utku ve Dogru, 2017) çalışmalarından elde edilen sonuçlar ile bu bölümde yapılan deneylerden elde edilen sonuçlar arasında ufak farklılıklar vardır. (Dini vd, 2012) ve (Lashkari vd, 2018) çalışmalarında dinamik analiz tekniği kullanıldığı için tam anlamıyla karşılaştırma yapmak doğru değildir. Ancak (Abawajy vd, 2021; Arslan ve Yurttakal, 2020; Utku ve Dogru, 2017) çalışmalarında bu bölümde yapılan deneylerde olduğu gibi izin tabanlı statik analiz tekniği kullanılmaktadır. Her ne kadar (Abawajy vd, 2021; Arslan ve Yurttakal, 2020; Utku ve Dogru, 2017) ile benzer sonuçları elde etsek de bu bölümde yapılan deneylerde tam anlamıyla dengeli veri kümesi kullanılmaktadır. KNN algoritması ile en düşük sınıflandırma başarımı (Kedziora vd, 2019) ve (Darus vd, 2019) çalışmalarından elde edilmektedir. (Kedziora vd, 2019) çalışmasında statik özelliklerden kaynak kodlar öznitelik olarak kullanılmaktadır. Bu özniteliğin başta izin olmak üzere diğer statik öznitelikler kadar etkili olmadığı düşünülebilir. (Darus vd, 2019) çalışmasının düşük sonuç vermesinin iki nedeni olabilir. Bunlardan birincisi diğer çalışmaların aksine üç sınıflı problem

oluşturularak sınıflandırılmaktadır. Çoğu çalışmada kötücül ve iyicil olmak üzere iki sınıflı problem ele alınmaktadır. İkinci neden olarak doğrudan öznitelikleri kullanmak yerine görüntü işleme tekniklerinden yararlanılarak kötücül yazılım tespit sistemi tasarlanmaktadır.

Tablo 6.5. Önceki çalışmalar ile karşılaştırma

Çalışma	Analiz	Veri Kümesi	Amaç	Sınıflandırma Algoritması	Sonuçlar
(Dini vd, 2012)	D	-	ABMD	KNN, $k = 1$	%93 (Doğruluk)
(Utku ve Dogru, 2017)	S	5555 Kötücül 1339 İyicil	MBC	KNN, $k = 30$, Euclidean	%89.39 (Doğruluk)
(Arslan ve Yurttakal, 2020)	S	492 Kötücül 697 İyicil	MBC	KNN, $k = 5$, Minkowski	%93.7 (Doğruluk)
(Cavli ve Sen, 2020)	H	2535 Kötücül 21 Kötücül aile	MFC	KNN	%98.04 (Doğruluk)
(Kedziora vd, 2019)	S	996 Kötücül 962 İyicil	MBC	KNN, $k = 1$, Euclidean	%80.33 (Doğruluk)
(Baldini ve Geneiatakis, 2019)	S	5559 Kötücül 123452 İyicil	MBC	KNN, $k = 1$, CityBlock	%99.48 (Doğruluk)
(Darus vd, 2018)	S	183 Kötücül 300 İyicil	MBC	KNN	%80.69 (Doğruluk)
(Darus vd, 2019)	S	418 Kötücül 300 İyicil	MBC	KNN	%72.69 (Doğruluk)
(Lashkari vd, 2018)	D	429 Kötücül 5065 İyicil	MBC	KNN	%88.10 (Doğruluk)
(Abawajy vd, 2021)	S	5500 Kötücül 6190 İyicil	MBC	KNN, $k = 3$	%80-90 (Doğruluk)
(Borja vd, 2013)	S	333 Kötücül 333 İyicil	MBC	KNN, $k = 1$	%95.22 (Doğruluk)
(W. Wang vd, 2018)	S	8701 Kötücül 107327 İyicil	MBC	KNN, $k = 9$	%97.95 (Doğruluk)
(Tahtaci ve Canbay, 2020)	S	1500 Kötücül 1500 İyicil	MBC	KNN, $k = 3$, Euclidean	%99.83 (Doğruluk)
AMD sonuçları	S	1000 Kötücül 1000 İyicil	MBC	KNN, $k = 3$, Euclidean	%94.55 (Doğruluk)
M0DROID sonuçları	S	200 Kötücül 200 İyicil	MBC	KNN, $k = 5$, Euclidean	%85.96 (Doğruluk)
Lopez'den elde edilen sonuçlar	S	279 Kötücül 279 İyicil	MBC	KNN, $k = 1$, Euclidean	%93.75 (Doğruluk)

S: Statik, **D:** Dinamik, **H:** Hibrit, **MFC:** Kötücül yazılımların ailelerine göre sınıflandırılması, **MBC:** İyicil-kötücül yazılımların sınıflandırılması, **ABMD:** Anomali tabanlı kötücül yazılım tespiti

Tablo 6.5 genel olarak göz önünde bulundurulduğunda çalışma (Baldini ve Geneiatakis, 2019) dışında KNN algoritmasında parametre seçimine önem veren çalışma ile karşılaşılmamaktadır. Araştırmacıların bazıları ön tanımlı parametreler ile sınıflandırma işlemini gerçekleştirmektedir. (Arslan ve Yurttakal, 2020; Borja vd, 2013; Utku ve Dogru, 2017) çalışmalarında ise en başarılı sınıflandırma başarımını bulmak için sadece k sayısına çeşitli parametreler verilerek sınıflandırma işleminin en iyi olduğu k değerinin bulunması sağlanmaktadır. Bu çalışmalara ek olarak bu

bölümde yapılan deneylerde izin tabanlı Android kötücül yazılım tespitinde hem k hem de uzaklık metriklerinin sınıflandırma başarımına etkisini geniş kapsamlı deneylerle gösterilmektedir.

7. DERİN ÖĞRENME TEKNİKLERİNİN KARŞILAŞTIRILMASI

Bu bölümde 3 farklı derin öğrenme algoritması kullanılarak Android kötücül yazılımların sınıflandırılması gerçekleştirilmektedir. Bu teknikler DNN, 1D-CNN ve 2D-CNN'dir. DNN ve 1D-CNN tekniklerine öznitelik vektörleri doğrudan girdi olarak verilirken 2D-CNN tekniğine ise özellik vektörlerinden oluşturulmuş siyah beyaz görüntüler verilmektedir.

7.1. Bölüm Motivasyonu ve Katkı

Son zamanlarda, derin öğrenme görüntü sınıflandırma, doğal dil işleme, konuşma tanıma ve makine çevirisi gibi birçok alanda kullanılmaktadır. Bu çalışma konularına ek olarak Android kötücül yazılım tespiti alanında da derin öğrenme yaklaşımları sıkça tercih edilmektedir. Bu nedenle farklı derin öğrenme yaklaşımlarının Android kötücül yazılımda nasıl sonuç vereceği bu çalışmanın ana motivasyonunu oluşturmaktadır. Bunun yanında, son yıllarda uygulamaların görüntülere dönüştürülmesiyle kötücül yazılım tespitinin yapıldığı gözlemlenmektedir. İki boyutlu evrişimsel sinir ağları görüntü sınıflandırmada oldukça başarılı olduğundan statik özelliklerden oluşan öznitelik vektörü siyah beyaz görüntülere dönüştürülerek kötücül yazılımları iyicillerden ayrıştıran alternatif bir yaklaşım sunulmaktadır. Çalışmanın ana katkıları ise şöyle özetlenebilir:

- Derin sinir ağları, bir boyutlu evrişimsel sinir ağları (1D-CNN) ve iki boyutlu evrişimsel sinir ağları (2D-CNN) kullanılarak farklı veri kümeleri altında sınıflandırma başarımları kıyaslanmaktadır. Bu veri kümeleri Malgenome-215 ve Drebin-215'tir.
- Bu veri kümelerinde yer alan 215 statik öznitelik DNN ve 1D-CNN'e doğrudan verilmektedir. Buna karşın statik özelliklere padding işlemi uygulanarak her vektör 15×15'lik görüntülere dönüştürülmüştür. Ardından bu görüntüler 2D-CNN ile sınıflandırılmaktadır. Genel olarak, görüntüye dönüştürülerek yapılan sınıflandırmanın da diğer teknikler kadar başarılı olduğu görülmektedir.

7.2. Deneysel Ayarlamalar

Bu bölümde yapılan çalışmada, (S. Y. Yerima ve Sezer, 2019) çalışmasında kullanılan veri kümeleri kullanılmaktadır. Bu veri kümeleri Malgenome-215 ve

Drebin-215'tir (Drebin-215, 2021; Malgenome-215, 2021). Malgenome-215 veri kümesinde toplam 3799 uygulama yer almaktadır. Bu uygulamalardan 1260 tanesi kötücül iken 2539 tanesi ise iyicildir. Bu veri kümesi doğrudan kullanılmaktadır. Drebin-215 veri kümesinde toplam 15036 uygulama yer almaktadır. Bu uygulamalardan 5560 tanesi kötücül iken 9476 tanesi ise iyicildir. Drebin-215 veri kümesindeki, 178, 1973, 2111, 2952 ve 5176. sırada bulunan uygulamaların TelephonyManager.getSimCountryIso özniteliği "?" olarak görünmektedir. Bu öznitelik, veri kümesindeki uygulamalarda çoğunlukla 0 olarak bulunduğundan, "?" yerine 0 bilgisi yerleştirilerek deneyler gerçekleştirilmektedir.

Veri kümelerinde dört farklı statik özellik gruplarından öznitelikler yer almaktadır. Bunlar uygulama izinleri, API çağrıları, niyet filtreleri ve sistem çağrılarıdır. Toplamda 215 tane öznitelik ile özellik vektörleri oluşturulmaktadır. Böylece her bir uygulama 1×215'lik vektörler ile temsil edilmektedir. Bu sayede makine öğrenmesi tekniklerinin çalıştırılabileceği uygun yapı oluşturulmaktadır. Veri kümesi bu haliyle DNN ve 1D-CNN ile doğrudan sınıflandırılmaktadır. 2D-CNN görüntü sınıflandırmada oldukça başarılıdır. Ayrıca görüntülerden özellik çıkarımını otomatik bir şekilde yaptığı için veri kümelerine Eşitlik 7.1 uygulanarak her bir uygulama görüntü olarak temsil edilmektedir. Ardından elde edilen görüntüler ile sınıflandırma işlemi gerçekleştirilmektedir.

$$piksel(x,y) = \begin{cases} 255, & eğer\ öznitelik\ değeri = 1 \\ 0, & eğer\ öznitelik\ değeri = 0 \end{cases} \quad \textbf{(7.1)}$$

Veri kümelerinde 215 tane öznitelik olduğundan boyutu en yakın tam kareye dönüştürmek için padding işlemi yapılmaktadır. Özellik vektörüne 10 tane 0 eklenerek boyut 225'e çıkartılmaktadır. Genel olarak, uygulamalar ilgili özniteliği içeriyorsa 1 içermiyorsa 0 şeklinde veri kümesi oluşturulmaktadır. İlgili özniteliğin değeri 0 ise ilgili pikselin değeri 0, ilgili özniteliğin değeri 1 ise ilgili pikselin değeri 255 yapılarak uygulamaların görüntülere dönüştürülmesi sağlanmaktadır. Sonuç olarak, her bir uygulama 15×15 siyah beyaz görüntülere dönüştürülerek temsil edilmektedir.

DNN, 2D-CNN ve 1D-CNN olmak üzere oluşturulan 3 farklı model üzerinde sınıflandırma yapılmıştır. Oluşturulan modellerde bütün ara katmanlarda aktivasyon fonksiyonu olarak ReLU, incelenen problem iki sınıflı bir problem olduğundan çıktı katmanlarında sigmoid aktivasyon fonksiyonu kullanılmıştır. Optimizasyon yöntemi

olarak ise öğrenme parametresini dinamik olarak güncelleyen yapısı nedeniyle RMSProp yöntemi kullanılmıştır.

Oluşturulan modellerden ilki olan DNN'de girdi katmanından sonra sırasıyla 128, 64, 32 boyutlu katmanlar ve son olarak 1 boyutlu çıktı katmanı kullanılmıştır. 2D-CNN modelinde ise sırasıyla 32 ve 64 boyutlu iki evrişim katmanı kernel boyutu 3×3 olacak şekilde kullanılmıştır. Bu iki katmanın devamında 2×2 kernel boyutuna sahip bir MaxPooling katmanı bulunmaktadır. Bu katmandan elde edilen çıktılar vektörleştirilerek 512 boyutlu bir Dense katmanına verilmektedir ve devamında 1 boyutlu çıktı katmanı bulunmaktadır. Ek olarak bu iki katman arasında 0.2 Dropout oranı bulunmaktadır. Dropout katmanı oluşturulan modellerin aşırı öğrenmesini engelleyerek model performasını artırmaktadır. 1D-CNN modelinde ise yine 2D-CNN modelindeki gibi aynı yapı 3×1 boyutlu bir filtre ile kullanılmıştır. Tek fark ise 2×1 kernel boyutlu MaxPooling katmanından elde edilen çıktıların 128 boyutlu bir Dense katmanına verilmesidir. 1D-CNN modelinde böyle bir tercihin yapılmasının sebebi 128 boyut yerine 512 boyutlu bir katman kullanıldığında eğitilmesi gereken parametre sayısının çok fazla artmasıdır. Bu durumda hesaplama maliyeti artacaktır. Bu nedenle bu ağ üzerinde böyle bir güncelleme yapılmaktadır. Her iki veri kümesinde de verinin %70'i eğitim, %15'i doğrulama ve geri kalan %15'i test işlemi için kullanılmaktadır. Adil değerlendirme yapabilmek için DNN, 1D-CNN ve 2D-CNN tekniklerine ayrılan aynı eğitim, doğrulama ve test örnekleri verilmektedir.

7.3. Elde Edilen Sonuçlar

Tablo 7.1, 7.2 ve 7.3'te Malgenome-215 veri kümesinden elde edilen sonuçlar yer almaktadır. Tablo 7.1'de DNN tekniğinden elde edilen sonuçlar yer almaktadır. Tablo 7.1'de verilen karmaşıklık matrisine göre elde edilen başarım doğruluk metriğine göre 0.9947'dir. Benzer şekilde Tablo 7.2'de 1D-CNN tekniğinden elde edilen sonuçlar verilmektedir. 1D-CNN tekniği ise doğruluk metriğine göre 0.9982 sonucunu vermektedir. Tablo 7.3'te görüntülerin 2D-CNN ile sınıflandırılmasından elde edilen sonuçlar yer almaktadır. Görüntü sınıflandırmanın da diğer yöntemler kadar başarılı olduğu görülmektedir. 2D-CNN tekniğinden elde edilen başarım doğruluk metriğine göre 0.9947'dir. DNN ile yapılan sınıflandırmada 1 kötücül uygulama iyicil olarak sınıflandırılırken, 2 tane iyicil uygulama ise kötücül olarak sınıflandırılmaktadır. Buna karşın 2D-CNN ile yapılan deneylerde 2 kötücül uygulama iyicil olarak sınıflandırılırken, 1 tane iyicil uygulama ise kötücül olarak

sınıflandırılmaktadır. Malgenome-215 veri kümesi göz önünde bulundurulduğunda, 1D-CNN tekniği DNN ve 2D-CNN'e göre daha başarılıdır. Çünkü 1D-CNN tekniği iyicil uygulamaların tamamını doğru tahmin etmektedir. Buna karşın sadece 1 kötücül uygulama iyicil olarak tahmin edilmektedir.

Tablo 7.1. Malgenome-215 veri kümesinden elde edilen sonuçlar (DNN)

		Tahmini Sınıf	
		Kötücül	**İyicil**
Gerçek Sınıf	**Kötücül**	188	1
	İyicil	2	379

Tablo 7.2. Malgenome-215 veri kümesinden elde edilen sonuçlar (1D-CNN)

		Tahmini Sınıf	
		Kötücül	**İyicil**
Gerçek Sınıf	**Kötücül**	188	1
	İyicil	0	381

Tablo 7.3. Malgenome-215 veri kümesinden elde edilen sonuçlar (2D-CNN)

		Tahmini Sınıf	
		Kötücül	**İyicil**
Gerçek Sınıf	**Kötücül**	187	2
	İyicil	1	380

Tablo 7.4, 7.5 ve 7.6'da Drebin-215 veri kümesinden elde edilen sonuçlar yer almaktadır. Tablo 7.4'te DNN tekniğinden elde edilen sonuçlar yer almaktadır. Tablo 7.4'te verilen karmaşıklık matrisine göre 0.9783 doğruluk oranı elde edilmektedir. Benzer şekilde, Tablo 7.5'te 1D-CNN tekniğinden elde edilen sonuçlar verilmektedir. 1D-CNN tekniği ise doğruluk metriğine göre 0.98 sonucunu vermektedir. Tablo 7.6'da görüntülerin 2D-CNN ile sınıflandırılmasından elde edilen sonuçlar yer almaktadır. Görüntü sınıflandırmanın da diğer yöntemler kadar başarılı olduğu görülmektedir. Bu sonuç doğruluk metriğine göre 0.9769'dur. DNN ile yapılan sınıflandırmada 24 kötücül uygulama iyicil olarak sınıflandırılırken, 25 tane iyicil uygulama ise kötücül olarak sınıflandırılmaktadır. Buna karşın 2D-CNN ile yapılan deneylerde 32 kötücül uygulama iyicil olarak sınıflandırılırken, 20 tane iyicil uygulama ise kötücül olarak sınıflandırılmaktadır. Son olarak, 1D-CNN tekniğinde 30 kötücül uygulama iyicil olarak sınıflandırılırken, 15 tane iyicil uygulama ise kötücül olarak sınıflandırılmaktadır. Drebin-215 veri kümesi göz önünde bulundurulduğunda, 1D-

CNN tekniği DNN ve 2D-CNN'e göre daha başarılıdır. 1D-CNN tekniği toplamda 45 tane uygulamanın etiketini yanlış tahmin etmektedir. Buna karşın yanlış tahmin sayısı DNN'de 49 olurken 2D-CNN'de ise 52'dir. Hem Malgenome-215 hem de Drebin-215 veri kümelerinde aynı özellik grupları kullanılmasına rağmen derin öğrenme teknikleri, Drebin-215 veri kümesi üzerindeki uygulamaları sınıflandırırken Malgenome-215'e göre daha fazla sınıflandırma hatası yapmıştır. Bunun nedeni, Drebin-215 veri kümesini oluşturan uygulamaların ayrım gücünün zor olmasından kaynaklanabilmektedir. Başka bir ifadeyle, bu veri kümesinde yer alan bazı kötücül uygulamalar iyicil gibi görünürken bazı iyicil uygulamalar ise kötücül gibi görünebilmektedir.

Tablo 7.4. Drebin-215 veri kümesinden elde edilen sonuçlar (DNN)

		Tahmini Sınıf	
		Kötücül	**İyicil**
Gerçek Sınıf	**Kötücül**	810	24
	İyicil	25	1396

Tablo 7.5. Drebin-215 veri kümesinden elde edilen sonuçlar (1D-CNN)

		Tahmini Sınıf	
		Kötücül	**İyicil**
Gerçek Sınıf	**Kötücül**	804	30
	İyicil	15	1406

Tablo 7.6. Drebin-215 veri kümesinden elde edilen sonuçlar (2D-CNN)

		Tahmini Sınıf	
		Kötücül	**İyicil**
Gerçek Sınıf	**Kötücül**	802	32
	İyicil	20	1401

8. GENEL SONUÇLAR VE DEĞERLENDİRME

Android işletim sistemi, mobil cihazlar başta olmak üzere günümüzde otomobiller, beyaz eşyalar, fotoğraf makineleri, akıllı saatler ve giyilebilir cihazlar gibi çok sayıda ürünün içerisinde bulunmaktadır. Android işletim sistemine sahip cihazları kötücül yazılımlardan ve saldırılardan korumak oldukça önemlidir. Bu nedenle, bu tez çalışmasında Android kötücül yazılımların tespitini daha verimli yapabilmenin yolları araştırılmaktadır. Bu bağlamda; Android işletim sisteminin güvenliğinde önemli yeri olan izinler öznitelik olarak değerlendirilmiştir. Ardından bu izinler makine öğrenmesi yaklaşımları ile değerlendirilerek kötücül yazılımlar ile iyicil yazılımların birbirlerinden ayrıştırılması sağlanmıştır.

Bölüm 3'te, filtreleme tabanlı öznitelik seçme yöntemlerinin izin tabanlı Android kötücül yazılım tespitinde kullanımı ele alınmıştır. Bölüm 3'te yapılan deneylerde, filtreleme tabanlı öznitelik seçme tekniklerinin Android kötücül yazılım tespiti üzerinde sonuçları detaylı bir şekilde incelenmiştir. Var olan ve uyarlanan 8 farklı metrik ile öznitelik seçme işlemi gerçekleştirilmiştir. Uyarlanan metriklerin var olan metriklere göre az öznitelikte genel olarak daha iyi sonuçlar verdiği görülmüştür. Özellikle uyarlanan RFFS metriği ile elde edilen sonuçlar dikkate değerdir. Ancak, Acc2 metriğinin başarımı diğer uyarlanan metriklere göre düşüktür. İzin tabanlı Android kötücül yazılım tespitinde, izinlerin hepsini kullanmak yerine ayrım gücü en iyi izinlerin seçilmesinin makine öğrenmesi algoritmaları için oldukça önemli olduğu gösterilmiştir. Ele alınan veri kümesi göz önünde bulundurulduğunda, ayrım gücü en iyi izinler yaklaşık olarak tüm izinlerin %5'i ile %10'una denk gelmektedir. Böyle bir alt kümenin seçilmesi oldukça önemlidir. Bu deneyler ile ayrım gücü en yüksek izinlerin seçilmesini sağlayan altyapı oluşturulmuştur. Makine öğrenmesi algoritmaları üzerinde genel bir değerlendirme yapıldığında özellik sayısından en az etkilenen NB algoritmasıdır. Çoğu durumda NB algoritması benzer sonuçlar vermiştir. Tüm durumlar göz önünde bulundurulduğunda en yüksek sınıflandırma başarımı RF algoritması ile elde edilmiştir.

Bölüm 4'te, doğrusal regresyon tabanlı öznitelik seçme yönteminin izin tabanlı Android kötücül yazılım tespitinde kullanımı incelenmiştir. Mobil cihazların veya tabletlerin donanımları sınırlı olduğundan öznitelik seçme işlemi oldukça önemlidir. Bu nedenle önerilen kötücül yazılım tespit sisteminde doğrusal regresyona dayalı öznitelik seçme adımına yer verilmiştir. Bu sayede önemsiz izinlerin özellik

vektöründen çıkartılması gerçekleştirilmiştir. Böylece oluşturulan sınıflandırma modelinin verimliliği, uygulanabilirliği ve anlaşılabilirliğini sağlanmıştır. Öznitelik seçme yönteminin yanında, çeşitli izin grupları ile özellik vektörleri oluşturularak bu alanda çalışacak araştırmacılara doğrudan kullanılabilecek izin grubu önerileri sunulmuştur. Elde edilen sonuçlar incelendiğinde, önerilen kötücül yazılım tespit sisteminin 27 uygulama iznini öznitelik olarak kullanması ile dikkate değer başarımlara ulaştığı gösterilmektedir.

Bölüm 5'te, doğrusal regresyon modellerinden yararlanılarak kural tabanlı iki sınıflandırma modeli ile Android kötücül yazılımların tespit edilmesi gerçekleştirilmiştir. Önerilen kural tabanlı sınıflandırıcılar KNN, NB, SVM ve DT gibi popüler sınıflandırma algoritmaları ile kıyaslanmıştır. Her iki yaklaşım da NB ve KNN'den daha başarılı sonuçlar vermiştir. SVM, KNN ve NB algoritmalarında çok sayıda parametre bulunmaktadır. Ancak doğrusal regresyon modeline dayalı sınıflandırıcılar oldukça basit yapıda olup kullanımı da kolaydır. Bu durum önerilen yaklaşımın en büyük avantajıdır. Ayrıca, Bölüm 5'te yapılan deneylerde, torbalama tekniğine dayalı topluluk öğrenmesi modelleri de geliştirilmiştir. Son olarak, doğrusal regresyon modelinde regresyon katsayılarına rastgele değerler verilerek çok sayıda model oluşturulmuştur. Ancak bu modellerden olumlu sonuç alınamamıştır.

Bölüm 6'da, KNN algoritmasında yer alan k ve uzaklık metriği parametrelerinin izin tabanlı kötücül yazılım tespitinde başarımı nasıl etkilediğinin incelenmesidir. Kullanılan 3 farklı veri kümesi üzerindeki sonuçlara göre uzaklık metriği olarak Euclidean, Manhattan ve Minkowski metriklerinin aynı sonucu vermekle birlikte en başarılı metrikler olduğu gözlemlenmiştir. Buna karşın en kötü sınıflandırma sonucu ise Chebyshev metriği kullanıldığında elde edilmiştir. k değeri 7 ve 9 olduğunda bütün uzaklık metriklerinin başarımı düşmektedir. Buna karşın, en iyi sınıflandırma başarımları ise k değerine 1, 3 veya 5 verilmesi durumunda elde edilmiştir.

Bölüm 7'de, farklı derin öğrenme teknikleri kullanılarak Android kötücül yazılım tespiti üzerindeki başarımlar kıyaslanmıştır. DNN ve 1D-CNN tekniklerine statik özniteliklerden oluşan özellik vektörleri verilerek kötücül yazılımların tespit edilmesi sağlanmıştır. Ayrıca aynı özelliklerden siyah beyaz görüntüler elde edilerek 2D-CNN ile kötücül yazılımların tespitinin mümkün olduğu gösterilmiştir. Görüntü sınıflandırmada oldukça başarılı olan 2D-CNN bu çalışmada elde edilen görüntüleri de yüksek oranda doğru sınıflandırmıştır.

Gelecek çalışmalarda ise filtreleme tabanlı öznitelik seçme tekniklerinin dışında sarmalama ve gömme tabanlı öznitelik seçme tekniklerinin Android kötücül yazılım tespit sistemlerinde etkisi araştırılacaktır. KNN algoritmasında olduğu gibi diğer makine öğrenmesi algoritmalarında da kullanılan parametreler oldukça önemlidir. Bu nedenle, gelecek çalışmalarda önemli sınıflandırma algoritmalarının parametre seçimleri üzerine çalışılması planlanmaktadır. Bölüm 7'de yapılan deneylerde kullanılan veri kümelerindeki öznitelik sayısı az olduğundan oldukça küçük görüntüler üzerinde işlemler gerçekleştirilmektedir. Statik özellik sayısı arttığında daha büyük görüntüler elde edilebilir. Bu sayede başarımın artması da muhtemeldir. Bu nedenle gelecek çalışmalarda öznitelik sayısının arttırılarak daha büyük görüntülerin oluşturulması hedeflenmektedir. Ayrıca transfer öğrenme tekniklerinin yanı sıra LSTM ve RNN gibi farklı derin öğrenme tekniklerini kullanarak modellerinin başarımlarının kıyaslanması düşünülmektedir.

KAYNAKLAR

AAPT2. (2021, Kasım). Android asset packaging tool. Retrieved from https://developer.android.com/studio/command-line/aapt2

Abawajy, J., Darem, A. and Alhashmi, A. A. (2021). Feature Subset Selection for Malware Detection in Smart IoT Platforms. *Sensors, 21*(4), 1374. Retrieved from https://www.mdpi.com/1424-8220/21/4/1374

Abdullah, Z., Saudi, M. M. and Anuar, N. B. (2017). ABC: android botnet classification using feature selection and classification algorithms. *Advanced Science Letters, 23*(5), 4717-4720.

Alazab, M., Alazab, M., Shalaginov, A., Mesleh, A. and Awajan, A. (2020). Intelligent mobile malware detection using permission requests and API calls. *Future Generation Computer Systems, 107*, 509-521. doi:https://doi.org/10.1016/j.future.2020.02.002

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., . . . Asari, V. K. (2019). A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics, 8*(3), 292. Retrieved from https://www.mdpi.com/2079-9292/8/3/292

Alshahrani, H., Mansourt, H., Thorn, S., Alshehri, A., Alzahrani, A. and Fu, H. (2018, 12-14 Jan. 2018). *DDefender: Android application threat detection using static and dynamic analysis.* Paper presented at the 2018 IEEE International Conference on Consumer Electronics (ICCE).

Alswaina, F. and Elleithy, K. (2018). Android Malware Permission-Based Multi-Class Classification Using Extremely Randomized Trees. *IEEE Access, 6*, 76217-76227. doi:10.1109/ACCESS.2018.2883975

Altaher, A. (2016). Classification of android malware applications using feature selection and classification algorithms. *VAWKUM Transactions on Computer Sciences, 4*(1), 31-35.

Alzaylaee, M. K., Yerima, S. Y. and Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. *Computers & Security, 89*, 101663. doi:https://doi.org/10.1016/j.cose.2019.101663

AMD. (2019, Ekim). Android Malware Dataset. Retrieved from http://amd.arguslab.org/

Amin, M., Tanveer, T. A., Tehseen, M., Khan, M., Khan, F. A. and Anwar, S. (2020). Static malware detection and attribution in android byte-code through an end-to-end deep system. *Future Generation Computer Systems, 102*, 112-126. doi:https://doi.org/10.1016/j.future.2019.07.070

Ananya, A., Aswathy, A., Amal, T. R., Swathy, P. G., Vinod, P. and Mohammad, S. (2020). SysDroid: a dynamic ML-based android malware analyzer using system call traces. *Cluster Computing, 23*(4), 2789-2808. doi:10.1007/s10586-019-03045-6

Androguard. (2021, Kasım). Welcome to Androguard's documentation. Retrieved from https://androguard.readthedocs.io/en/latest/

Android. (2021, Kasım). Open Handset Alliance. Retrieved from https://www.openhandsetalliance.com/

APKPure. (2021, Kasım). APKPure Android application store. Retrieved from https://apkpure.com/tr/

Apktool. (2021, Kasım). A tool for reverse engineering Android apk files. Retrieved from https://ibotpeaches.github.io/Apktool/

Arauzo-Azofra, A., Aznarte, J. L. and Benítez, J. M. (2011). Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert Systems with Applications, 38*(7), 8170-8177. doi:https://doi.org/10.1016/j.eswa.2010.12.160

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K. and Siemens, C. (2014). *Drebin: Effective and explainable detection of android malware in your pocket.* Paper presented at the Network and Distributed System Security Symposium (NDSS).

Arslan, R. S. (2021). AndroAnalyzer: android malicious software detection based on deep learning. *PeerJ Computer Science, 7*, e533.

Arslan, R. S., Doğru, İ. A. and Barişçi, N. (2019). Permission-Based Malware Detection System for Android Using Machine Learning Techniques. *International Journal of Software Engineering and Knowledge Engineering, 29*(01), 43-61. doi:10.1142/S0218194019500037

Arslan, R. S. and Yurttakal, A. H. (2020). K-nearest neighbour classifier usage for permission based malware detection in android. *ICONTECH INTERNATIONAL JOURNAL, 4*(2), 15-27.

Bai, H., Xie, N., Di, X. and Ye, Q. (2020). FAMD: A Fast Multifeature Android Malware Detection Framework, Design, and Implementation. *IEEE Access, 8*, 194729-194740. doi:10.1109/ACCESS.2020.3033026

Bakour, K. and Ünver, H. M. (2021a). DeepVisDroid: android malware detection by hybridizing image-based features with deep learning techniques. *Neural Computing and Applications, 33*(18), 11499-11516. doi:10.1007/s00521-021-05816-y

Bakour, K. and Ünver, H. M. (2021b). VisDroid: Android malware classification based on local and global image features, bag of visual words and machine learning techniques. *Neural Computing and Applications, 33*(8), 3133-3153. doi:10.1007/s00521-020-05195-w

Baldini, G. and Geneiatakis, D. (2019). *A performance evaluation on distance measures in KNN for mobile malware detection.* Paper presented at the 2019 6th international conference on control, decision and information technologies (CoDIT).

Berman, D. S., Buczak, A. L., Chavis, J. S. and Corbett, C. L. (2019). A Survey of Deep Learning Methods for Cyber Security. *Information, 10*(4), 122. Retrieved from https://www.mdpi.com/2078-2489/10/4/122

Bhat, P. and Dutta, K. (2019). A Survey on Various Threats and Current State of Security in Android Platform. *ACM Comput. Surv., 52*(1), Article 21. doi:10.1145/3301285

Bhattacharya, A. and Goswami, R. T. (2018, 2018//). *A Hybrid Community Based Rough Set Feature Selection Technique in Android Malware Detection.* Paper presented at the Smart Trends in Systems, Security and Sustainability, Singapore.

Bhattacharya, A., Goswami, R. T. and Mukherjee, K. (2019). A feature selection technique based on rough set and improvised PSO algorithm (PSORS-FS) for permission based detection of Android malwares. *International Journal of Machine Learning and Cybernetics, 10*(7), 1893-1907. doi:10.1007/s13042-018-0838-1

Borja, S., Santos, I., Laorden, C., Ugarte-Pedrero, X., Nieves, J., Bringas, P. G. and Álvarez Marañón, G. (2013). MAMA: MANIFEST ANALYSIS FOR MALWARE DETECTION IN ANDROID. *Cybernetics and Systems, 44*(6-7), 469-488. doi:10.1080/01969722.2013.803889

Burguera, I., Zurutuza, U. and Nadjm-Tehrani, S. (2011). *Crowdroid: behavior-based malware detection system for Android*. Paper presented at the Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, Chicago, Illinois, USA. https://doi.org/10.1145/2046614.2046619

Cavli, O. F. T. and Sen, S. (2020, 3-4 Dec. 2020). *Familial Classification of Android Malware using Hybrid Analysis.* Paper presented at the 2020 International Conference on Information Security and Cryptology (ISCTURKEY).

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering, 40*(1), 16-28. doi:https://doi.org/10.1016/j.compeleceng.2013.11.024

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273-297. doi:10.1007/BF00994018

Çoban, Ö. and Özel, S. A. (2019). Adapting text categorization for manifest based android malware detection. *Computer Science, 20*(3), 305-327. doi:10.7494/csci.2019.20.3.3285

Damshenas, M., Dehghantanha, A., Choo, K.-K. R. and Mahmud, R. (2015). M0Droid: An Android Behavioral-Based Malware Detection Model. *Journal of Information Privacy and Security, 11*(3), 141-157. doi:10.1080/15536548.2015.1073510

Dargan, S., Kumar, M., Ayyagari, M. R. and Kumar, G. (2020). A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering, 27*(4), 1071-1092. doi:10.1007/s11831-019-09344-w

Darus, F. M., Ahmad, N. A. and Ariffin, A. F. M. (2019, 5-7 Nov. 2019). *Android Malware Classification Using XGBoost On Data Image Pattern.* Paper presented at the 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS).

Darus, F. M., Salleh, N. A. A. and Ariffin, A. F. M. (2018, 13-15 Nov. 2018). *Android Malware Detection Using Machine Learning on Image Patterns.* Paper presented at the 2018 Cyber Resilience Conference (CRC).

Deepa, K., Radhamani, G. and Vinod, P. (2015). Investigation of Feature Selection Methods for Android Malware Analysis. *Procedia Computer Science, 46*, 841-848. doi:https://doi.org/10.1016/j.procs.2015.02.153

dex2jar. (2021, Kasım). Source codes of dex2jar. Retrieved from https://github.com/pxb1988/dex2jar

Dharmalingam, V. P. and Palanisamy, V. (2021). A novel permission ranking system for android malware detection—the permission grader. *Journal of Ambient Intelligence and Humanized Computing, 12*(5), 5071-5081. doi:10.1007/s12652-020-01957-5

Dietterich, T. G. (2000). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning, 40*(2), 139-157. doi:10.1023/A:1007607513941

Dimjašević, M., Atzeni, S., Ugrina, I. and Rakamaric, Z. (2016). *Evaluation of Android Malware Detection Based on System Calls*. Paper presented at the Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics, New Orleans, Louisiana, USA. https://doi.org/10.1145/2875475.2875487

Ding, Y., Zhao, W., Wang, Z. and Wang, L. (2018, 15-18 July 2018). *Automaticlly Learning Featurs Of Android Apps Using CNN.* Paper presented at the 2018 International Conference on Machine Learning and Cybernetics (ICMLC).

Dini, G., Martinelli, F., Saracino, A. and Sgandurra, D. (2012, 2012//). *MADAM: A Multi-level Anomaly Detector for Android Malware*. Paper presented at the Computer Network Security, Berlin, Heidelberg.

Doğru, İ. A. and Dincer, C. A. (2017). Android Kötücül Yazılım Tespiti Yaklaşımları. *Uluslararası Bilgi Güvenliği Mühendisliği Dergisi, 3*(2), 48-58.

Dorogush, A. V., Ershov, V. and Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Douglas C., M., Elizabeth A., P. and G. Geoffrey, V. (2012). *Introduction to Linear Regression Analysis* (5th ed.): Wiley.

Drebin-215. (2021, Kasım). Drebin-215: Android malware dataset for machine learning. Retrieved from https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653

DroidBox. (2021, Kasım). DroidBox dynamic analysis tool. Retrieved from https://github.com/pjlantz/droidbox

Duc, N. V. and Giang, P. T. (2018). *NADM: Neural Network for Android Detection Malware*. Paper presented at the Proceedings of the Ninth International Symposium on Information and Communication Technology, Danang City, Viet Nam. https://doi.org/10.1145/3287921.3287977

Enck, W., Ongtang, M. and McDaniel, P. (2009). *On lightweight mobile phone application certification*. Paper presented at the Proceedings of the 16th ACM conference on Computer and communications security, Chicago, Illinois, USA. https://doi.org/10.1145/1653662.1653691

Enck, W., Ongtang, M. and McDaniel, P. (2009). Understanding Android Security. *IEEE Security & Privacy, 7*(1), 50-57. doi:10.1109/MSP.2009.26

Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M. and Rajarajan, M. (2015). Android Security: A Survey of Issues, Malware Penetration, and Defenses. *IEEE Communications Surveys & Tutorials, 17*(2), 998-1022. doi:10.1109/COMST.2014.2386139

Fatima, A., Maurya, R., Dutta, M. K., Burget, R. and Masek, J. (2019, 1-3 July 2019). *Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning.* Paper presented at the 2019 42nd International Conference on Telecommunications and Signal Processing (TSP).

Feizollah, A., Anuar, N. B., Salleh, R., Suarez-Tangil, G. and Furnell, S. (2017). AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection. *Computers & Security, 65*, 121-134. doi:https://doi.org/10.1016/j.cose.2016.11.007

Feizollah, A., Anuar, N. B., Salleh, R. and Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. *Digital Investigation, 13*, 22-37. doi:https://doi.org/10.1016/j.diin.2015.02.001

Fereidooni, H., Conti, M., Yao, D. and Sperduti, A. (2016, 21-23 Nov. 2016). *ANASTASIA: ANdroid mAlware detection using STatic analySIs of Applications.* Paper presented at the 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS).

Ganesh, M., Pednekar, P., Prabhuswamy, P., Nair, D. S., Park, Y. and Jeon, H. (2017, 24-25 July 2017). *CNN-Based Android Malware Detection.* Paper presented at the 2017 International Conference on Software Security and Assurance (ICSSA).

Gdatasoftware. (2021, Kasım). Mobile malware report - Android malware. Retrieved from https://www.gdatasoftware.com/news/2019/07/35228-mobile-malware-report-no-let-up-with-android-malware

George, F. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research, 3*(March), 1289–1305.

Gharib, A. and Ghorbani, A. (2017). *DNA-Droid: A Real-Time Android Ransomware Detection Framework*, Cham.

Hakak, S., Alazab, M., Khan, S., Gadekallu, T. R., Maddikunta, P. K. R. and Khan, W. Z. (2021). An ensemble machine learning approach through effective feature extraction

to classify fake news. *Future Generation Computer Systems, 117*, 47-58. doi:https://doi.org/10.1016/j.future.2020.11.022

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explor. Newsl., 11*(1), 10–18. doi:10.1145/1656274.1656278

Hao, F. and Wang, H. (2012, 15-17 March 2012). *An efficient linear regression classifier*. Paper presented at the 2012 IEEE International Conference on Signal Processing, Computing and Control.

Haq, I. U., Khan, T. A., Akhunzada, A. and Liu, X. (2021). MalDroid: Secure DL-enabled intelligent malware detection framework. *IET Communications*(Early access). doi:https://doi.org/10.1049/cmu2.12265

Hou, S., Saas, A., Chen, L. and Ye, Y. (2016, 13-16 Oct. 2016). *Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs.* Paper presented at the 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW).

Hou, S., Saas, A., Chen, L., Ye, Y. and Bourlai, T. (2017). *Deep Neural Networks for Automatic Android Malware Detection*. Paper presented at the Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia. https://doi.org/10.1145/3110025.3116211

JADX. (2021, Kasım). Source codes of JADX. Retrieved from https://github.com/skylot/jadx

JD-GUI. (2021, Kasım). JD-GUI - Java Decompiler. Retrieved from http://java-decompiler.github.io/

Jung, J., Park, J., Cho, S.-j., Han, S., Park, M. and Cho, H.-H. (2021). Feature Engineering and Evaluation for Android Malware Detection Scheme. *Journal of Internet Technology; Vol 22, No 2 (2021)*. Retrieved from https://jit.ndhu.edu.tw/article/view/2500

Jusoh, R., Firdaus, A., Anwar, S., Osman, M. Z., Darmawan, M. F. and Ab Razak, M. F. (2021). Malware detection using static analysis in Android: a review of FeCO (features, classification, and obfuscation). *PeerJ Computer Science, 7*, e522.

Kaggle. (2021, Kasım). Dataset malware/beningn permissions Android. Retrieved from https://www.kaggle.com/xwolf12/datasetandroidpermissions

Kaspersky. (2021a, Kasım). All about Android app permissions. Retrieved from https://www.kaspersky.com/blog/android-permissions-guide/14014/

Kaspersky. (2021b, Kasım). IT threat evolution Q2 2020. Mobile statistics. Retrieved from https://securelist.com/it-threat-evolution-q2-2020-mobile-statistics/98337/

Kaspersky. (2021c, Kasım). Malware in Minecraft mods: story continues. Retrieved from https://www.kaspersky.com/blog/minecraft-mod-adware-google-play-revisited/40202/

Kedziora, M., Gawin, P., Szczepanik, M. and Jozwiak, I. (2019). Malware detection using machine learning algorithms and reverse engineering of android java code. *International Journal of Network Security & Its Applications (IJNSA), 11*(1), 1-14.

Khandoker, T. I., Huang, D. and Sreeram, V. (2011, 13-16 Dec. 2011). *A low complexity linear regression approach to time synchronization in underwater networks.* Paper presented at the 2011 8th International Conference on Information, Communications & Signal Processing.

Khashei, M., Zeinal Hamadani, A. and Bijari, M. (2012). A novel hybrid classification model of artificial neural networks and multiple linear regression models. *Expert Systems with Applications, 39*(3), 2606-2620. doi:https://doi.org/10.1016/j.eswa.2011.08.116

Kim, T., Kang, B., Rho, M., Sezer, S. and Im, E. G. (2019). A Multimodal Deep Learning Method for Android Malware Detection Using Various Features. *IEEE Transactions on Information Forensics and Security, 14*(3), 773-788. doi:10.1109/TIFS.2018.2866319

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M. and Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing, 151*, 107398. doi:https://doi.org/10.1016/j.ymssp.2020.107398

Kouliaridis, V. and Kambourakis, G. (2021). A Comprehensive Survey on Machine Learning Techniques for Android Malware Detection. *Information, 12*(5), 185. Retrieved from https://www.mdpi.com/2078-2489/12/5/185

Kouliaridis, V., Potha, N. and Kambourakis, G. (2021, 2021). *Improving Android Malware Detection Through Dimensionality Reduction Techniques.* Paper presented at the Machine Learning for Networking, Cham.

Kural, O. E., Şahin, D. Ö., Akleylek, S. and Kılıç, E. (2019, 11-15 Sept. 2019). *Permission Weighting Approaches in Permission Based Android Malware Detection.* Paper presented at the 2019 4th International Conference on Computer Science and Engineering (UBMK).

Kurniawan, H., Rosmansyah, Y. and Dabarsyah, B. (2015, 10-11 Aug. 2015). *Android anomaly detection system using machine learning classification.* Paper presented at the 2015 International Conference on Electrical Engineering and Informatics (ICEEI).

Lan, M., Tan, C. L., Su, J. and Lu, Y. (2009). Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(4), 721-735. doi:10.1109/TPAMI.2008.110

Lashkari, A. H., Kadir, A. F. A., Taheri, L. and Ghorbani, A. A. (2018, 22-25 Oct. 2018). *Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification.* Paper presented at the 2018 International Carnahan Conference on Security Technology (ICCST).

Latif, J., Xiao, C., Tu, S., Rehman, S. U., Imran, A. and Bilal, A. (2020). Implementation and Use of Disease Diagnosis Systems for Electronic Medical Records Based on Machine Learning: A Complete Review. *IEEE Access, 8*, 150489-150513. doi:10.1109/ACCESS.2020.3016782

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444. doi:10.1038/nature14539

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324. doi:10.1109/5.726791

Lee, J., Jang, H., Ha, S. and Yoon, Y. (2021). Android Malware Detection Using Machine Learning with Feature Selection Based on the Genetic Algorithm. *Mathematics, 9*(21). doi:10.3390/math9212813

Li, J., Sun, L., Yan, Q., Li, Z., Srisa-an, W. and Ye, H. (2018). Significant Permission Identification for Machine-Learning-Based Android Malware Detection. *IEEE Transactions on Industrial Informatics, 14*(7), 3216-3225. doi:10.1109/TII.2017.2789219

Li, W., Wang, Z., Cai, J. and Cheng, S. (2018, 5-8 March 2018). *An Android Malware Detection Approach Using Weight-Adjusted Deep Learning.* Paper presented at the 2018 International Conference on Computing, Networking and Communications (ICNC).

Liang, H., Song, Y. and Xiao, D. (2017, 22-24 July 2017). *An end-to-end model for Android malware detection.* Paper presented at the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI).

Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D. and Liu, H. (2020). A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access*, *8*, 124579-124607. doi:10.1109/ACCESS.2020.3006143

Liu, Z., Wang, R., Japkowicz, N., Tang, D., Zhang, W. and Zhao, J. (2021). Research on unsupervised feature learning for Android malware detection based on Restricted Boltzmann Machines. *Future Generation Computer Systems*, *120*, 91-108. doi:https://doi.org/10.1016/j.future.2021.02.015

Lu, N., Li, D., Shi, W., Vijayakumar, P., Piccialli, F. and Chang, V. (2021). An efficient combined deep neural network based malware detection framework in 5G environment. *Computer Networks*, *189*, 107932. doi:https://doi.org/10.1016/j.comnet.2021.107932

Luo, W., Wei, L., Weng, J., Zhong, Y., Zhang, X. and Yan, Z. (2017). Machine Learning-Based Malicious Application Detection of Android. *IEEE Access*, *5*, 25591-25601. doi:10.1109/ACCESS.2017.2771470

Ma, Z., Ge, H., Wang, Z., Liu, Y. and Liu, X. (2020). Droidetec: Android malware detection and malicious code localization through deep learning. *arXiv preprint arXiv:2002.03594*.

Malgenome-215. (2021, Kasım). Malgenome-215: Android malware dataset for machine learning. Retrieved from https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_1/5854590/1

Masum, M. and Shahriar, H. (2019, 9-12 Dec. 2019). *Droid-NNet: Deep Learning Neural Network for Android Malware Detection.* Paper presented at the 2019 IEEE International Conference on Big Data (Big Data).

McLaughlin, N., Rincon, J. M. d., Kang, B., Yerima, S., Miller, P., Sezer, S., . . . Ahn, G. J. (2017). *Deep Android Malware Detection*. Paper presented at the Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, Scottsdale, Arizona, USA. https://doi.org/10.1145/3029806.3029823

Milosevic, N., Dehghantanha, A. and Choo, K.-K. R. (2017). Machine learning aided Android malware classification. *Computers & Electrical Engineering*, *61*, 266-274. doi:https://doi.org/10.1016/j.compeleceng.2017.02.013

Morales-Ortega, S., Escamilla-Ambrosio, P. J., Rodriguez-Mota, A. and Coronado-De-Alba, L. D. (2016, 18-21 Oct. 2016). *Native malware detection in smartphones with android OS using static analysis, feature selection and ensemble classifiers.* Paper presented at the 2016 11th International Conference on Malicious and Unwanted Software (MALWARE).

Naseem, I., Togneri, R. and Bennamoun, M. (2010). Linear Regression for Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(11), 2106-2112. doi:10.1109/TPAMI.2010.128

Nissim, N., Moskovitch, R., BarAd, O., Rokach, L. and Elovici, Y. (2016). ALDROID: efficient update of Android anti-virus software using designated active learning methods. *Knowledge and Information Systems*, *49*(3), 795-833. doi:10.1007/s10115-016-0918-z

Nix, R. and Zhang, J. (2017, 14-19 May 2017). *Classification of Android apps and malware using deep neural networks.* Paper presented at the 2017 International Joint Conference on Neural Networks (IJCNN).

Padmanabhan, J. and Johnson Premkumar, M. J. (2015). Machine Learning in Automatic Speech Recognition: A Survey. *IETE Technical Review, 32*(4), 240-251. doi:10.1080/02564602.2015.1010611

Pan, Y., Ge, X., Fang, C. and Fan, Y. (2020). A Systematic Literature Review of Android Malware Detection Using Static Analysis. *IEEE Access, 8*, 116363-116379. doi:10.1109/ACCESS.2020.3002842

Pehlivan, U., Baltaci, N., Acartürk, C. and Baykal, N. (2014, 9-12 Dec. 2014). *The analysis of feature selection methods and classification algorithms in permission based Android malware detection.* Paper presented at the 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS).

Pektaş, A. and Acarman, T. (2020). Deep learning for effective Android malware detection using API call graph embeddings. *Soft Computing, 24*(2), 1027-1043. doi:10.1007/s00500-019-03940-5

Pektaş, A., Çavdar, M. and Acarman, T. (2016, 2016//). *Android Malware Classification by Applying Online Machine Learning.* Paper presented at the Computer and Information Sciences, Cham.

Permissions, M. (2021, Kasım). The official site for Android app developers. Retrieved from https://developer.android.com/reference/android/Manifest.permission.html

Peynirci, G., Eminağaoğlu, M. and Karabulut, K. (2020). Feature Selection for Malware Detection on the Android Platform Based on Differences of IDF Values. *Journal of Computer Science and Technology, 35*(4), 946-962. doi:10.1007/s11390-020-9323-x

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines.

Polat, Ö. (2015). A robust regression based classifier with determination of optimal feature set. *Journal of applied research and technology, 13*(4), 443-446.

Sahin, D. Ö., Akleylek, S. and Kılıç, E. (2021). On the Effect of k Values and Distance Metrics in KNN Algorithm for Android Malware Detection. *Advances in Data Science and Adaptive Analysis*, 2141001. doi:10.1142/S2424922X21410011

Salah, A., Shalabi, E. and Khedr, W. (2020). A Lightweight Android Malware Classifier Using Novel Feature Selection Methods. *Symmetry, 12*(5). doi:10.3390/sym12050858

Santoku. (2021, Kasım). Welcome to Santoku Linux. Retrieved from https://santoku-linux.com/

Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P. G. and Álvarez, G. (2013, 2013//). *PUMA: Permission Usage to Detect Malware in Android.* Paper presented at the International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions, Berlin, Heidelberg.

Sayfullina, L., Eirola, E., Komashinsky, D., Palumbo, P., Miche, Y., Lendasse, A. and Karhunen, J. (2015, 20-22 Aug. 2015). *Efficient Detection of Zero-day Android Malware Using Normalized Bernoulli Naive Bayes.* Paper presented at the 2015 IEEE Trustcom/BigDataSE/ISPA.

Seal, A., Bhattacharjee, D., Nasipuri, M. and Basu, D. K. (2015). UGC-JU face database and its benchmarking using linear regression classifier. *Multimedia Tools and Applications, 74*(9), 2913-2937. doi:10.1007/s11042-013-1754-8

Senanayake, J., Kalutarage, H. and Al-Kadri, M. O. (2021). Android Mobile Malware Detection Using Machine Learning: A Systematic Review. *Electronics, 10*(13), 1606. Retrieved from https://www.mdpi.com/2079-9292/10/13/1606

Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C. and Weiss, Y. (2012). "Andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems, 38*(1), 161-190. doi:10.1007/s10844-010-0148-x

Shanmugam, P., Venkateswarulu, B., Dharmadurai, R., Ranganathan, T., Indiran, M. and Nanjappan, M. (2022). Electro search optimization based long short-term memory network for mobile malware detection. *Concurrency and Computation: Practice and Experience*, e7044. doi:https://doi.org/10.1002/cpe.7044

Sharmeen, S., Huda, S., Abawajy, J. and Hassan, M. M. (2020). An adaptive framework against android privilege escalation threats using deep learning and semi-supervised approaches. *Applied Soft Computing, 89*, 106089. doi:https://doi.org/10.1016/j.asoc.2020.106089

Shiqi, L., Shengwei, T., Long, Y., Jiong, Y. and Hua, S. (2018). Android malicious code classification using deep belief network. *KSII Transactions on Internet and Information Systems (TIIS), 12*(1), 454-475.

Srinivasan, K. and Fisher, D. (1995). Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering, 21*(2), 126-137. doi:10.1109/32.345828

Statista. (2021a, Kasım). Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2018. Retrieved from https://www.statista.com/statistics/266136/

Statista. (2021b, Kasım). Number of smartphone users from 2016 to 2021. Retrieved from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Su, X., Zhang, D., Li, W. and Zhao, K. (2016, 23-26 Aug. 2016). *A Deep Learning Approach to Android Malware Feature Learning and Detection.* Paper presented at the 2016 IEEE Trustcom/BigDataSE/ISPA.

Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P. and Blasco, J. (2014). Dendroid: A text mining approach to analyzing and classifying code structures in Android malware families. *Expert Systems with Applications, 41*(4, Part 1), 1104-1117. doi:https://doi.org/10.1016/j.eswa.2013.07.106

Şahin, D. Ö., Akleylek, S. and Kılıç, E. (2022). LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers. *IEEE Access, 10*, 14246-14259. doi:10.1109/ACCESS.2022.3146363

Şahin, D. Ö. and Kılıç, E. (2019). Two new feature selection metrics for text classification. *Automatika, 60*(2), 162-171. doi:10.1080/00051144.2019.1602293

Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2020, 5-7 Oct. 2020). *Comparison of Regression Methods in Permission Based Android Malware Detection.* Paper presented at the 2020 28th Signal Processing and Communications Applications Conference (SIU).

Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2021a). A novel Android malware detection system: adaption of filter-based feature selection methods. *Journal of Ambient Intelligence and Humanized Computing*. doi:10.1007/s12652-021-03376-6

Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2021b). A novel permission-based Android malware detection system using feature selection based on linear regression. *Neural Computing and Applications*. doi:10.1007/s00521-021-05875-1

Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2021c). Permission-based Android malware analysis by using dimension reduction with PCA and LDA. *Journal of Information Security and Applications, 63*, 102995. doi:https://doi.org/10.1016/j.jisa.2021.102995

Şahın, D. Ö., Kural, O. E., Akleylek, S. and Kiliç, E. (2018, 22-25 March 2018). *New results on permission based static analysis for Android malware*. Paper presented at the 2018 6th International Symposium on Digital Forensic and Security (ISDFS).

Tahtaci, B. and Canbay, B. (2020, 15-17 Oct. 2020). *Android Malware Detection Using Machine Learning.* Paper presented at the 2020 Innovations in Intelligent Systems and Applications Conference (ASYU).

Tang, L., Lu, H., Pang, Z., Li, Z. and Su, J. (2019). A distance weighted linear regression classifier based on optimized distance calculating approach for face recognition. *Multimedia Tools and Applications, 78*(22), 32485-32501. doi:10.1007/s11042-019-07943-0

Tao, G., Zheng, Z., Guo, Z. and Lyu, M. R. (2018). MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs. *IEEE Transactions on Reliability, 67*(1), 355-369. doi:10.1109/TR.2017.2778147

Taşcı, Ş. and Güngör, T. (2013). Comparison of text feature selection policies and using an adaptive framework. *Expert Systems with Applications, 40*(12), 4871-4886. doi:https://doi.org/10.1016/j.eswa.2013.02.019

Urcuqui-López, C. and Cadavid, A. N. (2016). Framework for malware analysis in Android. *Sistemas y Telemática, 14*(37), 45-56.

Urooj, B., Shah, M. A., Maple, C., Abbasi, M. K. and Riasat, S. (2022). Malware Detection: A Framework for Reverse Engineered Android Applications through Machine Learning Algorithms. *IEEE Access*, 1-1. doi:10.1109/ACCESS.2022.3149053

Utku, A. (2022). Ağ trafiği analizi ile derin öğrenme tabanlı Android kötücül yazılım tespiti. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 37*(4), 1823-1838.

Utku, A. and Dogru, I. (2017). Permission based detection system for android malware. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, *32*, 1015-1024.

VirusShare. (2021, Kasım). Official web site of VirusShare. Retrieved from https://virusshare.com/

Wang, H., Liu, Z., Liang, J., Vallina-Rodriguez, N., Guo, Y., Li, L., . . . Xu, G. (2018). *Beyond Google Play: A Large-Scale Comparative Study of Chinese Android App Markets*. Paper presented at the Proceedings of the Internet Measurement Conference 2018, Boston, MA, USA. https://doi.org/10.1145/3278532.3278558

Wang, W., Li, Y., Wang, X., Liu, J. and Zhang, X. (2018). Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers. *Future Generation Computer Systems, 78*, 987-994. doi:https://doi.org/10.1016/j.future.2017.01.019

Wang, W., Zhao, M., Gao, Z., Xu, G., Xian, H., Li, Y. and Zhang, X. (2019). Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy and Directions. *IEEE Access, 7*, 67602-67631. doi:10.1109/ACCESS.2019.2918139

Wang, Z., Cai, J., Cheng, S. and Li, W. (2016, 19-21 Sept. 2016). *DroidDeepLearner: Identifying Android malware using deep learning.* Paper presented at the 2016 IEEE 37th Sarnoff Symposium.

Wei, F., Li, Y., Roy, S., Ou, X. and Zhou, W. (2017, 2017//). *Deep Ground Truth Analysis of Current Android Malware.* Paper presented at the Detection of Intrusions and Malware, and Vulnerability Assessment, Cham.

Wei, X., Chen, W. and Li, X. (2021). Exploring the financial indicators to improve the pattern recognition of economic data based on machine learning. *Neural Computing and Applications, 33*(2), 723-737. doi:10.1007/s00521-020-05094-0

Wen, L. (2013, 14-17 July 2013). *Mutiple classifier system based android malware detection.* Paper presented at the 2013 International Conference on Machine Learning and Cybernetics.

Wu, D., Mao, C., Wei, T., Lee, H. and Wu, K. (2012, 9-10 Aug. 2012). *DroidMat: Android Malware Detection through Manifest and API Calls Tracing.* Paper presented at the 2012 Seventh Asia Joint Conference on Information Security.

Xiao, L., Wan, X., Lu, X., Zhang, Y. and Wu, D. (2018). IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Processing Magazine, 35*(5), 41-49. doi:10.1109/MSP.2018.2825478

Xiao, X. and Yang, S. (2019, 11-15 Nov. 2019). *An Image-Inspired and CNN-Based Android Malware Detection Approach.* Paper presented at the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE).

Xu, K., Li, Y. and Deng, R. H. (2016). ICCDetector: ICC-Based Malware Detection on Android. *IEEE Transactions on Information Forensics and Security, 11*(6), 1252-1264. doi:10.1109/TIFS.2016.2523912

Yerima, S. Y. and Sezer, S. (2019). DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. *IEEE Transactions on Cybernetics, 49*(2), 453-466. doi:10.1109/TCYB.2017.2777960

Yerima, S. Y., Sezer, S. and McWilliams, G. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security, 8*(1), 25-36.

Yerima, S. Y., Sezer, S., McWilliams, G. and Muttik, I. (2013, 25-28 March 2013). *A New Android Malware Detection Approach Using Bayesian Classification.* Paper presented at the 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA).

Yildiz, O. and Doğru, I. A. (2019). Permission-based Android Malware Detection System Using Feature Selection with Genetic Algorithm. *International Journal of Software Engineering and Knowledge Engineering, 29*(02), 245-262. doi:10.1142/S0218194019500116

Yuan, H., Tang, Y., Sun, W. and Liu, L. (2020). A detection method for android application security based on TF-IDF and machine learning. *PLOS ONE, 15*(9), e0238694. doi:10.1371/journal.pone.0238694

Yuan, Z., Lu, Y., Wang, Z. and Xue, Y. (2014). *Droid-Sec: deep learning in android malware detection*. Paper presented at the Proceedings of the 2014 ACM conference on SIGCOMM, Chicago, Illinois, USA. https://doi.org/10.1145/2619239.2631434

Yuan, Z., Lu, Y. and Xue, Y. (2016). Droiddetector: android malware characterization and detection using deep learning. *Tsinghua Science and Technology, 21*(1), 114-123. doi:10.1109/TST.2016.7399288

Zhang, Y., Yang, Y. and Wang, X. (2018). *A Novel Android Malware Detection Approach Based on Convolutional Neural Network*. Paper presented at the Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, Guiyang, China. https://doi.org/10.1145/3199478.3199492

Zhao, K., Zhang, D., Su, X. and Li, W. (2015, 6-9 July 2015). *Fest: A feature extraction and selection tool for Android malware detection.* Paper presented at the 2015 IEEE Symposium on Computers and Communication (ISCC).

Zou, X., Hu, Y., Tian, Z. and Shen, K. (2019, 19-20 Oct. 2019). *Logistic Regression Model Optimization and Case Analysis.* Paper presented at the 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT).

EKLER

EK-1 BAZI İZİNLER ve AÇIKLAMALARI

android.permission.MOUNT_UNMOUNT_FILESYSTEMS

Uygulamanın, çıkarılabilir depolama için dosya sistemlerini bağlamasına ve bağlantısını kesmesine izin verir.

android.permission.GET_TASKS

Uygulamanın mevcut çalışan ve son zamanlarda çalışan görevler hakkında bilgi almasına izin verir. Ayrıca uygulamanın cihazda hangi uygulamaların kullanıldığı hakkında bilgi almasına izin verir.

android.permission.READ_PHONE_STATE

Geçerli hücresel ağ bilgileri, devam eden aramaların durumu ve cihazda kayıtlı herhangi bir *PhoneAccounts* listesi dâhil olmak üzere telefon durumuna salt okunur erişim sağlar.

android.permission.CHANGE_WIFI_STATE

Uygulamaların Wi-Fi bağlantı durumunu değiştirmesine izin verir.

android.permission.CHANGE_NETWORK_STATE

Uygulamaya, ağ bağlantı durumunu değiştirme izni verir.

android.permission.READ_SMS

Uygulamaya, SMS mesajlarını okuma izni verir.

android.permission.SEND_SMS

Uygulamaya, SMS mesajları gönderme izni verir.

android.permission.RECEIVE_SMS

Uygulamaya, SMS mesajları alma izni verir.

android.permission.WRITE_SETTINGS

Uygulamaya, sistem ayarlarını okuma veya yazma izni verir.

android.permission.WRITE_SMS

Uygulamaya, cihazda veya SIM kartda depolanan SMS mesajlarına yazma izni verir. Kötü amaçlı uygulamalar var olan mesajları değiştirebilir veya silebilir.

android.permission.UPDATE_APP_OPS_STATS

Uygulamaya, uygulama çalışma istatistiklerini güncelleme izni verir. Üçüncü taraf uygulamaların kullanımına yönelik değildir.

android.permission.READ_SETTINGS

Uygulamaya, ana sayfadaki ayarları ve kısayolları okuma izni verir.

android.permission.RECEIVE_MMS

Uygulamaya, gelen MMS mesajlarını izleme izni verir.

android.permission.ACCESS_MTK_MMHW

FM radyo yongasına erişmek için kullanılır. MTK, elektronik yonga üreticisi olan MediaTek'i ifade etmektedir.

android.permission.SAMSUNG_TUNTAP

Samsung marka cihazlara özgü bir izindir.

android.permission.ACCESS_CACHE_FILESYSTEM

Bir uygulamanın önbellek bölümünü okumasına ve yazmasına izin verin.

android.permission.WRITE_APN_SETTINGS

Uygulamaların erişim noktası (Access Point Name veya apn) ayarlarını yazmasına ve kullanıcı ve şifre gibi mevcut apn ayarlarının hassas alanlarını okumasına izin verir. Üçüncü taraf uygulamaların kullanımına yönelik değildir.

android.permission.READ_INTERNAL_STORAGE

Dosyaları cihazın dâhili belleğine kaydetme konusunda destek almak için uygulama tarafından gereklidir. Aynı zamanda uygulamanın, uygulama tarafından oluşturulan dosyaları okumasını da sağlayacaktır.

android.permission.INTERNET

Uygulamaların ağ soketlerini açmasına izin verir.

android.permission.ACCESS_NETWORK_STATE

Uygulamaların ağlar hakkındaki bilgilere erişmesine izin verir.

android.permission.WRITE_EXTERNAL_STORAGE

Bir uygulamanın harici depolama birimine yazmasına izin verir.

android.permission.WAKE_LOCK

İşlemcinin uyku moduna geçmesini veya ekranın kararmasını önlemek için PowerManager WakeLocks kullanımına izin verir.

android.permission.ACCESS_WIFI_STATE

Uygulamaların Wi-Fi ağları hakkındaki bilgilere erişmesine izin verir.

android.permission.READ_EXTERNAL_STORAGE

Bir uygulamanın harici depolama biriminden okumasına izin verir.

android.permission.VIBRATE

Uygulamanın titreşim donanımını kullanılabilmesi sağlanır.

android.permission.RECEIVE_BOOT_COMPLETED

Uygulamaya, sistem önyüklemeyi tamamladıktan sonra yayınlanan Intent.ACTION_BOOT_COMPLETED'i alma izni verir.

android.permission.ACCESS_COARSE_LOCATION

Bir uygulamanın yaklaşık konuma erişmesine izin verir. ACCESS_FINE_LOCATION izni de alternatif olarak kullanılabilmektedir.

android.permission.SYSTEM_ALERT_WINDOW

Başka bir uygulamanın kullanıcı ara yüzünün nasıl görüntülediğinin değiştirilmesine izin verir. Bir uygulamanın, kullanıcıyla sistem düzeyinde tür etkileşimi kullanarak pencereleri açmasına izin verir.

com.android.vending.BILLING

Google Play'i kullanarak uygulama içi faturalandırma işlemlerini yönetmek için basit, basit bir ara yüz sağlar.

android.permission.ACCESS_LOCATION_EXTRA_COMMANDS

Bir uygulamanın ekstra konum sağlayıcı komutlarına erişmesine izin verir.

android.permission.CAMERA

Kameraya erişimle veya cihazdan görüntü / video çekmeyle ilişkili izinler için kullanılır. Kamera cihazına erişebilmek için gereklidir.

android.permission.ANSWER_PHONE_CALLS

Uygulamaya, gelen bir telefon aramasını yanıtlama izni verir.

android.permission.CALL_PHONE

Uygulamaya, kullanıcının aramayı onaylaması için çevirici ara yüzüne girmeden bir telefon görüşmesi başlatma izni verir.

android.permission.WRITE_CALENDAR

Uygulamaya, kullanıcının takvim verilerini yazma izni verir.

android.permission.READ_CALENDAR

Uygulamaya, kullanıcının takvim verilerini okuma izni verir.

android.permission.READ_PHONE_NUMBERS

Cihazın telefon numaralarının bulunduğu rehbere okuma erişimi sağlar. Bu, READ_PHONE_STATE izni tarafından sağlanan yeteneklerin bir alt kümesidir, ancak anlık uygulamalara açıktır.

android.permission.USE_BIOMETRIC

Uygulamaya, cihaz tarafından desteklenen biyometrik cihazları kullanma izni verir.

ÖZ GEÇMİŞ

Durmuş Özkan ŞAHİN, Tarsus Cengiz Topel Lisesi'ni bitirdikten sonra 2008 yılında Süleyman Demirel Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği bölümünü kazandı. 2013 yılında buradan mezun oldu. 2016 yılında OMÜ Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği ABD Yüksek Lisans programını bitirdi. 2016 yılından bu yana ise OMÜ Lisansüstü Eğitim Enstitüsü Hesaplamalı Bilimler ABD doktora programında eğitimine devam etmektedir. 2014 yılından bu yana Ondokuz Mayıs Üniversitesi Bilgisayar Mühendisliği bölümünde Araştırma Görevlisi olarak görev yapan Durmuş Özkan ŞAHİN, iyi derecede İngilizce bilmektedir. Temel ilgi alanları, programlama, makine öğrenmesi ve veri bilimidir.

24.05.2022

İletişim Bilgileri

ORCID ID : 0000-0002-0831-7825

Yayınlar:

1. Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kiliç, E. (2018, 22-25 March 2018). *New results on permission based static analysis for Android malware.* Paper presented at the 2018 6th International Symposium on Digital Forensic and Security (ISDFS).

2. Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2020, 5-7 Oct. 2020). *Comparison of Regression Methods in Permission Based Android Malware Detection.* Paper presented at the 2020 28th Signal Processing and Communications Applications Conference (SIU).

3. Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2021a). A novel Android malware detection system: adaption of filter-based feature selection methods. *Journal of Ambient Intelligence and Humanized Computing*. doi:10.1007/s12652-021-03376-6

4. Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2021b). A novel permission-based Android malware detection system using feature selection based on linear regression. *Neural Computing and Applications*. doi:10.1007/s00521-021-05875-1

5. Şahin, D. Ö., Akleylek, S. and Kılıç, E. (2021). On the Effect of k Values and Distance Metrics in KNN Algorithm for Android Malware Detection. *Advances in Data Science and Adaptive Analysis*, 2141001. doi:10.1142/S2424922X21410011

6. Şahin, D. Ö., Yazar, B. K., Akleylek, S, Kılıç, E. and Debasis, G. (2021, 1-3 Oct. 2021). On the Android Malware Detection System Based on Deep Learning.

Paper presented at the International Conference on Artificial Intelligence and Applied Mathematics in Engineering (ICAIAME)

7. Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2021c). Permission-based Android malware analysis by using dimension reduction with PCA and LDA. *Journal of Information Security and Applications, 63,* 102995. doi:https://doi.org/10.1016/j.jisa.2021.102995

8. Şahin, D. Ö., Akleylek, S. and Kılıç, E. (2022). LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers. *IEEE Access, 10,* 14246-14259. doi:10.1109/ACCESS.2022.3146363