Android Zararlı Yazılım Tespit Sistemi

Tülay Avan

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Haziran 2020

Android Malware Detection System

Tülay Avan

MASTER OF SCIENCE THESIS

Department of Computer Engineering

June 2020

Android Zararlı Yazılım Tespit Sistemi

Tülay Avan

Eskişehir Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Bilimleri Bilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman: **:** Dr. Öğr. Üyesi Esra N. Yolaçan

Haziran 2020

ETİK BEYAN

Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre, Dr. Öğr. Üyesi Esra N. Yolaçan danışmanlığında hazırlamış olduğum "Android Zararlı Yazılım Tespit Sistemi" başlıklı YÜKSEK LİSANS tezimin özgün bir çalışma olduğunu; tez çalışmamın tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; tezimde verdiğim bilgileri, verileri akademik ve bilimsel etik ilke ve kurallara uygun olarak elde ettiğimi; tez çalışmamda yararlandığım eserlerin tümüne atıf yaptığımı, kaynak gösterdiğimi, bilgi, belge ve sonuçları; bilimsel etik ilke ve kurallara göre sunduğumu beyan ederim. 29/06/2020

Tülay Avan

İmza

ÖZET

Akıllı telefonlar, sağladığı özelliklerin yanında kullanım ve taşıma kolaylığı ile birlikte kısa sürede her kesime hitap eden ve en çok kullanılan teknolojik aygıt durumuna gelmiştir. Mobil cihazlar kişisel kullanım dışında sektörel olarak da birçok alanda kullanılmaktadır. Mobil cihazların kullanımındaki hızlı artışla beraber siber tehditler, mobil dünyasının en büyük problemlerinden biri haline gelmiştir. Özellikle Nesnelerin Interneti (IOT-Internet of Things) kavramının hayatımıza girmesi ile birlikte, mobil cihazların güvenliği sadece kişisel bilgi güvenliği ile sınırlı kalmayıp, siber güvenliğin her alanında ve hatta fiziksel güvenlikte önemli bir yere sahip olmuştur. Her geçen gün saldırıların sayısının ve çeşitinin artması mobil zararlı yazılımların tespitinde daha hızlı ve doğru tespitler yapan yöntemler geliştirilmesi ihtiyacını doğurmuştur. Bunun yanında gerek versiyonlarının gerekse yöntemlerinin değişmesinden dolayı, yapay zeka tekniklerinin kullanılması da ön plana çıkmaya başlamıştır. Bu çalışmada, en çok kullanılan platform olan Android için geliştirilen mobil yazılımlarda zararlı yazılım tespiti yapmak amacı ile makine öğrenmesi ve derin öğrenme yöntemleri incelenmiştir. İncelenen yöntemler Android yazılımların statik analiz yöntemiyle elde edilen özellikleri kullanılarak üretilmiş olan veri seti kullanılarak test edilip, karşılaştırılmıştır. Elde edilen sonuçlar hem doğruluk oranları hem de ROC eğrisi ile karşılaştırmalı olarak sunulmuştur. Yapılan çalışma sonucunda derin öğrenme tabanlı sistemlerin daha iyi doğruluk oranına sahip olduğu görülmektedir.

Anahtar Kelimeler: Android, Zararlı Yazılım, Derin Öğrenme, Makine Öğrenmesi

SUMMARY

Smart phones have become the most widely used technological devices that appeal to all segments in a short period of time together with the ease of use and transportation they provide.In addition to personal use, mobile devices are used in many areas in the sector. With the rapid increase in the use of mobile devices, cyber threats have become one of the biggest problems in the mobile world. Especially with the introduction of the Internet of Things (IOT) concept, the security of mobile devices is not only limited to the security of personal information, but also has an important role in every aspect of cyber security and even physical security. As the number and type of attacks increase day by day, the need to develop faster and more accurate methods for detecting mobile malware has emerged. In addition, the use of artificial intelligence techniques has come to the forefront due to changes in both versions and methods. In this study, machine learning and deep learning methods were examined in order to detect malware in mobile software developed for Android which is the most used platform. The methods examined were tested and compared using the data set produced using the features of static analysis method of Android software. The results obtained are presented in comparison with both accuracy rates and ROC curve. As a result of this study, it is seen that deep learning based systems have better accuracy.

Keywords: Android, Malware, Deep Learning, Machine Learning.

TEŞEKKÜR

Tez çalışmam sırasında tüm sabırlarıyla ve inançlarıyla beni destekleyen aileme, teknik konuda ve bu süreçteki destekleri için Esra N. Yolaçan'a ve her seferinde beni motive eden benimle beraber endişelenen arkadaşlarıma çok teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

ÇİZELGELER DİZİNİ

Çizelge

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
DBN	Deep Belief Networks
CNN	Convolutional Neural Network
LR	Logistic Regression
MLP	Multi Layer Perceptron
SVM	Support Vector Machine
NB	Naïve Bayes
LDA	Linear Discriminant Analysis
RF	Random Forest
AB	Adaptive Boosting
BA	Bootstrap Aggregating (Bagging)
DT	Desicion Trees
RBM	Restricted Boltzmann Machine
FPR	False Positive Rate
TPR	True Positve Rate
TP	True Positive
FN	False Negative
FP	False Positive
TN	True Negative
ROC	Reciever Operator Characteristics

1. GİRİŞ VE AMAÇ

Teknolojinin ilerlemesine paralel olarak akıllı telefon kullanımındaki artış, internetin ve mobil uygulamaların hızlı bir şekilde yaygınlaşmasını beraberinde getirmektedir (Wielen, B., 2018). Dünya çapında popüler bir işletim sistemi olan Android ise günümüzde cep telefonları, tabletler, televizyonlar gibi mobil ve gömülü cihazlar için mobil işletim sistemi pazarındaki ana aktör haline gelmiştir. Android direktörü Stephanie Cuthbertson'a göre, bugün dünya çapında iki buçuk milyardan fazla aktif Android cihazı bulunmaktadır. Statista'nın Ocak 2018'den Ocak 2020'ye kadar dünya çapında Android mobil işletim sistemi sürümlerinin pazar payı istatistiklerini paylaştığı rapora göre Ocak 2020'de Pie 9.0 Android sürümünün dünyadaki tüm mobil Android cihazlar arasında yüzde 42.72'lik bir paya sahip olduğu görülmektedir (S. O'Dea, 2020). Statista'nın paylaştığı raporda 2019'un dördüncü çeyreğinden itibaren, Android kullanıcıları 2.57 milyon uygulama arasından seçim yapabildiği ve Google Play'i en fazla sayıda uygulama içeren uygulama mağazası olduğu görülmektedir (J. Clement, 2020). Google, en güvenilir platformlardan biridir ve günlük olarak milyonlarca kullanıcı tarafından kullanılmaktadır. Bu popülerlik nedeniyle, uygulama geliştiriciler tarafından en çok tercih edilen platformdur. Bu nedenle Play Store'daki uygulama sayısında bir artış vardır. AppBrain tarafından yapılan araştırmaya göre, bu sayının 2022 sonuna kadar daha da artması beklenmektedir.

Mobil cihazlar ve üzerlerinde çalışan uygulama sayısındaki dramatik artış, güvenlik tehditlerini de beraberinde getirmektedir. Bu sebeple, mobil iletişim kanallarını ve hizmetlerini istismar eden güvenlik açıklarının sayısında ve çeşitliliğinde artış yaşanmaktadır. Android güvenlik zafiyetlerinin yaşanma riski en yüksek sistemlerden biri olma özelliğini de taşımaktadır. McAfee firmasının 2018 1. çeyrek raporuna göre; 2017'nin üçüncü çeyreğinde 16 milyon kötü amaçlı yazılım enfenksiyonu tespit edildiği ve şirketin geçen yılki tehdit raporuna göre bir önceki yılın aynı dönemine göre neredeyse iki katına çıktığı gözlenmektedir. Rapora göre; 2018'de küresel siber suçun 600 milyar dolara mal olacağı tahmin edilirken, dünya nüfusunun büyük çoğunluğunun tercih ettiği mobil cihazlar için mobil kötü amaçlı yazılımların yılı olabileceğini öne sürülmüştür. Raporda, mobil cihazlar için yayılan kötü amaçlı yazılımların artış hızının "endişe verici" olduğu

belirtilmektedir. Örneğin, bankacılık uygulamaları üzerinden Truva atları milyonlarca dolarlık zarara neden olmaktadır. McAfee araştırmacıları, kullanıcılar tarafından çevrimiçi mağazalara sızan dolandırıcılık ve şifreleme reklamlarına tıklanarak çok daha fazla istismar yaşanacağını ön görmektedir (Anonim, 2018). McAfee firmasının 2018 2. çeyrek raporuna göre; ikinci çeyrekte yeni mobil zararlı yazılım örneklerinin sayısı yüzde 27 artmıştır. Bu çeyreğin, üst üste artış yaşanan ikinci çeyrek olduğu belirtilmiştir. Bölgesel mobil kötü amaçlı yazılım oranlarına göre Güney Amerika yüzde 14 oran ile en fazla etkilenen bölge olmuştur. Toplam mobil kötü amaçlı yazılım sayısı yıllara göre dramatik bir artış göstermektedir (Anonim, 2018). Kötü amaçlı uygulamaların sayısındaki artış büyük bir sorundur tespit edilmeli ve önlenmelidir.

Geliştiriciler tarafından performans ve tasarımın ön plana alındığı mobil uygulamalarda güvenlik çoğu zaman geri plana atılabilmektedir. Güvenli kod yazımına dikkat etmemekten kaynaklanan açıklar kötü niyetli kişiler tarafından kullanılabilmektedir. Bunun yanında Android uygulama marketleri çeşitli geliştiriciler tarafından yüklenen uygulamaları detaylı güvenlik taramalarına tabi tutmadan yayınlamakta ve kullanıcı uygulamayı yüklerken gereken izinlerden kendisi sorumlu tutulmaktadır. Meşru kullanımlar için yüklenen birçok uygulama, telefonunuzdan kişisel bilgilerinizin ve kişi bağlantılarınızın sızdırılması, finansal bilgilerin çalınması gibi kötüye kullanılmalara neden olabilmektedir. Bu nedenle Android işletim sistemleri için geliştirilen uygulamalar için güvenlik üzerinde durulması gereken önemli bir bileşen olmaktadır. Ayrıca, kullanıcının bir uygulamanın veri ve cihaz güvenliği üzerindeki etkisinin farkında olması gerekir. Kötü amaçlı yazılım önleme çözümleri, zorunlu işletim sistemi güvenlik modeli nedeniyle kötü amaçlı yazılım denetimleri gerçekleştirmek için yeterli değildir. Güvenli bir sistem için istenen özellikler tipik olarak iyi bilinen üç kategoriye ayrılır: gizlilik (confidentiality), bütünlük (integrity) ve kullanılabilirlik (availability) (CIA). CIA üçgeni, bilgisayar güvenliğinin başlangıcından beri bilgi güvenliği için endüstri standardını oluşturmaktadır. Güvenlik tehditleri geleneksel olarak CIA üçgeninin bir veya daha fazla köşesine yönelik bir tehditle kendini gösterir.

Kötü amaçlı yazılım, CIA modelini tehdit eden herhangi bir istenmeyen yazılımı tanımlamak için kullanılan genel bir terimdir. Genellikle bir bilgisayar sistemine zarar vermek veya istenmeyen diğer işlemleri yapmak için tasarlanmış bir yazılım olarak

tanımlanır. Kötü amaçlı yazılım tespit sistemleri önleyici ve ajan sistemler olarak sınıflandırılabilir. Önleyici bilgi güvenliği analiz teknikleri, yazılım ve sistemleri kamuya açmadan önce yapılan analiz tekniğidir. Bu kategori, zayıf yönlerin kodlarını analiz etmeye yönelik geliştirici araçlarını içerir. Bilgi güvenliği tehdidi tespit mekanizmaları, iyi huylu sistemlere bulaşan ya da potansiyel olarak zarar verecek olan kötü amaçlı yazılımların yerini belirlemeye yöneliktir. Android durumunda, önleyici tedbirler, yayınlanmadan önce Android uygulamalarının analizini içerir; Oysa, ajan önlemler arasında internetten indirilebilecek, halka açık olan kötü amaçlı uygulamaların bulunması yer almaktadır. Kötü amaçlı yazılım önleme mekanizmasının geliştirilmesi için Android sürümlerinde bildirilmiş veya çalıştırılabilecek zararlı faaliyetler hakkında genel bilgilerin bilinmesi önemlidir.

Cihazın kök erişimini elde etmek için açık kaynak kodlu bir işletim sistemi olan Android'in keşfedilen çekirdek(kernel) güvenlik açıkları kullanılabilir. Cihazın kök erişimi kullanılarak Android bileşenlerine ve tehlikeli izinlere erişilebilir. Kişisel bilgi hırsızlığı, kullanıcılar kötü amaçlı uygulamalara tehlikeli izinler verdiğinde ve bilmeden hassas verilere erişme izin vermiş olmalarından dolayı gerçekleşir. Kötü niyetli uygulamalar sesli aramaları veya SMS'leri izleyerek, kullanıcı bilgisi ya da izni olmadan ses veya video kaydederek kullanıcıları gözetleyebilir ya da aramalar yapabilir. Botnet uygulamaları cihazlara erişim sağlar ve cihazdaki uygulamaları kontrol ederek kötü niyetli faaliyetlerde bulunabilir. Android kötü amaçlı yazılımların yaygın büyümesi göz önüne alındığında, etkili bir şekilde bunları hafifletmek veya onlara karşı savunmak için acil bir ihtiyaç vardır. Android uygulamalardaki tüm bu tehditlerden dolayı kötü amaçlı yazılımların tespiti için önleyici ve ajan sistemlerin geliştirilmesi önemli bir konu haline gelmektedir.

Bu tez kapsamında Android için zararlı yazılım tespit sistem modellemesi tasarlanmaktadır. Bu çalışmada, Drebin, Contaio, Sandroid ve Google Play Store'dan alınan uygulamalar ile veri seti oluşturulmuştur. 6000 Zararlı yazılım ve 6000 zararlı olmayan yazılım olan veri setinde toplan 12000 yazılım bulunmaktadır. Oluşturulan verisetindeki Android uygulamaların statik analizleri yapılarak Android Manifest uygulama izinleri, API çağrıları ve intentlerden oluşan toplam 205 öznitelik çıkarılmıştır.

Öznitelikleri çıkarılarak ön işlemesi yapılan veri seti makine öğrenmesi algoritmaları kullanılarak eğitilmiş ve başarım oranları karşılaştırılmıştır. Literatürde yapılan diğer çalışmalarda kullanılan yöntemlerde öznitelik seçimlerinde bir ya da birkaç bileşen seçilirken bu çalışmada uygulama izinleri, API çağrıları ve intentler öznitelik olarak seçilmiştir. Literatürde en çok kullanılan makine öğrenmesi teknikleri bu çalışmada aynı veri seti üzerinde kullanılmış bunun yanında derin inanç ağları gibi derin öğrenme yaklaşımları da kullanılarak algoritmaların başarım oranları karşılaştırılmıştır. En yüksek başarım oranının derin öğrenme yöntemleri kullanılan sistemlerde gerçekleştirildiği görülmüştür.

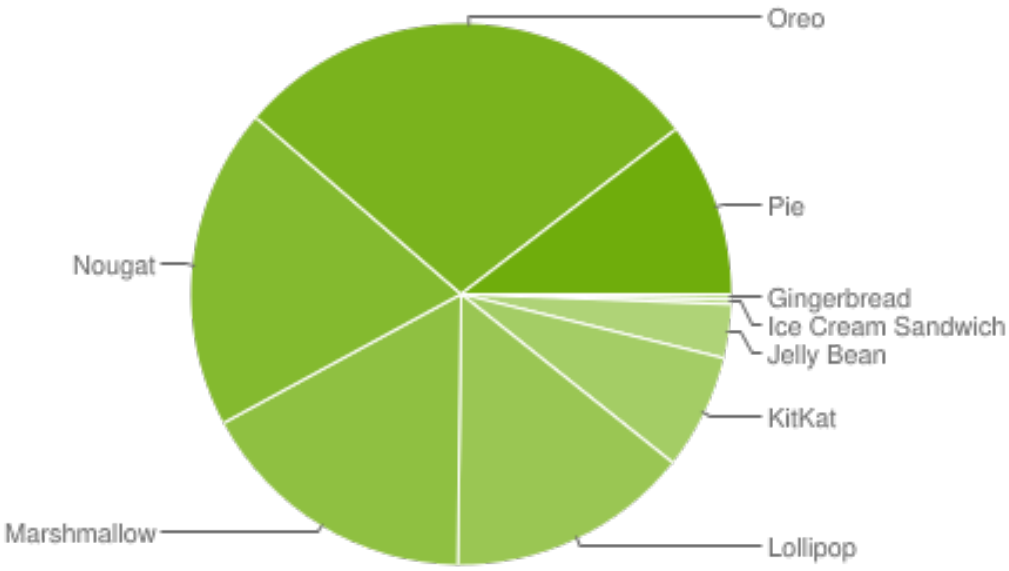
Bu tez çalışmasında, Bölüm 2'de gerekli arka plan bilgilerine odaklanılmıştır. Bölüm 3'de tezin ana gerçeklemesi bulunmaktadır. Bölüm 4'de bulgular, Bölüm 5'de ise sonuç ve öneriler bulunmaktadır.

2. LİTERATÜR ARAŞTIRMASI

Bu tez çalışmasında; Adroid cihazlardaki zararlı yazılım tespiti ile ilgili yapılan çalışmalar ve çalışmalardaki kavramlar bu başlık altında incelenmiştir. Çalışmada odaklanılan ana konular Android işletim sistemi, uygulamalar, Android yazılım analizleri, makine öğrenmesi yaklaşımları ve ilgili çalışmalar olmak üzere başlıklar halinde anlatılmıştır. Android uygulama analizlerinin daha iyi anlaşılması için öncelikle Android işletim sistemi, uygulamalar ve uygulama analizleri üzerinde durulmuştur. Bu tez zararlı yazılım davranışlarına odaklandığı için çalışmada kullanılan makine öğrenmesi teknikleri hakkında temel bilgiler ve literatürde bulunan çalışmalar bu bölümde yer almaktadır.

2.1. Android İşletim Sistemi

Linux çekirdeğini kullanan Android işletim sistemi, açık kaynak kodlu bir işletim sistemidir. Android, dünyada mobil cihazlarda en çok kullanılan işletim sistemidir. Android, Android Inc. firmasında akıllı telefonu geliştirme fikri olan 4 kişi; Andy Rubin, Rich Miner, Chris White, ve Nick Sears tarafından Ekim 2003’de Kaliforniya'da geliştirilmeye başlanmıştır. Şirket 2005 yılında Google tarafından satın alındıktan sonra Andy Rubin liderliğini yaptığı ekip tarafında linux kerneli ile desteklenen mobil cihaz işletim istemi üzerinde çalışılmaya başlanmıştır (Khamlichi, M., 2015). Android ilk sürümü 2008 yılında Android 1.0’ı ile piyasaya sürülmüştür. Android işletim sistemi genel olarak, yeni sürümler yeni özellikler, API ve güvenlik eklerinde değişiklikler getirmektedir. Android’in ana API sürümleri bir Cupcake, Lollipop, Oreo gibi tatlı isimleriyle adlandırılır, ancak API düzeyindeki Android 1.0 Android 22, 23 gibi sayılarla kullanımı daha sıktır. Bunun nedeni Lollipop gibi bir isim Android API seviye 22 ve 23 anlamına gelir. Android ekosisteminin platform sürümlerinin dağıtımı, uygulamaların gelişimi için önemlidir. Android platformunun belirli bir sürümünü çalıştıran cihazların Mayıs 2019'dan itibaren dağılım Şekil 2.1 de gösterilmektedir (Anonim, 2019).



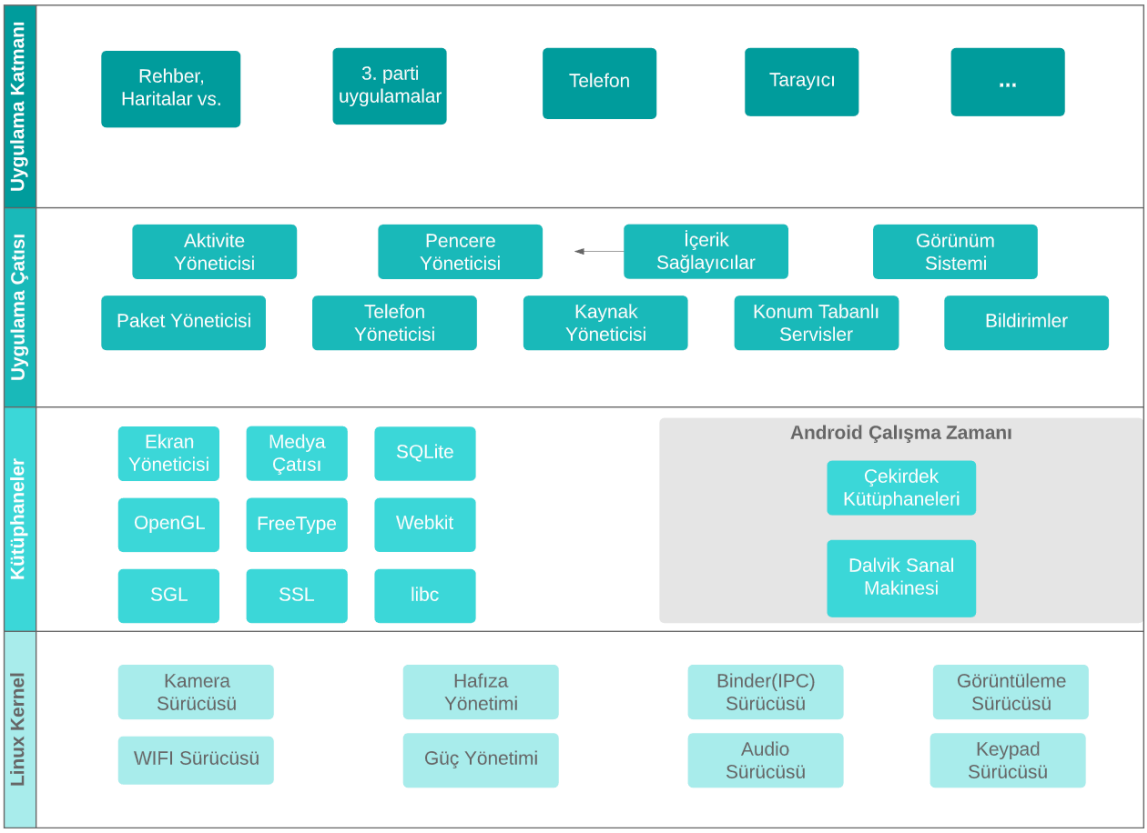
Şekil 2.1. Android Platform Dağılımı

Açık kaynak kodlu bir işletim istemi olan Android yazılımının çoğu Apache 2.0 ile lisanslanmıştır. Projede tercih edilen lisansa uymaya çalışırken, durum bazında ele alınan istisnalar örneğin Linux çekirdek yamaları, GPLv2 lisansı altındadır (Anonim , 2019).

2.2. Android Uygulamalar

Üst katmanları bir programlama dilinde yazılmış, Android olarak da bilinen cihazlar için bir işletim sistemidir. Android işletim sitemi, uygulamalar, çalışma zamanı ortamı, ara katman yazılımı, hizmetler ve kütüphanelerden oluşan bir yazılım yığını şeklinde yapılandırılmıştır. Alt katmanda çalışan Android işletim siteminin temeli olan Linux kernel, wifi ve ses gibi donanımlar için ağ ve aygıt sürücüleri (driver) sağlamanın yanı sıra, bellek, işlem ve güç yönetimi sağlayan donanım ile üst katmalar arasındaki katmadır. İkinci katmanda Android Çalışma Zamanı (Android Runtime – ART) ve yerel kütüphaneler bulunur. Yerel kütüphaneler C/C++ dilinde yazılmış Webkit, OpenMAX AL, OpenGL kütüphaneleridir. Android çalışma zamanı (ART) 4.4 sürümü ile gelmiş ve 5.0 sürümü ile Dalvik'in yerini almıştırDalvik sanal makinası uygulama çalıştığı zaman uygulamayı derlemektedir.ART ise Android uygulaması yüklendiğinde uygulamayı derler. Android üçüncü katmanda OpenGL gibi kütühaneleri uygulamaların kullanımına sunmak için Java API'leri sağlamaktadır.

Android uygulamalar ise yazılım yığınının tepesinde bulunan yazılımlardır. Android yazılım yığını Şekil 2.2 de gösterilmektedir.



Şekil 2.2. Android Mimarisi

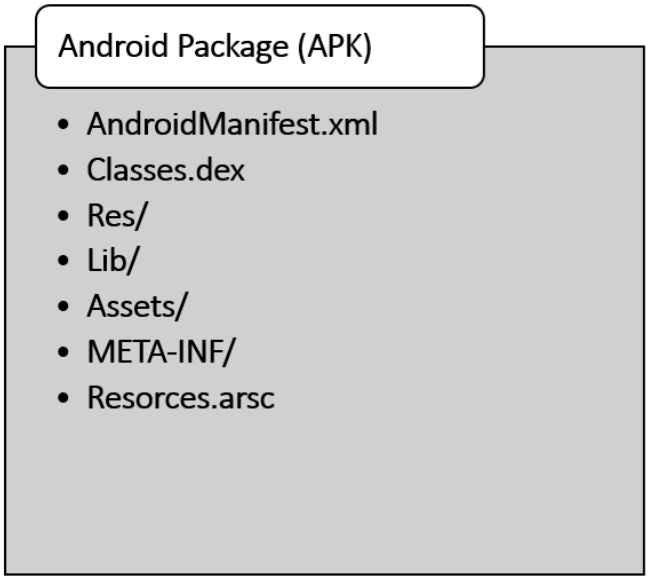
Bir uygulama geliştiricisi Android uygulaması oluşturabilir ve bunu Google Play Store üzerinden yayınlayabilir. Android'in açık kaynak kodlu olması nedeniyle, alternatif pazar yerleri veya doğrudan bağlı bir cihaz aracılığıyla uygulamayı yüklemek mümkündür. Mobil uygulamaları bir Android cihaza aktarmak ve kurmak için APK (Android Package) dosya formatı kullanılır. Bir Android uygulaması, temelde birkaç farklı dosya ve klasör içeren uygulama geliştiricisinin özel anahtarı ile imzalanmış olan sıkıştırılmış bir zip dosyasıdır. Uygulamanın Dalvik bayt kodunu (Java kaynak kodundan derlenmiştir), uygulamanın ihtiyaç duyduğu verileri (resim, ses vb.) ve uygulamanın yapısını ve uygulamanın gerektirdiği izinleri açıklayan bir manifest dosyasını içerir. Android uygulamalarının uygulama geliştiricisinin özel anahtarıyla imzalanmış olması, uygulamaların yalnızca aynı geliştirici tarafından imzalanan kodla güncellenebilmesini ve aynı anahtarla imzalanan uygulamaların izinleri ve UID'yi paylaşma olanağına sahip olmasını sağlar. Uygulama imzası, uygulama bütünlüğünü ve geliştiricinin itibarını

sağlamak için Google ile üçüncü taraf geliştiriciler arasındaki güven noktasını oluşturur. Uygulama imzalama prosedürü, uygulamayı benzersiz bir UID atayarak yalıtılmış bir sanal alana yerleştirilmektedir. Bir A uygulamasının sertifikası cihazda önceden kurulmuş bir B uygulamasıyla eşleşiyorsa, Android aynı UID'yi A ve B uygulamalarına atar; Bu istenmeyen paylaşım, kötü niyetli geliştiriciler iki sertifika üretebileceğinden kötü amaçlı yazılım geliştircileri tarafından kullanılabilir. Kötüye kullanımları önlemek için geliştircilerin sertifikalarını gizli tutmaları önerilir (Faruki vd., 2014). Android uygulamaları Java, Kotlin veya C ++ kullanılarak yazılabilir. Her uygulama kendi sanal makinesinde izole bir ortamda çalışır.Uygulamanın Bluetooth, GPS veya kişi listesi gibi belirli özelliklere erişmek için izin sistemi kullanılmaktadır. Uygulama içerisinde belirli özelliklere erişmek için bir izin listesinin beyan edilmesi zorunludur. Android 6.0' dan önce (API seviye 26) uygulamayı yüklemek için kullanıcı tarafından izinlerin verilmiş olması gerekmekteydi. API 23 seviyesinden bu yana, çalışma zamanı izinleri kullanılarak izin sistemi değiştirilmiştir. Yani uyulama yüklendikten sonra uygulama izinleri gerektiren işlemler gerçekleştirileceği zaman kullanıcı tarafından izinlerin onaylanması beklenmektedir. Çalışma zamanı izinleri üç gruba ayrılır: normal, tehlikeli, imza izinleridir. Normal izinlerin kullanıcı, sistem uygulaması veya cihaz üzerinde minimum risk taşıyan izinlerdir ve sistem tarafından otomatik olarak verilir. Tehlikeli izinler, özel verilere ve cihazın önemli sensörlerine erişebilmeleri nedeniyle yüksek risk grubuna girer. Bir kullanıcı, kurulum sırasında tehlikeli izinlerin yüklenmesini kabul etmelidir. İmza izinleri, yalnızca izin talep eden uygulamaların aynı geliştirici sertifikasıyla imzalanması durumunda kurulum sırasında otomatik olarak verilir (Anonim, 2019).

Android uygulamalar bileşenlerden oluşmaktadır. Bir Android uygulamasının dört temel bileşenini içerir. Bunlar Aktiviteler (Activities), Servisler (Services), İçerik Sağlayıcılar (Content Provider) ve Yayın Alıcıları (Broadcast Receiver)’dır. Her bileşen belirli bir amacı yerine getirmek için kendi yaşam döngüsü ile birlikte gelmektedir. Aktiviteler, Android uygulamaların kullanıcı ile etkileşimini sağlayan arayüzler olarak adlandırılabilir. Servisler, Android uygulamaların arka planda çalışan verileri işlemek veya ağ verilerini çekmek için kullanılan bileşenlerdir. Uzun süre çalışabilir ve kullanıcı arayüzü gerektirmez. İçerik sağlayıcılar, veri ve veri tabanı yönetimini sağlayan bileşendir. Bu bileşen, uygulamanın bir API üzerinden verileri okunmasına veya yazılmasına izin verir. Yayın Alıcıları, işletim sistemi ile uygulamalar arasındaki iletişimi sağlayan bileşendir.

İşletim sistemi, ekran durumu gibi durum mesajlarını bildirmek için yayın alıcısını kullanır. Bileşenlerin birbirleriyle iletişim kurması veya sistemdeki diğer uygulamalarla etkileşime girmesi Intent adı verilen senkronize olmayan mesajlaşma yoluyla sağlanır (Anonim ,2019).

Android APK, Şekil 2.3 de gösterildiği gibi **META-INF/ , .lib/, .res/ , assets/ , AndroidManifest.xml, classes.dex, resources.arsc** bileşenlerinden oluşmaktadır.



Şekil 2.3. Android APK

AndroidManifest.xml, uygulama hakkında önemli bilgiler içerir. XML olarak biçimlendirilmiştir ve uygulama adı, uygulama sürümü veya uygulamanın hedef API düzeyi gibi temel bilgileri içerir. Ayrıca, uygulama için uygun olan minimum API'yi de içermektedir. Android cihazdaki API uygulamanın hedef minumun API'dan daha düşükse uygulama cihazda çalışmaz. Manifet ayrıca uygulamanın gerektirdiği izinleri de içerir. (Android, 2019). *Classes.dex* dosyası Android uygulamasının Dalvik sanal makinesi tarafından anlaşılabilir bayt kodunu içerir. Bir aktivite için kullanılan arayüz elemanları, örneğin metin veya resim alanları, butonlar XML'ler içerisinde tanımlanır. Tüm bu kaynaklar ise *res* klasöründe bulunur. Uygulama içerisindeki UI öğeleri ile program kodu arasındaki bağlantı özel id'ler ve Android R sınıfı tarafından gerçekleştirilir. R sınıfı ile ilgili detaylar *resources.arsc* dosyasında saklanır (Anonim, 2019). *Assets* klasörü, uygula-

ma üzerinden AssetManager tarafından erişilebilen ek dosyaları içeren dizindir. *Lib* işlemcinin bir yazılım katmanına özgü derlenmiş native kodu içeren klasördür. *META-INF* dizini APK'deki dosyaları için hash toplamları (hash sums) ve dijital imza gibi bilgiler içerir.

2.3. Uygulama Analizleri

Bu bölüm Android uygulamalarını analiz etmenin temellerini ele almaktadır. Genel olarak, uygulamaların program kodu analizi statik, dinamik ve bulguların kombinasyonu olan hibrit analizle uygulamanın davranışı hakkında bilgi almak için yapılır. Genellikle statik analiz, bir programın kaynak kodunu girdi olarak alan ve çalıştırmadan kodu inceleyen, kod yapısını kontrol ettikten sonra sonuçları döndüren bir araçtır. Statik analizin temel avantajı, daha az kaynak gerektirmekle birlikte çoğu zaman tüm kodu kapsamasıdır. Dezavantajları, dinamik kod yüklemesinin bulunmaması ve program kodunun gizlenmesi ile ilgili sorunların bulunmasıdır. Dinamik analiz, çalışma zamanı sırasındaki davranışını incelemek için uygulamayı izole edilmiş bir ortamda çalıştırır. Çalışma sırasında kaynakların uygulanması genellikle maliyetlidir, ancak API çağrıları, ağ trafiği ve diğer ölçümler toplanabilir. Dinamik analiz, kod kapsamından yoksundur, çünkü analiz ortamındaki belirli koşulları karşılayan kodlar yürütülür.

Bu çalışmada statik analiz yöntemleri kullanılmıştır. Android uygulamalarının statik analizini yapabilmek için uygulamalar üzerinde tersine mühendislik işleminin yapılması gerekmektedir. Tersine mühendislik kullanılarak java dosyalarının içerikleri görüntülenebilmekte ve zararlı kod parçacıkları bulunabilmektedir. Android ortamının farklılıkları nedeniyle, çoğu statik analiz teknikleri Android için yeniden özelleştirilmiştir. Geleneksel statik analiz teknikleri aşağıda açıklanmaktadır.

Giriş Noktası Analizi (Entry Point Analysis) bir programın nereden başlanarak çalıştırılacağını belirler. Java’da program main method olarak belirtilen tek bir noktadan başlar. Android uygulamalarının ise bir main metodu yoktur. Her uygulama, çalışma zamanında çağrılan activity denilen giriş noktalarını içerir. Activity’lerin kendi içerinsinde bir yaşam döngüsü vardır ve uygulama çalıştığında onCreate(), onStart(), onResume(),

onPause(), onStop(), onDestroy() fonksiyonları çalıştırılır. Birkaç giriş noktası bulunması nedeniyle bu analiz Android uygulamar için zor bir analiz şeklidir (Schmeelk vd., 2015).

Ulaşılabilirlik analiz (Reachability Analysis) incelenmekte olan kodun statik bir temsili üzerinde soyut yorumlar yapmak için önemli bir statik analiz tekniğidir (Schmeelk vd., 2015) . Nikolic ve Spoto (2014) göre bir program değişkeninin v'den başka bir program değişkeni olan w'ye ulaşılabilirliğinin mümkünlüğünü belirleyen analiz tekniğidir. Ulaşılabilirlik analizinin doğruluğuna dayanan birden fazla analiz türü vardır. Yan Etki Analizi (Side-effects Analysis), Kusur Analizi (Taint Analysis), Uzunluk Analizi (Path-Length Analysis) vb. bunlardan birkaçıdır.

İşaretçi analizi (Pointer Analysis) belirli özellikler için bellek referanslarının analizidir. Bu analiz hangi işaretçilerin birlikte aynı yığın belleğine erişebildiğini kontrol eder. Örtüşen veri yapılarına bağlı olabilecek olası değişkenleri izler, bir işaretçi değişkenin çalışma zamanında point edebileceği nesneleri hesaplar (Schmeelk vd., 2015).

Veri akışı analizi (Data-Flow Analysis), bir program yürütülürken değişkenlerin değerlerinin zaman içinde nasıl değiştiğini analiz etmek için kullanılan bir tekniktir. Bir program içindeki veri akışını doğru ve kesin olarak yakalamanın birden fazla statik yolu vardır. İşlemler arası veri akışı analizi, genellikle çağrı grafiği olarak adlandırılan bir teknik aracılığıyla prosedürler arasındaki değerleri analiz eder. Bağlama duyarlı veri akışı analizi, bir işlev çağrısının hedefini analiz ederken çağıran bağlamı dikkate alan bir süreçler arası analizdir. Akışa duyarlı veri akışı analizi, bir programdaki ifadelerin sırasını dikkate alır (Schmeelk vd., 2015).

Çağrı diyagramları (Call-Graph Construction), program içerisindeki tüm fonksiyon çağrılarının statik diyagramlarıdır. Java'da bir program genellikle main method olarak belirtilen tek bir noktadan başlar. Main metodundan tüm fonksiyon çağrıları izlemek ve bu bağlamda, bazı işlem gerçekleştiren tüm kodların bir listelenebilir. Android de birden fazla giriş noktası bulunur. Bu nedende Android için, birden çok "arama" grafiği ile statik olarak analiz edilebilir (Schmeelk vd., 2015).

Statik analiz hızlı ve etkili bir tekniktir. Çeşitli kod karıştırma teknikleri statik analiz yönteminin dezavantajları arasındadır. Yukarıda bahsedilen statik analiz tekniklerinin kullanıldığı imza tabanlı, izin tabanlı, dalvik-bayt kodu analizi gibi Android statik analiz yaklaşımları bulunmaktadır. İmza tabanlı yaklaşımlarda zararlı yazılımlardan toplanan imza verileriyle uygulama verilerinin karşılaştırılması ile yapılmaktadır. İmza tabanlı yaklaşımlar, bilinen zararlı yazılımları kolayca tespit edebilmelerine rağmen, mevcut bir modeli olmayan zararlı yazılımları tespit edememektedir. İzin tabanlı yaklaşım AndroidManifest.xml dosyasında bulunan izinlerin çıkarılmasıyla yapılan analiz yaklaşımıdır. Dalvik-Bayt kod analizi; sınıflar, metotlar ve API çağrıları gibi uygulamanın davranışını anlamayı sağlayacak bilgilerin elde edilmesine yönelik yapılan analiz şeklidir. Bu bilgileri elde etmek için Dalvik Bytecode' unun Java Bytecode'una dönüştürülmesi (decompiler) gerekmektedir. Dalvik-Bayt kod analizinde Java kodu üzerinden kontrol akışı analizi ve veri akışı analizi gerçekleştirilmektedir. Statik analiz sonucuda çıkarılan özniteliklere Intent nesneleri, stringler, Manifest izinleri, donanım bileşenleri, API çağrıları örnek olarak verilebilir. Android uygulamalarını analizini yapmak için birkaç kütüphane ve araç kiti bulunmaktadır. Apktool, dex2jard, Smali/Baksmali, dexdump, Androguard bunlardan birkaçıdır.

2.4. Makine Öğrenmesi

Yapay zeka alanında makine öğrenmesi yıllardır en çok kullanılan yöntemlerin başında gelmektedir. Günümüzde makine öğrenme modelleri; örneğin çevrimiçi mağazalarda öneri sistemleri, kredi kartı şirketlerinde sahtekarlık tespiti veya hastanelerde tıbbi teşhis için kullanılmaktadır. Makine öğrenmesi, özellikle derin öğrenme, ses tanıma ve yüz tanıma gibi alanlarda kullanılır. Bugün olduğu gibi birçok bilimsel topluluk, araştırmaları ve bilimsel ilerlemeleri için makine öğrenmesini kullanmaktadır. Makine öğrenmesinde sistem eğitilirken ne kadar fazla veri olursa o kadar iyi eğitim gerçekleşmektedir. Verilerin artışıyla birlikte verinin işlenmesi, özniteliklerin çıkarılması ve sistemlerin eğitilmesi daha karmaşık bir hale gelmektedir. Teknolojinin gelişmesiyle birlikte işlemci gücü yüksek olan cihazların sayısı artmıştır. Literatürdeki çalışmalarda Android uygulamaların dinamik, statik ya da hibrit analiz tabanlı zararlı yazılım tespit sistemlerinde çoğunlukla makine öğrenmesi algoritmaları kullanılmaktadır. Son yıllarda

derin öğrenme ile yapılan çalışmaların artışı ile birlikte en iyi sonuçlara derin öğrenme yöntemleriyle ulaşıldığı görülmektedir.

Makine öğrenmesi yaklaşımları denetimli (Supervised Learning), denetimsiz (Unsupervised Learning), yarı denetimli (Semi-Supervised Learning) ve pekiştirmeli öğrenme(Reinforcement Learning) olarak dört gruba ayrılabilir (Moubayed vd., 2018). Supervised learning kavramı literatürde hem denetimli hem de gözetimli öğrenme olarak geçmektedir. Bu çalışmada denetimli öğrenme şeklinde kullanılacaktır. Bu tez çalışmasında ağırlıklı olarak denetimli öğreme yaklaşımları kullanılmıştır. Aşağıda bu tez çalışmasında kullanılan makine öğrenmesi yöntemleri açıklanmaktadır.

2.4.1. Denetimli Öğrenme

Denetimli öğrenme yönteminde, etiketlenmiş eğitim verilerine dayanarak bir fonksiyonun çıkarıldığı bir makine öğrenme yaklaşımıdır. Eğitim verileri, her biri bir çift (x, y) olan ve buradaki x' in bir girdi vektörü ve y'nin çıktı değeri olduğu bir grup eğitim örneğinden oluşur. Algoritma, gelecekteki bilinmeyen girdileri eşlemek için kullanılabilecek bir fonksiyon üretir (Moubayed vd., 2018). Denetimli öğrenme algoritmaları genellikle iki ana kategoriden birine girer: regresyon algoritmaları veya sınıflandırma algoritmalarıdır. Sınıflandırma algoritmalarıdır bir verinin hangi sınıfa ait olduğunu tahmin etmek için kullanılan algoritmalardır. Regresyon algoritmalarında ise iki değişken arasında bir ilişki aranır ve ona göre bir model oluşturulur. Çalışmada kullanılan denetimli öğrenme algoritmaları aşağıda sıralanmaktadır.

Lojistik Regresyon (LR- Logistic Regression): Sınıflandırma problemlerini çözmekte kullanılan Lojistik Regresyon bağımlı değişken ile bağımsız değişken(ler) arasındaki ilişkiyi açıklamak için kullanılır. Modelin amacı, bağımlı değişkenin tahmini değerlerini olasılık olarak hesaplayarak olasılıklara uygun olarak sınıflandırma yapmaktır.

Destek Vektör Makinesi (SVM- Support Vector Machine): İstatiksel öğrenme teorisine dayanan bir denetimli öğrenme algoritması olan Destek Vektör Makinesi, N-boyutlu bir uzayda (N- özelliklerin sayısı) veri noktalarını belirgin bir şekilde sınıflandıran hiper düzlemi elde elde ederek verileri birbirinden en uygun şekilde ayırmak için kullanılır.

Naif Bayes (NB- Naïve Bayes): Sınıflandırma yaparken özellikler arasında bağımsızlık varsayımına dayanan bir sınıflandırma yöntemi olan Naïve Bayes olasılık ilkelerine göre tanımlanmış bir dizi hesaplama ile, sisteme sunulan verilerin sınıfını tespit etmekte kullanılır.

Doğrusal Ayrımcılık Analizi (LDA- Linear Discriminant Analysis): 1936 yılında R. A. Fischer tarafından geliştirilmiş olan Doğrusal Ayrımcılık Analizi, sınıfları birbirinden ayırmak için, sınıfların dağılımını ve ortalama değerleri arasındaki farklılığı kullanır.

Rastgele Orman (RF- Random Forest): Denetimli bir sınıflandırma algoritması olan Rastgele Orman, hem regresyon hem de sınıflandırma problemlerinde kullanılabilmektedir. Rastgele Orman algoritması, düğümleri bölme, kök düğümü bulma gibi işlemleri torbalama yöntemiyle yapan karar ağaçlarından oluşur. Rastgele Orman, tüm değişkenler arasından en iyi dalı kullanarak her bir düğümde rastgele olarak seçilen değişkenler arasından en iyisini kullanarak düğümleri dallara ayırır. Sonrasında rastgele özellik seçimi kullanılarak ağaçlar geliştirilir. Geniş bir çeşitlilikle sonuçlan bu yöntem genellikle daha iyi bir model elde edilmesini sağlar.

K en yakın komşu (K-NN- K Nearest Neighbors): Bir veri noktasının hangi gruba en yakın olduğunu ve o gruba ilave edilmesinin ne kadar muhtemel olduğunun tahmin eden bir sınıflandırma algoritmasıdır. Algoritmadaki K değerinin anlamı bakılacak eleman sayısını ifade etemktedir. Yeni bir değer geldiğinde en yakın K kadar elemana bakılarak mmesafeler hesaplanır ve yeni değer kümeye eklenir. Uzaklık hesaplama işleminde k-means, öklid uzaklığı, manhattan uzaklığı, Minkowski ve Hamming gibi mesafe hesaplama yöntemleri kullanılabilir.

Adaptif Yükseltme (AB- Adaptive Boosting): Bu yötem bir öğrenicinin, kendinden önce gelen güçsüz öğrenicileri kullanarak öğrenicilerini düzeltmesi işlemine dayanır. Böylece güçsüz öğrenicilerin verisiyle daha iyi bir öğrenicinin oluşturması sağlanır.

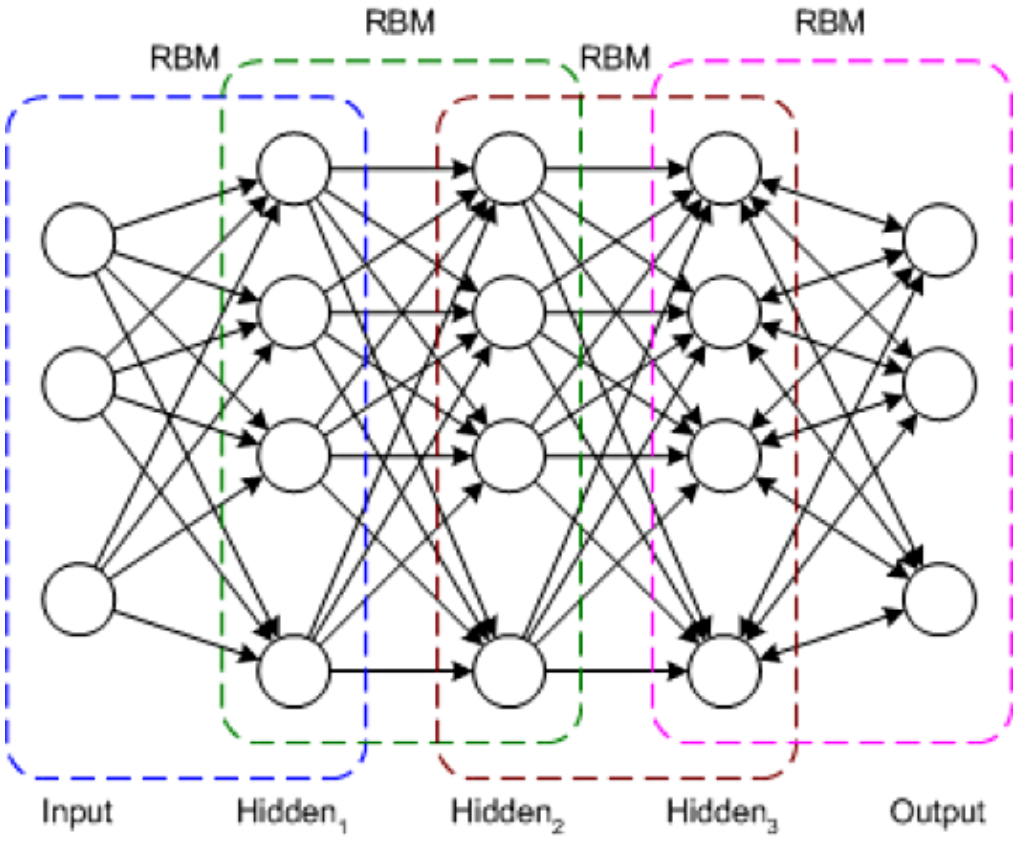
Torbalama (BA- Bootstrap Aggregating (Bagging)): Bu yöntem varyansı azaltarak makine öğrenme algoritmalarının kararlılığını ve doğruluğunu artıran bir meta algoritmadır. Meta algoritmalar, bir sınıflandırıcı yerine birden çok sınıflandırıcı üreterek

üretilen sınıflandırıcıların tahminlerinden alınan oylar ile yeni veriyi sınıflandıran algoritmalarıdır.

Karar Ağaçları CART (DT- Desicion Trees (CART)): Ağaç tabanlı algoritmalar makine öğrenmesinde en çok kullanılan denetimli (supervised) öğrenme algoritmalarıdır. Karar ağacı, ağaç yapısı şeklinde sınıflandırma veya regresyon modelleri oluşturur. Yaptığımız çalışmada karar ağaçları oluşturmak için CART (Classification and Regression Trees) kullanılmıştır.

Kısıtlı Boltzmann Makinesi (RBM- Restricted Boltzmann Machine): Girdiyi işleyerek işlenilen girdi üzerinde olasılık dağılımlarını öğrenip bu girdinin iletilip iletmeyeceği konusunda rastlantısal olarak kararlar alabilen bir yapay sinir ağıdır.

Derin İnanç Ağları (DBN- Deep Belief Networks), DBN, RBM gibi denetlenmeyen ağların bir birleşimi olarak tanımlanmaktadır. Her bir alt ağın gizli katmanı bir sonraki ağın görünür katmanıdır. DBN gerçek gücünü, bir dizi RBM’ ni kullanılarak birçok katmandan oluşan derin bir inanç ağı oluşturulur. Bir DBN’ de, her tabaka bir dizi ikili veya gerçek değerli birim içerir. İki bitişik katmanın aralarında tam bir bağlantı seti bulunur, ancak aynı katmandaki iki birim birbirine bağlanmaz (Khamlichi, M., 2015). Hinton ve vd. (2006) derin inanç ağlarını eğitmek için etkin bir algoritma önermiş ve her katmanı (en alçaktan en yükseğe doğru) önceki katmanın girişler olarak aktivasyonlarını kullanarak bir RBM olarak eğitmişlerdir. DBN mimarisi Şekil 2.4.’de gösterilmektedir.



Şekil 2.4. DBN Mimarisi (Jones, M., 2017)

Çok katmanlı algılayıcı (MLP- Multi Layer Perceptron): Rumelhart vd. (1985) tarafından geliştirilen MLP, bir dizi girdiden bir çıktı seti üreten, geri yayılımlı bir yapay sinir ağıdır .

2.5. İlgili Çalışmalar

Bu bölümde literatürde bulunan çalışmaların yaklaşımları, kullandıkları veri setleri, araçlar, kullanılan yöntem ve algoritmalar incelenerek genel bir bakış sunulmaktadır.

Android uygulamalarının uygulama izinleri ve API çağrıları kullanılarak 1200 zararlı yazılım ve 1200 zararsız yazılım üzerinde yapılan bu çalışmada makine öğrenmesi kullanılarak yazarlı yazılım tespit sistemi geliştirilmiştir (Peiravian vd., 2013). Makine öğrenmesi algoritmalarından J48 Karar Ağacı, torbalama ve destek vektör makineleri

kullanılan çalışmada 96.39 ile 92.36 arasında doğruluk (accuracy) oranları elde edilmiştir. En yüksek sonuç 96.39 ile torbalama yöntemi ile elde edilmiştir.

DroidMat, uygulama izinleri ve API çağrılarını kullanan zararlı yazılım tespit aracıdır (Wu vd., 2012). Çalışmada kullanılan zaralı yazılımlar Contagio (Anonim, 2019) sitesinden alınmıştır. İyi uygulamalar ise GooglePlay resmi Android pazarındaki her bir kategoriden 50 uygulamayı rastgele indirilerek elde edilmiştir. İyi yazılımların Android kötü amaçlı yazılım olarak tanımadığından emin olmak için VirusTotal kullanılmıştır. Toplamda 34 kötü niyetli aileyi içeren toplam 238 Android kötü amaçlı yazılımın ve 30 kategori içeren 1500 iyi huylu uygulama toplanmıştır. Çalışmada K en yakın komşu (KNN - k Nearest Neighbours) algoritması kullanılarak yüzde 97,87 doğruluk (accuracy) oranına sahip Android zararlı yazılım tespit sistemi geliştirilmiştir.

Hou vd. (2016) yaptığı çalışmada Linux çekirdek sistemi çağrılarına dayanarak, ağırlıklı olarak yönlendirilmiş grafikler oluşturmakta ve daha sonra yeni bilinmeyen Android kötü amaçlı yazılım tespiti için grafik tabanlı özelliklere dayanan bir derin öğrenme çerçevesi sunulmaktadır. Çalışmada Comodo’ dan alınan 1.500 iyi yazılım, 1.500 kötü amaçlı yazılım olmak üzere toplam 3.000 yazılım kullanılmıştır. Bir Android uygulamasının sistem çağrılarını dinamik analiz yoluyla izlemek için, her zaman uygulama ile etkileşimler gerektirir. Çok sayıda Android uygulaması verildiğinde, hepsini manuel olarak yürütmek mümkün değildir. Android Test Aracı (ADT) Monkey, Android uygulamalarını otomatik olarak yürütmek için yaygın olarak kullanılmasına rağmen, kötü amaçlı yazılım analizi için uygulamanın tüm bileşenlerini yürütmek için yeterli bir yaklaşım değildir. Bu sorunu çözmek için, tüm uygulama kodunun yürütülmesini otomatikleştirmenin bir yolu olarak Component Traversal yöntemini önerilmektedir. Tüm yürütülebilir uygulama bileşenlerini manifest dosyasından bularak, tüm çalıştırılabilir kodun yürütülmesi sağladığı için olabileceğinden daha eksiksiz bir sistem çağrı listesi oluşturula bilinmiştir. Her bir Android uygulamasında, aralarındaki ilişkileri yakalamak için grafik oluşturulacaktır. Her bir grafik düğümü bir Linux çekirdek sistemi çağrısını temsil edecek ve büyüklüğü frekansını gösterecektir. Derin öğrenme mimarisi olarak Stacked AutoEncoders (SAEs) model kullanılarak yüzde 93.68 doğruluk (accuracy) oranı elde edilmiştir.

McLaughlin vd. (2017) yaptığı çalışmada Android uygulamaların statik analiz özelliklerini kullanarak yapılan derin öğrenme tabanlı Android zararlı yazılım tespit sistemidir. Bu çalışma, çözümlenecek bayt kodunun analiz edilecek bir metin olarak ele alınmasıyla, konvolüsyonel sinir ağlarının (Convolution Neural Network- CNN) kötü amaçlı yazılım tespitine uygulanması temeline dayanır. Çalışmada n-gram tabanlı yaklaşım kullanılmış olup, konvolüsyonel ağının kötü amaçlı yazılımın göstergesi olan opcodes dizilerini tespit ederek n-gram gibi imzaları algılamayı öğrenebilmesi şeklindedir. McLaughlin vd. (2017) yaptığı bu çalışmada veri seti, Android Malware Genome Projesinden (AMGP) (AMGP, 2015) alınan 2123 uygulama ve yaklaşık 18.000 Android uygulama içeren McAfee Labs tarafından sağlanan uygulamalardan oluşturulmuştur. Veri setinde 9268 iyi uygulama ve 9902 kötü amaçlı uygulama bulunmaktadır. Önerilen derin otomatik kodlayıcıya dayalı modelde 98.10% doğruluk (accuracy) oranında başarı sağladığı görülmektedir.

Droiddelver .smali dosyalarından çıkarılan API çağrılarının (statik analiz özelliği) analizi ile Derin İnanç Ağı (DBN) kullanılarak yapılan zararlı yazılım tespit sistemidir (Hou, 2016). Çalışma .smali kodundaki API çağrıları bazı yöntemlere bloklara ayrılır. Kod blokları, yeni bilinmeyen Android kötü amaçlı yazılım tespiti için DBN ve RBM kullanılarak oluşturulan derin öğrenme mimarisiyle eğitilmiştir. Comodo Cloud Security Center'dan alınan yarısı iyi niyetli, diğer yarısı ise kötü amaçlı yazılım olan 5,000 Android uygulaması bulunan veri seti kullanılmıştır. FPR, TPR, doğruluk (accuracy) gibi değerlendirme metrikleri kullanılan çalışmada %96.66 doğruluk (accuracy) oranına sahiptir.

DeepFlow (Zhu vd, 2017) Android uygulamalardaki statik analiz ile elde edilen hassas verileri kullanarak derin öğrenme yöntemi ile yapılan zararlı yazılım tespit sistemidir. Deepflow; bir Android uygulamasında hassas veri akışlarını tanımlamak için aktivite yönetim metotlarına(onCreate(), onStart(), onDestroy()), callback metotlarına ve bunların yanı sıra kaynak çağrılarına da bakarak kötü amaçlı yazılımların kullanacağı hassas verileri elde etmeye çalışır. Çalışmada toplam 323 öznitelik çıkarılmıştır. Kullanılan DBN modeli, bilinen kötü amaçlı yazılım kaynaklarından (örneğin, Genome, VirusTotal ve Drebin) ve Google Play'den taranan iyi yazılımlardan (benignware) tarama yapmak için hiyerarşik olarak bir dizi RBM kullanarak oluşturulmuştur.

Zhang vd. (2016), derin öğrenme modeline dayanan Android platform için kötü amaçlı yazılım algılama yaklaşımı önerilmektedirler. DroidDeep ilk önce Android uygulamalarının davranış biçimini nitelemek için izinler, API çağrıları ve bileşenleri gibi statik bilgileri göz önünde bulundurarak 30.000'den fazla özellik içeren Android uygulamalarından öznitelikleri otomatik olarak çıkarmaktadır. Sınıflandırmada kullanılacak özniteliklerin sayısını etkili bir şekilde azaltabilecek tipik özellikleri öğrenmek için DBN algoritmasını kullanılmıştır. Sistem 4 bölümden oluşmaktadır. Birinci aşama, özniteliklerin Android uygulamalardan otomatik bir şekilde çıkarılmasıdır. İkinci aşama, Android uygulamalardan otomatik bir şekilde çıkarılmış özellikleri çok boyutlu bir vektöre dönüştürülmesidir. Üçüncü aşama, çok boyutlu vektörü girdi olarak alan ve Android kötü amaçlı yazılım tespiti için benzersiz özellikleri öğrenen derin öğrenme modelidir. Sonuncu aşama ise öğrenme özelliklerine dayanan sınıflandırma işlevidir. Sınıflandırma Destek Vektör Makinesi algoritması ile yapılmıştır. 3,986 iyi uygulamalar ve 3,986 kötü amaçlı yazılımdan oluşan bir deneme sonucunda, DroidDeep yüzde 99,4 oranında bir algılama doğruluğu sağlamaktadır.

Shabtai vd. (2014) yaptığı çalışmada dinamik analiz ve makine öğrenmesi yöntemleri kullanılarak lokal öğrenme ile mobil uygulamanın ağ davranışında anlamlı sapmaları algılamak için bir davranış tabanlı anormal algılama sistemi geliştirmişlerdir. Veri setinde 500.000 uygulama mevcuttur. Sitem uygulamaların Android cihazda bulunan istemci ve sunucu bileşenleri sayesinde ağ desenlerini çıkartarak zararlı yazılımları tespit edebilmektedir. İstemci bileşeni cihaza önceden yüklenen uygulamaların dinamik analizlerini yaparak izlemektedir. Sunucu bileşeni ise mobil cihazlar tarafından rapor edilen verileri ile her uygulama için kullanıcıların ortak trafik modellerini temsil eden modeller oluşturmaktadır. Çalışma sonucunda aynı tipteki uygulamaların benzer ağ trafik desenlerine sahip oldukları gözlemlenmiştir.

DroidDetector derin öğrenmeye dayalı Android kötü amaçlı yazılım tespit sistemidir (Yuan vd., 2016). DroidDetector statik analizden elde edilen özellikler ile dinamik analizinden elde edilen özellikleri ilişkilendirip ve öğrenme teknikleri kullanarak kötü amaçlı yazılımları karakterize etmektedir. Bu sayede bir uygulamanın kötü amaçlı yazılım olup olmadığını otomatik olarak algılayabilen çevrimiçi derin öğrenme tabanlı bir

Android kötü amaçlı yazılım tespit motoru (DroidDetector) geliştirmişlerdir. Derin öğrenme mimarisi denetimli (supervised) ve denetimsiz (unsupervised) öğrenme olmak üzere iki aşamadan oluşmaktadır. Derin İnanç Ağı (Deep Belief Network- DBN) mimarisinde Android uygulamalarını daha iyi karakterize etmek için Kısıtlı Boltmann Makinalarının (Restricted Boltzmann Machines - RBM) hiyerarşik olarak inşa edilmiştir. Eğitim için kullanılan 20.000 zararlı olmayan uygulama Google Play Store'dan rastgele taranarak elde edilmiştir. Kötü niyetli uygulama veri seti Contagio (Anonim, 2019) Topluluğu'ndan 500 uygulama, Genome (Anonim, 2015) Projesinden 1260 uygulama toplam 1760 adet kötü amaçlı uygulama toplanmıştır.

McLaughlin vd. (2017) yaptığı çalışmada çözümlenecek bayt kodunun analiz edilecek bir metin olarak ele alınmasıyla, konvolüsyon ağlarının kötü amaçlı yazılım tespitine uygulanması temeline dayanmaktadır. Doğal dil işlemede (Natural Language Process)'de, n-gram olarak bilinen yerel semboller, çeşitli görevler için öznitelik olarak kullanılmıştır. Özetle çalışmada n-gram tabanlı yaklaşım kullanılmış olup Android zararlı yazılımların statik analiz yöntemi ile ele alınacak özniteliklerine odaklanmıştır. Bir Android uygulaması, kod dosyalarını, AndroidManifest.xml dosyasını ve uygulama kaynak dosyalarını içeren sıkıştırılmış bir apk dosyasıdır. Bir kod dosyası, her smali dosyasının tek bir sınıfı temsil ettiği ve böyle bir sınıfın yöntemlerini içerdiği, smali dosyalarına dönüştürülebilen bir dex dosyasıdır. Her yöntem talimatlar içerir ve her talimat tek bir opcode ve çoklu işlenenlerden oluşur. Uygulamanın Dalvik bayt kodunu içeren smali dosyalarını elde etmek için her bir uygulamayı baksmali kullanarak parçalarına ayırır daha sonra her bir yöntemden opcode dizisini ayıklanarak operandlar atılır. Ön işlemenin sonucu olarak, tüm opcode dizileri uygulamanın tüm sınıflarından elde edilmiş olur. Tüm sınıflardan gelen opcode dizileri daha sonra bütün uygulamayı temsil eden tek bir opcodes dizisini olmak üzere birleştirilir. Çalışmada konvolüsyon ağının, kötü amaçlı yazılımın göstergesi olan opcodes dizilerini tespit ederek n-gram gibi imzaları algılamayı öğrenebilmesi amaçlanmıştır.

Çizelge 2.1 de literatürdeki zararlı yazılım tespit sistemleri ile ilgili çalışmaların karşılaştırılması verilmiştir.

Çizelge 2.1. Literatürdeki Android Zararlı Yazılım Tespit Sistemleri

Makale	**Teknik**	**Algoritma**	**Özellikler**	**Veriseti**	**Değerlendirme Metriği**	**Başarı Oranı**
Peiravian, 2013	Statik Analiz	J48 Karar Ağacı, Torbalama ve Destek Vektör Makineleri	Uygulama izinleri ve API çağrıları	2400	Duyarlılık(recall), AUC ve Doğruluk (accuracy)	% 96.39
DroidMat, Wu,2012	Statik Analiz	K en yakın komşu	Uygulama izinleri ve API çağrıları	Contagio	FPR, TPR, AUC, Doğruluk(accuracy), F-measure	% 97,87
McLaughlin, 2017	Statik Analiz	Autoencoders, RBM	opcodes(.dex files feature)	AMGP, McAfee	TPR, FPR, Kesinlik(Precision), Doğruluk(accuracy), Duyarlılık(recall), F skor	% 95.0
Droiddelver	Statik Analiz	DBN	API çağrıları	Comodo	FPR, TPR, Doğruluk(accuracy)	% 96.66
DeepFlow	Statik Analiz	DBN	Hassas kaynaklar(SMS API vb.)	Genome, VirusTotal ve Drebin ve Google play store	Kesinlik (Precision) , Duyarlılık(recall) ve F1 skor	% 95.05

Çizelge 2.1. Literatürdeki Android Zararlı yazılım Tespit Sistemleri(devam)

Zhang (Su,2016)	Statik Analiz	DBN	Uygulama izinleri ve API çağrıları	Google Play Store, AMGP, third-party market in China	Doğruluk(accuracy)	% 99,4
Deep4maldroid	Dinamik analiz	SAEs	Sistem çağrılar	Comodo	FPR, TPR,Doğruluk(accuracy)	% 93.68
DroidDetector	Hibrti Analiz	DBN	Uygulama izinleri API çağrılar ve dinamik davranışlar	Contagio, AMGP, Google Play Store	Kesinlik (Precision), Duyarlılık(recall), Doğruluk(accuracy)	% 96.76
Sheen ve ark. (2015)	İzin Tabanlı Analiz Statik Analiz	Ensemble	Uygulama izinleri ve API çağrıları	AMGP	Kesinlik (Precision), Duyarlılık(recall), F-skor	% 88
McLaughlin, 2017	Statik analiz	Konvolüsyonel Nöral Ağ (CNN)		AMGP, McAfee Labs		% 98.10

3. MATERYAL VE YÖNTEM

Bu tez çalışmasında ele alınan zararlı yazılım tespit yaklaşımı olarak makine öğrenmesi yöntemleri kullanılmıştır. Önerilen yöntemler, kullanılan veri seti, kullanılan araçlar aşağıda maddeler halinde verilmektedir.

3.1. Materyal

Çalışmada kullanılan yazılımlar Google Play Store , Sandroid, Contagio ve Drebin veri setinden alınmıştır. Toplanan yazılımların analizi için dex2jar, AXMLprinter2 vb. araçları kullanılmıştır. Analiz işlemleri yapılan veri setinin eğitimi için pyton proramlama dili ve scikit learn, pandas, keras gibi kütüphanelerden yararlanılmıştır. Kullanılan araçlar, kütüphalere ve programlama dili aşağıdaki alt bölümlerde verilmektedir.

3.1.1. Analiz Araçları ve Kütüphaneler

Dex2jar Android apk dosyasında dex dosyalarını java beyte kodlarına çevirmektedir. AXMLprinter2 manifest izinleri ve intentleri öznitelik olarak çıkarmamızı sağlar. smali / baksmali dex formatı için bir assembler/disassembler çiftidir. Baksmali bir dex dosyası alır ve insan tarafından okunabilir bir dosya üretir ve smali insan tarafından okunabilir dosyayı alır ve bir dex dosyası üretir. Apktool, bir apk paketini açmak ve yeniden paketlemek için daha genel bir süreçtir. Apktool Android uygulamaların apk dosyalarını decompile ederek smali kodlarına dönüştürür. Ayrıca dex formatı için smali / baksmali yapısını kullanır.

3.1.2. Python ve Kütüphaneler

Python, nesne yönelimli, yüksek seviyeli bir programlama dilidir (McKinney, 2012). Diğer üst seviye programlama dillere göre kod okunabilirliği kolaydır. Python, derleyici programa ihtiyaç duymaması ve temiz bir sözdizimine sahip olması tercih edilme nedenlerindendir.

Pythonda, verinin okuması, görselleştirilmesi ve verinin dizilerde tutulması için birçok Python kütüphanesi vardır. Bunlardan bazıları Pandas, NumPy, Matplotlib'dir. Numpy, Python ile bilimsel hesaplama için temel bir kütüphanedir. Güçlü bir N-boyutlu dizi nesnesine , gelişmiş fonksiyonlara ve C / C ++ ve Fortran kodunu entegre etmek için araçlara sahiptir (Anonim, 2019). Sayısal hesaplamalar, numpy kütüphanesi ile basit bir seviyede ve az kod yazacak şekilde yapılabilir. Python programlama dili için açık kaynaklı BSD lisanslı, kullanımı kolay veri yapıları ve veri analiz araçları sağlayan, yüksek performanslı bir kütüphanedir (Anonim, 2019). Bu kütüphane, Csv ve metin dosyalarını açmada ve içerisindeki verileri işleme ve temizlemede efektif şekilde kullanılır. Pandas kütüphanesi "pip install pandas" şeklinde yüklenir ve Python kodu içerisnde kullnamak için "import pandas as pd" ile paket koda eklenir. Matplotlib, platformlar arasında çeşitli basılı formatlarda ve etkileşimli ortamlarda grafik çizimleri üreten açık kaynak kodlu bir Python 2D çizim kütüphanesidir. Matplotlib kütphanesi ile birkaç satır kodla grafikler, histogramlar, güç spektrumları, çubuk grafikler, hata grafikleri, dağılım grafikleri vb. oluşturulabilir(Anonim, 2019). Oluşturulan grafikler dış ortama .jpg ve .png fotmatlarında aktarılabilir. Python kodu içerisnde kullnamak için "import matplotlib.pyplot as plt" ile paket koda eklenir. Scikit-learn, Python programlama dilinde yazılmış ücretsiz bir makine öğrenmesi kütüphanesidir. Karar ağaçları, doğrusal regresyon, rastgele orman, lojistik regresyon, gibi birçok makine öğrenmesi yöntemini içermektedir. Ayrıca çapraz doğrulama yapmak, veri setindeki eksik değerleri doldurmak, öznitelikleri seçmek ve sonuçları değerlendirmek için birçok modülü bünyesinde barındırmaktadır (Anonim 2019). TensorFlow, makine öğrenimi için uçtan uca açık kaynaklı bir platformdur. Araştırmacıların makine öğrenmesinde en son teknolojiyi kullanmasına olanak tanıyan kapsamlı ve esnek bir araç, kitaplık ve topluluk kaynakları ekosistemine sahiptir ve geliştiriciler makine öğrenmesi destekli uygulamaları kolayca oluşturup dağıtırlar (Anonim, 2019). Keras python dilinde yazılmış bir derin öğrenme kütüphanesi-dir. Kolay ve hızlı prototip oluşturmaya olanak tanıyan, CPU ve GPU üzerinde sorunsuz çalışan bir kütüphanedir (Anonim, 2019). Hem evrişimli ağları hem de tekrarlayan ağları ve bu ikisinin kombinasyonlarını destekler.

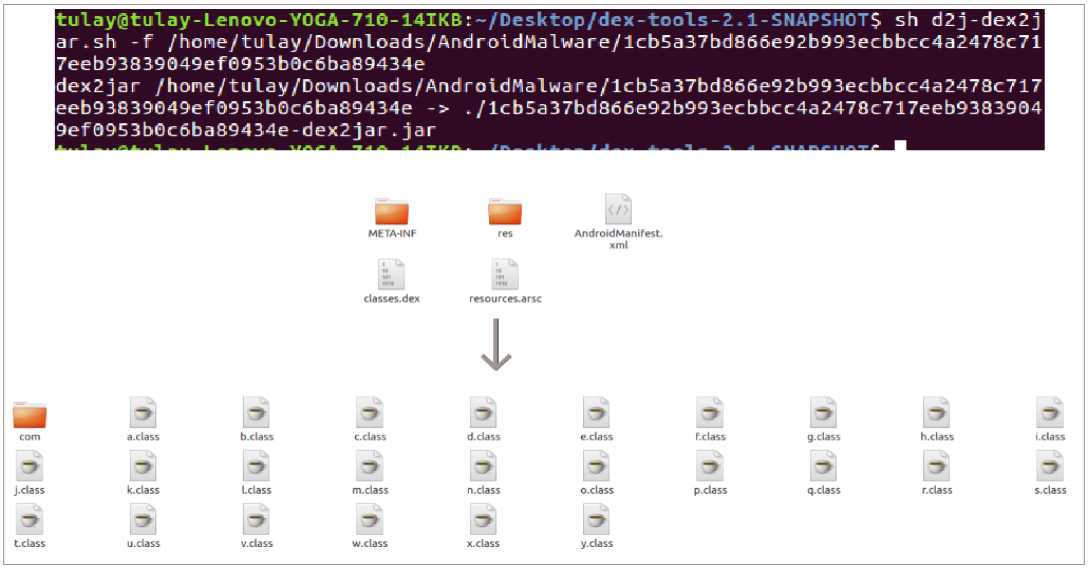
3.2. Yöntem

Bu başlık altında tez çalışmasında kullanılan veri setinin ön işlenmesi, öznitelik seçimi, zararlı yazılım tespit sistemlerinin modellenmesi ve eğitimi açıklanmaktadır.

3.2.1. Veri Önişleme

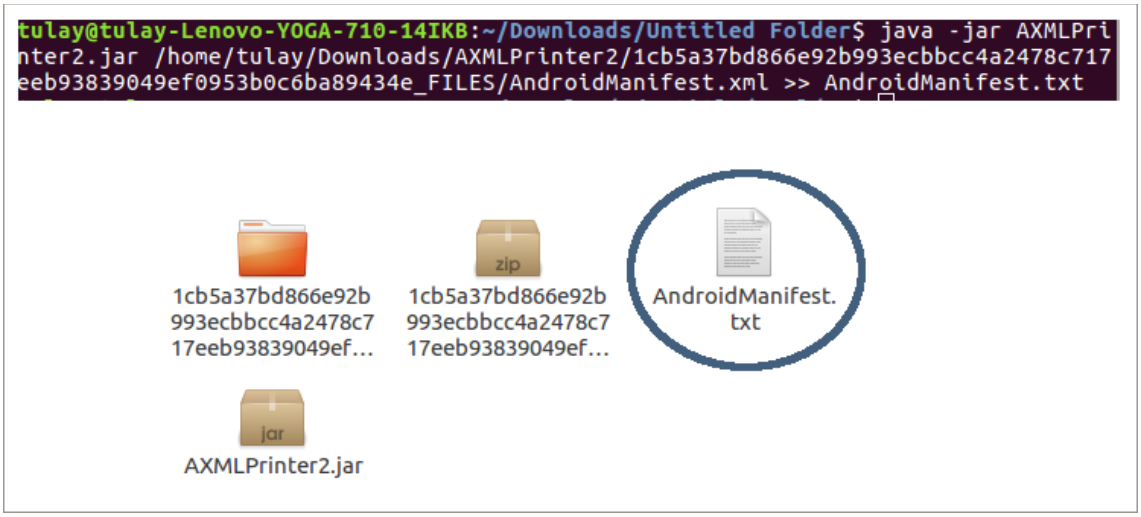
Bu tez çalışmasında kullanılan zaralı yazılım veri seti, Ağustos 2010 - Ekim 2012 döneminde toplanan Drebin veri setinden alınan 5560 zararlı uygulama ile Sanddroid ve Contagio sitelerinden alınan son dört yıla ait 540 zararlı yazılımın statik analiz sonucunda özniteliklerinin ayrıştırılması sonucu oluşmuştur. (Anonim, 2018). Veri setinde kullanılan zararlı olmayan yazılımlar ise Google Play Storedan alınan indirme sayısı yüksek popüler uygulamalardan 6000 tanesi seçilerek oluşturulmuştur. Bir sonraki adım, uygulamaların statik analizleri ile izinler gibi özniteliklerinin ayrıştırılması olacaktır.

Android uygulama paketi (APK), uygulama yazılımını Google'ın Android işletim sistemine dağıtmak ve yüklemek için kullanılan paket dosya biçimidir. Uygulamaları tersine mühendislik işlemleri ile decompile ederek uygulamanın kaynak kodlarına erişmek mümkündür. Bunun için apk decompiler araçlarından olan dex2jar kullanılmıştır. Uygulama analizlerinin için Dex2jar, ApkTool gibi tersine mühendislik araçlarının koşacağı makineye Ubuntu işletim sistemi kurulmuş ve kullanılacak araçlar yüklenerek ortam hazırlanmıştır. 12000 den fazla uygulamanın analizleri bu ortamda yapılarak analiz sonuçları NoSQL bir veritabanında tutulmuştur. Dex2jar aracı dex dosyalarını java beyte kodlarına dönüştürür. Apk dosyasının decopiler edilmiş hali Şekil 3.1 de gösterilmiştir.



Şekil 3.1. Dex2jar Apk Decompiler

AXMLprinter2 (Android apk dosyalarını ayrıştırmak için bir kütüphane) ile apk ayrıştırıldıktan sonra uygulama dosyasından manifest izinleri ve intentleri öznitelik olarak çıkarmamızı sağlar. AXMLprinter2 aracı AndroidManifest.xml dosyasının okunabilir haline çevirimi Şekil 3.2 de gösterilmektedir.



Şekil 3.2. AXMLprinter2 Apk Decompiler

AndroidManifest.xml dosyasının elde edilen okunabilir hali (human readable) Şekil 3.3'de gösterilmektedir.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
        xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.loader"
        platformBuildVersionCode="19"
        platformBuildVersionName="4.4.2-1456859">
        <uses-permission android:name="android.permission.INTERNET"     ></uses-permission>
        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" ></uses-permission>
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
        <uses-permission android:name="android.permission.GET_TASKS"></uses-permission>
        <uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
        <uses-permission android:name="android.permission.WAKE_LOCK"></uses-permission>
        <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"></uses-permission>
        <application
                android:theme="@7F060000"
                android:label="@7F050000"
                android:icon="@7F020000"
                android:name="com.example.loader.MyApplication"
                android:allowBackup="false">
                <activity
                        android:label="@7F050000"
                        android:name="com.example.loader.MainActivity"  >
                        <intent-filter>
                                <action android:name="android.intent.action.MAIN"></action>
                                <category android:name="android.intent.category.LAUNCHER"></category>
                        </intent-filter>
                </activity>
                <activity android:name="com.example.loader.InstallActivity"></activity>
                <activity android:name="com.example.loader.DeviceAdminChecker"></activity>
                <service android:name="com.example.loader.MainService"></service>
                <receiver
                        android:name=".MyDeviceAdminReceiver"
                        android:permission="android.permission.BIND_DEVICE_ADMIN">
                        <intent-filter>
                                <action android:name="android.app.action.DEVICE_ADMIN_ENABLED"></action>
                        </intent-filter>
                        <meta-data
                                android:name="android.app.device_admin"
                                android:resource="@7F030000"></meta-data>
                </receiver>
                <receiver
                        android:name="com.example.loader.ServiceStarter"
                        android:enabled="true"
                        android:exported="true" >
                        <intent-filter>
                                <action android:name="android.intent.action.BOOT_COMPLETED"></action>
                        </intent-filter>
                </receiver>
                <receiver
                        android:name="com.example.loader.SDCardServiceStarter"
                        android:enabled="true"
                        android:exported="true">
                        <intent-filter>
                                <action android:name="android.intent.action.ACTION_EXTERNAL_APPLICATIONS_AVAILABLE"></action>
                        </intent-filter>
                </receiver>
        </application>
</manifest>
```

Şekil 3.3. Örnek Manifest Dosyası

Okunabilir manifest dosyasından python programlama dili kullanılarak <intent-filter> <action> ve <uses-permission> etiketleri kullanılarak izinler ve intentler öznitelik olarak çıkarılmıştır. Buna ek olarak, API çağrılarını ayırıştırılması için Baksmali kullanılmıştır. Baksmali aracılığıyla .dex dosyalarının tersine mühendislik yoluyla API çağrıları çıkarılmıştır. Bunun yanında Sanddroid online otomatik bir uygulama analiz sistemidir, gönderilen apk dosyalarının statik, dinamik analizini gerçekleştirerek kapsamlı bir rapor sunmaktadır. Son dört yıla ait toplanan 540 zaralı yazılımın analizinde Sandroid kullanılmıştır.

Veri setinde Android Manifest uygulama izinleri, API çağrıları ve intentler öznitelik (feature) olarak seçilmiştir.

a) İzinler: İzin modelleri, hassas bilgilere (rehber ve SMS gibi) veya donanıma (kamera ve internet erişimi gibi) erişim kontrolü sağlamak için akıllı telefon işletim sistemlerinin birincil güvenlik mekanizmalarından biri haline gelmiştir. Android'in resmi geliştirici belgelerine (Anonim, 2019) göre, hiçbir Android uygulaması varsayılan olarak kullanıcının bilgilerini, işletim sistemini veya diğer uygulamaları etkileyen işlemleri gerçekleştirme iznine sahip değildir. Her Android uygulaması, işlem sanal alanında farklı sistem kimliğiyle çalışır, bu nedenle her uygulama diğerlerinden ve çalışan sistemden ayrılır. Bu nedenle, bu mimaride, her uygulama, sistem izinlerinde bulunan paylaşmaları veya kullanmaları gereken donanımı, bilgileri veya kaynakları açıkça beyan etmelidir. Android sistemi ayrıca her uygulamanın tam olarak adlandırılmış bir AndroidManifest.xml dosyası içermesini ister. Bu dosyanın amacı, <uses-permission> etiketi altındaki sistem izinleri de dahil olmak üzere, Android sistemini bilgilendirmek için uygulama ile ilgili tüm bilgileri listelemektir. Bu izinler, Manifest dosyasında benzersiz bir etiketle tanımlanır (Anonim, 2019).

b) Intents: Bir işlemin gerçekleşmesi için tanımlanan soyut ifadelerdir. Intent'leri çok çeşitli görevlerde kullanabilmektedir. Bir Aktivite başlatmak için startActivity, bir arka plan Servisi ile iletişim kurmak için Context.startService (Intent) yada Context.bindService (Intent, ServiceConnection, int) kullanılır (Anonim, 2019).

c) Api Çağrıları: Android API, Android SDK'nın tamamını oluşturan çeşitli yazılım modüllerinin koleksiyonunu ifade eder. Android yazılımı çeşitli işlemleri gerçekleştirmek için ilgili yazılımla etkileşime geçmesi gerekir bu durumlarda API çağrıları kullanılır.

Uygulamadan çıkarılan 205 özniteliğin vektörlerinden oluşan veri setinde bulunan özniteliklerden (features) Manifest İzinler Çizelge 3.1. de, Intent'ler Çizelge 3.2. de, API çağrıları ise Çizelge 3.3. de gösterilmektedir.

Çizelge 3.1. Veri Setinde Kullanılan Manifest İzinleri

Manifest İzinleri	**Açıklaması**
ACCESS_CHECKIN_PROPERTIES	Yüklenen değerleri değiştirmek için, check-in veritabanındaki "özellikler" tablosuna okuma / yazma izin verir.
ACCESS_COARSE_LOCATION	Uygulamaya, baz istasyonları ve Kablosuz ağ gibi konum kaynaklarından alınan yaklaşık konuma erişme izni verir.
WRITE_SECURE_SETTINGS	Uygulamaya, güvenli sistem ayarlarını okuma veya yazma izni verir.
SET_ACTIVITY_WATCHER	Uygulamaya izleme ve denetleme izni verir.
SET_WALLPAPER_HINTS	Uygulamalara duvar kağıdını ayarlama izni verir.
ACCESS_FINE_LOCATION	Kullanıcının geçerli konumuna erişime izin veren izinler için kullanılır. Uygulamaya, GPS, baz istasyonları ve Wi-Fi gibi konum kaynaklarından kesin konuma erişme izni verir.
WRITE_EXTERNAL_STORAGE	Uygulamaya harici depolama birimine yazma izni verir.
BIND_VPN_SERVICE	Bir VpnService tarafından istenir. Sistemin servise bağlanabilmesini sağlamak için verilir.
DISABLE_KEYGUARD	Ekran kilidiyle ilgili izin grubudur. Uygulamalara tuş kilidini devre dışı bırakma izni verir.
ACCESS_MOCK_LOCATION	Uygulamaya, test için sahte konum sağlayıcıları oluşturma izni verir.
GET_PACKAGE_SIZE	Uygulamaya, herhangi bir paket tarafından kullanılan alanı bulma izni verir.
MODIFY_PHONE_STATE	Telefon durumunun değiştirilmesine izin verir, güç açık, mmi, vb.
CHANGE_COMPONENT_ENABLED_STATE	Uygulamaya, kendi bileşeninden başka bir uygulama bileşeninin üçüncü taraf uygulamalar tarafından kullanılmayacağını değiştirme izni verir.
CLEAR_APP_CACHE	Uygulamaya, cihazda yüklü tüm uygulamaların önbelleklerini temizleme izni verir.
SET_ORIENTATION	Yön ayarına düşük seviyeli erişim sağlar
READ_CONTACTS	Uygulamaya, kullanıcının kişi verilerini okuma izni verir.
DEVICE_POWER	Güç yönetimine düşük seviyeli erişim sağlar.
HARDWARE_TEST	Donanım birimlerine erişim sağlar. Yalnızca donanım testi için tasarlanmıştır.
ACCESS_WIFI_STATE	Uygulamalara, kablosuz ağlar hakkındaki bilgilere erişme izni verir
WRITE_EXTERNAL_STORAGE	Uygulamaya harici depolama birimine yazma izni verir.
WRITE_CONTACTS	Uygulamaya, kullanıcının kişi verilerini yazma izni verir.
EXPAND_STATUS_BAR	Durum çubuğunu değiştiren izinler için kullanılır
INTERNAL_SYSTEM_WINDOW	Uygulamaya, pencereleri açma izni verir.

Çizelge 3.1. Veri Setinde Kullanılan Manifest İzinleri(devam)

Manifest İzinleri	**Açıklaması**
BROADCAST_SMS	Uygulamaya, SMS bildirimi yayınlama izni verir.
CHANGE_WIFI_STATE	Uygulamalara kablosuz bağlantı durumunu değiştirme izni verir
READ_FRAME_BUFFER	Uygulamaya ekran görüntüsü alma izni verir
ACCESS_SURFACE_FLINGER	Uygulamaya, SurfaceFlinger'in düşük seviye özelliklerini kullanma izni verir.
SYSTEM_ALERT_WINDOW	Uygulamaya, kullanıcıyla sistem düzeyinde etkileşim türünü kullanarak pencere açma izni verir.
MOUNT_FORMAT_FILESYSTEMS	Çıkarılabilir depolama için dosya sistemlerinin biçimlendirilmesine izin verir.
CHANGE_CONFIGURATION	Uygulamaya, yerel ayar gibi geçerli yapılandırmayı değiştirme izni verir.
CLEAR_APP_USER_DATA	Uygulamaya, kullanıcı verilerini temizleme izni verir.
MOUNT_UNMOUNT_FILESYSTEMS	Çıkarılabilir depolama için dosya sistemlerinin takılmasına ve sökülmesine izin verir.
BIND_TEXT_SERVICE	Bir TextServisi tarafından istenir. Sistemin servise bağlanabilmesini sağlamak için verilir.
INSTALL_LOCATION_PROVIDER	Uygulamaya, Konum Yöneticisi'ne bir konum sağlayıcı yükleme izni verir.
SET_PROCESS_LIMIT	Uygulamaya, maksimum sayıda ayarlama izni verir.
BIND_APPWIDGET	Uygulamaya, AppWidget hizmetini söyleme izni verir.
FLASHLIGHT	El fenerine erişim sağlar.
READ_LOGS	Uygulamaya, alt düzey sistem günlük dosyalarını okuma izni verir.
PERSISTENT_ACTIVITY	Bir uygulamanın etkinliklerini kalıcı hale getirmesine izin ver.
CONTROL_LOCATION_UPDATES	Üçüncü taraf uygulamaların kullanmaması için konum güncelleme bildirimlerinin etkinleştirilmesine / devre dışı bırakılmasına izin verir.
BROADCAST_WAP_PUSH	Uygulamaya, WAP PUSH bildirimi yayınlama izni verir.
BIND_ACCESSIBILITY_SERVICE	Sistemin bağlanabilmesini sağlamak için erişilebilirliğin bildirilmesini sağlar.
CALL_PHONE	Uygulamaya, gerçekleştirilmeden bir telefon görüşmesi başlatma izni verir.
SET_TIME_ZONE	Uygulamalara, sistem saat dilimini ayarlama izni verir.
RECEIVE_MMS	Uygulamaya gelen MMS mesajlarını izleme, bu mesajları kaydetme veya üzerinde işlem yapma izni verir.
CHANGE_NETWORK_STATE	Uygulamalara ağ bağlantı durumunu değiştirme izni verir.

Çizelge 3.1. Veri Setinde Kullanılan Manifest İzinleri(devam)

STATUS_BAR	Durum çubuğunu değiştiren izinler için kullanılır Uygulamaya durum çubuğunu genişletme veya daraltma izni verir.
KILL_BACKGROUND_PROCESSES	Uygulamaya çağrı yapma izni verir.
SET_ALARM	Uygulamaya, kullanıcı için alarm ayarlamak üzere bir intent yayınlama izni verir.
ACCOUNT_MANAGER	Uygulamalara AccountAuthenticators'ı arama izni verir.
WRITE_GSERVICES	Uygulamaya, Google hizmet haritasını değiştirme izni verir.
BIND_DEVICE_ADMIN	Cihaz yönetim alıcısı tarafından, yalnızca sistemin onunla etkileşime girebilmesini sağlamak için verilir.
WRITE_SETTINGS	Uygulamaya, sistem ayarlarını okuma veya yazma izni verir.
REBOOT	Cihazı yeniden başlatabilmek için gereklidir.
BLUETOOTH_ADMIN	Uygulamalara, bluetooth cihazlarını bulma ve eşleştirme izni verir
BIND_WALLPAPER	Bir WallpaperService tarafından istenir. Sistemin servise bağlanabilmesini sağlamak için verilir.
RECEIVE_WAP_PUSH	Uygulamaya, gelen WAP push mesajlarını izleme izni verir.
DUMP	Uygulamaya, sistem hizmetlerinden durum dökümü bilgilerini alma izni verir.
BATTERY_STATS	Uygulamaya, pil istatistiklerini toplama izni verir.
SET_TIME	Uygulamalara, sistem saat dilimini ayarlama izni verir.
READ_SOCIAL_STREAM	Uygulamaya, kullanıcının sosyal akış verilerini okuma izni verir.
READ_USER_DICTIONARY	Uygulamaya, kullanıcının kullanıcı sözlüğünde depolamış olabileceği tüm kelimeleri, adları ve kelime öbeklerini okuma izni verir.
PROCESS_OUTGOING_CALLS	Telefonun durumuna erişim ve değişiklik yapma ve telefonun durumunu değiştirme ile ilişkili izinler için kullanılır.
CALL_PRIVILEGED	Uygulamaya, acil durum da dahil olmak üzere herhangi bir telefon numarasını arama izni verir.
REORDER_TASKS	Uygulamaya, görevlerin sırasını değiştirme izni verir.
SET_WALLPAPER	Uygulamalara duvar kağıdını ayarlama izni verir.
BIND_INPUT_METHOD	Bir InputMethodService tarafından istenir. Sistemin servise bağlanabilmesini sağlamak için verilir.
UPDATE_DEVICE_STATS	Uygulamaya, cihaz istatistiklerini güncelleme izni verir.
WRITE_PROFILE	Durum çubuğunu değiştiren izinler için kullanılır.

Çizelge 3.1. Veri Setinde Kullanılan Manifest İzinleri(devam)

WRITE_CALL_LOG	Uygulamaya, kullanıcının çağrı günlüğü verilerini yazma izni verir.
DELETE_PACKAGES	Uygulamaya paketleri silme izni verir.
GET_TASKS	Uygulamaları çalıştıran veya arka plan işlemlerini öldüren diğer uygulamalarla ilgili izin grubu.
GLOBAL_SEARCH	Bu izin, içerik sağlayıcılarda küresel aramanın yanı sıra genel kullanıma izin vermek için kullanılabilir.
DELETE_CACHE_FILES	Uygulamaya, önbellek dosyalarını silme izni verir.
SEND_SMS	Bir uygulamanın MMS alan veya okuyan mesajlar göndermesine izin veren izinler için kullanılır. Uygulamaya SMS mesajları gönderme izni verir.
WRITE_USER_DICTIONARY	Uygulamaya, kullanıcının kullanıcı sözlüğünde depolamış olabileceği tüm kelimeleri, adları ve kelime öbeklerini yazma izni verir.
MASTER_CLEAR	Üçüncü taraf uygulamalar tarafından kullanılmaz.
WRITE_CALENDAR	Uygulamaya, kullanıcının takvim verilerini yazma izni verir.
GET_ACCOUNTS	Hesap Yöneticisi tarafından yönetilen hesaplara doğrudan erişim izinleri. Hesaplar Hizmeti'ndeki hesap listesine erişime izin verir
CHANGE_WIFI_MULTICAST_STATE	Uygulamaların Wi-Fi çok noktaya yayın moduna girmesine izin verir
SUBSCRIBED_FEEDS_READ	Uygulamaya, abone olunan özet akışlarına ContentProvider erişimine izin verme izni verir.
ACCESS_NETWORK_STATE	Uygulamaların ağlar hakkındaki bilgilere erişmesine izin verir
VIBRATE	Vibratöre erişim sağlar.
BLUETOOTH	Uygulamalara eşleştirilmiş bluetooth cihazlarına bağlanma izni verir.
READ_CALENDAR	Uygulamaya, kullanıcının takvim verilerini okuma izni verir.
READ_CALL_LOG	Uygulamaya, kullanıcının çağrı kaydını okuma izni verir.
SUBSCRIBED_FEEDS_WRITE	Uygulamaya, o anda senkronize edilmiş özet akışları değiştirme izni verir.
READ_EXTERNAL_STORAGE	Uygulamaya, harici depolama biriminden okuma izni verir.
RESTART_PACKAGES	Uygulamaya, diğer uygulamaların arka plan işlemlerini sonlandırma izni verir.
RECEIVE_BOOT_COMPLETED	Uygulamaya, sistem yeniden başlatmayı tamamladıktan sonra yayınlanan Intent.ACTION_BOOT_COMPLETED etkinliğini alma izni verir. Bu izni istemezseniz, yayını o anda almazsınız.
WAKE_LOCK	İşlemcinin uyku veya ekranın kararmasını önlemek için PowerManager WakeLocks kullanılmasına izin verir.
READ_SYNC_STATS	Uygulamalara senkronizasyon istatistiklerini okuma izni verir.

Çizelge 3.1. Veri Setinde Kullanılan Manifest İzinleri(devam)

BROADCAST_STICKY	Uygulamaya, hedefler yayınlama izni verir. Bunlar bir sonraki yayını beklemek içindir.
MODIFY_AUDIO_SETTINGS	Uygulamaya, genel ses ayarlarını değiştirme izni verir.
READ_PROFILE	Uygulamaya adınız ve iletişim bilgileriniz gibi cihazınızda saklanan kişisel profil bilgilerini okuma izni verir.
BIND_REMOTEVIEWS	Bir RemoteViewsService tarafından istenir. Sistemin servise bağlanabilmesini sağlamak için verilir.
WRITE_APN_SETTINGS	Uygulamalara apn ayarlarını yazma izni verir.
READ_PHONE_STATE	Telefon durumuna salt okunur erişim sağlar.
ACCESS_LOCATION_EXTRA_COMMANDS	Uygulamaya, ekstra konum sağlayıcı komutlarına erişme izni verir
NFC	Uygulamalara, NFC üzerinden Girdi / Çıktı işlemleri yapma izni verir.
RECORD_AUDIO	Uygulamaya ses kaydetme izni verir.
INTERNET	Uygulamalara ağ soketleri açma izni verir.
READ_HISTORY_BOOKMARKS	Uygulamaya, kullanıcının göz atma geçmişini ve yer işaretlerini okuma izni verir.
WRITE_SYNC_SETTINGS	Uygulamalara senkronizasyon ayarlarını yazma izni verir.
CAMERA	Kamera cihazına erişebilmek için gereklidir.
INSTALL_PACKAGES	Uygulamaya, paketleri yükleme izni verir.
WRITE_HISTORY_BOOKMARKS	Uygulamaya, kullanıcının göz atma geçmişini ve yer işaretlerini yazma izni verir.
WRITE_SMS	Uygulamaya, SMS mesajları yazma izni verir.
READ_SYNC_SETTINGS	Senkronizasyon ayarlarına erişen veya ilgili bilgileri senkronize eden izinler için kullanılır.
RECEIVE_SMS	Uygulamaya gelen SMS mesajlarını izleme, bu mesajları kaydetme veya işleme koyma izni verir.
AUTHENTICATE_ACCOUNTS	Uygulamaya, AccountManager için AccountAuthenticator görevi yapma izni verir
USE_CREDENTIALS	Uygulamaya, AccountManager'dan kimlik doğrulaması isteme izin verir
MANAGE_ACCOUNTS	Uygulamaya, AccountManager'daki hesap listesini yönetme izni verir
RECEIVE_BOOT_COMPLETED	Uygulamaya, kullanıcıya alma izni verir.
READ_SMS	Uygulamaya, SMS mesajlarını okuma izni verir.

Çizelge 3.2. Veri Setinde Kullanılan Intentler

Intent Adı	**Açıklaması**
Android.intent.action.BOOT_COMPLETED	Cihaz yeniden başlatıldıktan sonra çalışır.
Android.intent.action.SEND_MULTIPLE	Bir başkasına birden çok veri iletmek için kullanılır.
Android.intent.action.TIME_SET	Sistemin zamanının değişmesine cevap verir.
Android.intent.action.PACKAGE_REMOVED	Mevcut bir uygulama paketini cihazdan kaldırır.
Android.intent.action.TIMEZONE_CHANGED	Saat dilimi değiştirir.
Android.intent.action.ACTION_POWER_DISCONNECTED	Bu, “Harici güç aygıttan çıkarıldı.” bildirimine kaydolmak için kullanılır.
Android.intent.action.PACKAGE_ADDED	Mevcut bir uygulama paketini cihaza ekler.
Android.intent.action.ACTION_SHUTDOWN	Cihaz kapatılırken (tamamen kapalı, uykuda değil) yayınlanır.
Android.intent.action.PACKAGE_DATA_CLEARED	Bir paketin verilerini temizler.
Android.intent.action.PACKAGE_CHANGED	Mevcut bir uygulama paketini değiştirir.
Android.intent.action.NEW_OUTGOING_CALL	Giden çağrıları yönetmek için kullanılır.
Android.intent.action.SENDTO	Veriler tarafından belirtilen birine mesaj gönderir.
Android.intent.action.CALL	Verdiğimiz telefon numarası verisiyle arama yapmamızı sağlar.
Android.intent.action.SCREEN_ON	Cihaz uyandığında ve etkileşimli hale geldiğinde gönderilir.
Android.intent.action.BATTERY_OKAY	Cihazın düşük pil durumundan çıktığını haber verir.
Android.intent.action.PACKAGE_RESTARTED	Kullanıcı bir paketi yeniden başlattı ve tüm süreçler öldürüldü.
Android.intent.action.CALL_BUTTON	Kullanıcının “Ara” butonuna bastığını haber verir.
Android.intent.action.SCREEN_OFF	Cihazın ekranının kapatıldığını haber verir.
intent.action.RUN	Verileri çalıştırır
Android.intent.action.SET_WALLPAPER	Duvar kağıdı seçme ayarlarını gösterir.
Android.intent.action.BATTERY_LOW	Cihazdaki düşük pil durumunu gösterir.
Android.intent.action.ACTION_POWER_CONNECTED	Bu, özellikle “Harici güç aygıta bağlandı.” bildirime kaydolmak isteyen uygulamalar için tasarlanmıştır.

Çizelge 3.3. Veri Setinde Kullanılan API Çağrıları

API Çağrıları	**Açıklaması**
onServiceConnected	Android sistem istemci ve servis arasında bağlantı oluşturduğunda çağırılır.
bindService	İstemci bindServise ile bir hizmete bağlanır.
attachInterface	Belirli bir arabirimi Binder ile ilişkilendirmek için kullanılır.
ServiceConnection	Bir uygulama hizmetinin durumunu izlemek için arabirimdir.
android.os.Binder	`IBinder` için temel bir sınıftır.
Ljava.lang.Class.getCanonicalName	Java Dili tarafından tanımlanan temel sınıfın Canonical adını döndürür.
Ljava.lang.Class.getMethods	Bu yöntem class veya interface in üst sınıfını yada miras alan sınıfları yansıtmaktadır.
Ljava.lang.Class.cast	Sınıflar arasında miras ilişkisi bulunması halinde (inheritance), bu sınıflardan türetilen nesnelerin birbirine akıtılmasını sağlar.
Ljava.net.URLDecoder	URL yapısını decode ederek bilinen URL formatına getirir.
android.content.pm.Signature	Bir uygulama paketiyle ilişkili imza sertifikası için kullanılır.
android.telephony.SmsManager	Veri, metin ve SMS mesajları gönderme gibi SMS işlemlerini yönetir.
getBinder	Bu Messenger'ın ilişkili Handler ile iletişim kurmak için kullandığı IBinder'ı almak için kullanılır.
ClassLoader	ClassLoader sınıfların yüklenmesinden sorumlu bir nesnedir. Soyut bir sınıftır.
Landroid.content.Context.registerReceiver	Varsayılan olarak uyguulama diğer uygulamalardaki alıcılarla etkileşime giremez; bu yöntem, uygulamanın'ın etkileşime girebileceği bir alıcıya kaydolmamıza olanak tanır.
Ljava.lang.Class.getField	Class nesnesi tarafından temsil edilen sınıfın veya arabirimin belirtilen genel üye alanını yansıtan bir Field nesnesi döndürür.
Landroid.content.Context.unregisterReceiver	Önceden kaydedilmiş bir BroadcastReceiver kaydının kaldırmasında kullanılır.
getCallingUid	Geçerli işlemi gönderen, işleme atanan Linux kullanıcı kimliğini döndürün.
Ljavax.crypto.spec.SecretKeySpec	Bu sınıf, sağlayıcıdan bağımsız bir şekilde gizli bir anahtar belirtir.
android.content.pm.PackageInfo	Bir paketin içeriği hakkında genel bilgileri elde etmek için kullanılır. Bu, AndroidManifest.xml dosyasından toplanan tüm bilgilere karşılık gelir.
DexClassLoader	Bu, uygulamanın bir parçası olarak yüklenmemiş kodu yürütmek için kullanılabilir.

Çizelge 3.3. Veri Setinde Kullanılan API Çağrıları (devam)

API Çağrıları	**Açıklaması**
HttpGet.init	Get işlemleri için kullanılır.
SecretKey	Bu arayüzün amacı tüm gizli anahtar arayüzlerini gruplandırmaktır. (ve tip güvenliği sağlamak).
Ljava.lang.Class.getMethod	Bu Class nesnesi tarafından temsil edilen sınıfın veya arayüzün belirtilen genel üye yöntemini yansıtan bir metot nesnesi döndürür.
System.loadLibrary	Libname argümanı tarafından belirtilen yerel kütüphaneyi yükler.
android.intent.action.SEND	Bazı verileri başka birine iletir.
Ljavax.crypto.Cipher	Bu sınıf, şifreleme ve şifre çözme için bir şifreleme şifresinin işlevselliğini sağlar.
android.telephony.gsm.SmsManager	Veri, metin ve SMS mesajları gönderme gibi SMS işlemlerini yönetir.
TelephonyManager.getSubscriberId	Benzersiz bir abone kimliği, örneğin bir GSM telefonu için IMSI değerini döndürür.
Runtime.getRuntime	Geçerli Java uygulamasıyla ilişkili çalışma zamanı nesnesini döndürür.
Ljava.lang.Object.getClass	Nesnenin çalışma zamanı sınıfını döndürür.
Ljava.lang.Class.forName	Bir sınıf veya arayüzün tam adı verildiğinde bu yöntem sınıfı veya arayüzü bulmaya, yüklemeye ve bağlamaya çalışır.
Binder	IBinder tarafından tanımlanan bir uzak yordam çağrı mekanizmasının temel sınıfıdır.
IBinder	Sistem `Ibinder ile` servis iletişim kanalını önbelleğe alır
android.os.IBinder	İşlem içi çağrıları yaparken yüksek performans için tasarlanmış bir uzak çağrı mekanizmasının temel arayüzüdür.
abortBroadcast	Alıcının geçerli yayını iptal etmesi gerektiğini belirten bayrağı ayarlamak için kullanılır.
TelephonyManager.getDeviceId	Benzersiz cihaz kimliğini döndürür; örneğin, GSM için IMEI ve CDMA telefonlar için MEID veya ESN.
getCallingPid	İşlenmekte olan geçerli işlemin kimliğini döndürün.
Ljava.lang.Class.getDeclaredClasses	Java.lang.Class sınıfının getDeclaredClasses () yöntemi, bu sınıfın özel, genel, korumalı veya varsayılan sınıflar olan ve üyeleri olan, ancak devralınan sınıfları almak için kullanılır.
TelephonyManager.getSimSerialNumber	SIM'in seri numarasını verir.

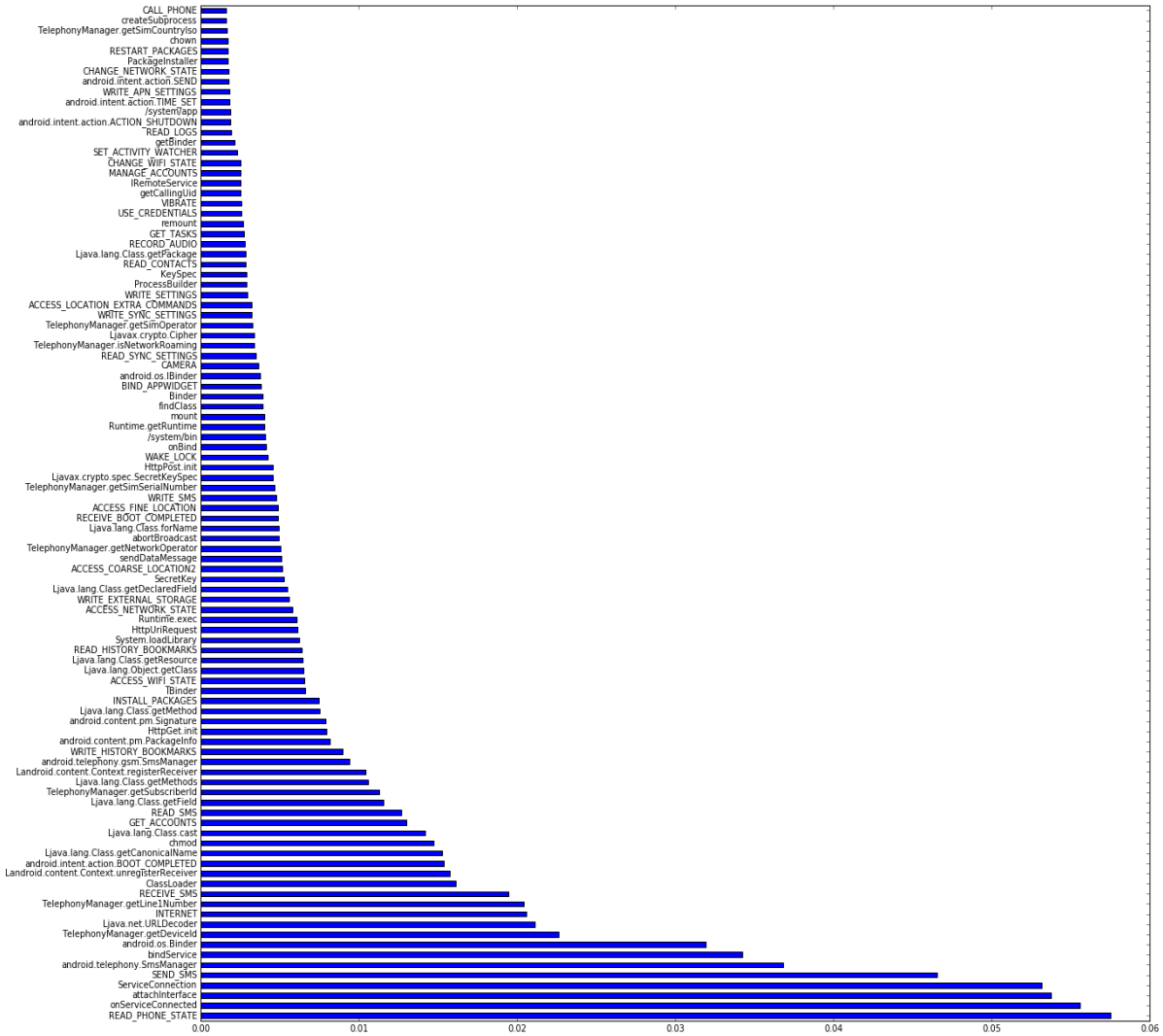
Çizelge 3.3. Veri Setinde Kullanılan API Çağrıları (devam)

API Çağrıları	**Açıklaması**
TelephonyManager.getCallState	Cihazdaki tüm aramaların durumunu döndürür.
TelephonyManager.getSimCountryIso	SIM sağlayıcısının ülke kodu için ISO-3166 ülke koduna eşdeğer bir değer döndürür.
sendMultipartTextMessage	Multi-part metin tabanlı bir SMS'i göndermek için kullanılır.
PackageInstaller	Cihazdaki uygulamaları yükleme, yükseltme ve kaldırma olanağı sunar.
sendDataMessage	Belirli bir uygulama bağlantı noktasına veri tabanlı bir SMS göndermek için kullanılır.
HttpPost.init	Post işlemleri için kullanılır.
Ljava.lang.Class.getClasses	Java.lang.Class.getClasses () yöntemi, bu Class nesnesi tarafından temsil edilen sınıfın üyeleri olan tüm ortak sınıfları ve arabirimleri temsil eden Class nesnelerini içeren bir dizi döndürür
TelephonyManager.isNetworkRoaming	Cihaz, GSM şebekesi için mevcut şebekede dolaşım olarak değerlendirilirse true değerini döndürür.
HttpUriRequest	HttpClient veri göndermek ve almak için bir HttpUriRequest kullanır. HttpUriRequest'in önemli alt sınıfı HttpGet ve HttpPost'tur.
divideMessage	Bir mesaj metnini, maksimum SMS mesajı boyutundan daha büyük olmayan birkaç parçaya böler.
Runtime.exec	Belirtilen komut ve bağımsız değişkenleri, belirtilen ortam ve çalışma diziniyle ayrı bir işlemde yürütür.
TelephonyManager.getNetworkOperator	Mevcut kayıtlı operatörün sayısal adını (MCC + MNC) döndürür.
IRemoteService	Uzak bir hizmete çalışma zamanı erişimi sağlayan arabirim.
TelephonyManager.getSimOperator	SIM sağlayıcısının MCC + MNC'sini (mobil ülke kodu + mobil ağ kodu) döndürür.
onBind	Sistem `onBind() ile` istemci ilk bağlandığında hizmetin yöntemini çağırır.
Context.bindService	Uygulamanızla servis arasında bir bağımlılık tanımlar.
ProcessBuilder	Bu sınıf, işletim sistemi süreçleri oluşturmak için kullanılır.
Ljava.lang.Class.getResource	Java Class sınıfının getResource () yöntemi, bu sınıfın bulunduğu modülün kaynaklarını döndürmek için kullanılır.
defineClass	Sınıf dosyası biçiminde bir sınıf tanımı içeren bir bayt dizisinden yeni bir sınıf oluşturur.

Çizelge 3.3. Veri Setinde Kullanılan API Çağrıları (devam)

API Çağrıları	**Açıklaması**
findClass	Belirtilen binary isme sahip sınıfı bulur.
TelephonyManager.getLine1Number	Hat 1 için telefon numarası dizesini, örneğin bir GSM telefonu için MSISDN'yi döndürür.
KeySpec	Bir kriptografik anahtar oluşturan anahtarın bir belirtimi. Anahtar bir donanım aygıtında saklanıyorsa, belirtimi aygıttaki anahtarın tanımlanmasına yardımcı olacak bilgiler içerebilir.
PathClassLoader	Yerel dosya sistemindeki dosyalar ve dizinler listesinde çalışan, ancak ağdan sınıf yüklemeye çalışmayan basit bir ClassLoader uygulaması sağlar.
Runtime.load	Dosya adı değişkeni tarafından belirtilen yerel kitaplığı yükler.

Yukarıda Çizelge 3.1. , Çizelge 3.2. ve Çizelge 3.3. de gösterilmekte olan veri setindeki özniteliklerin önem sırasına göre ilk 100 öznitelik Şekil 3.4 ‘de gösterilmektedir.



Şekil 3.4. Öznitelik Önem Sırası

Özniterliklerin önem sırasına göre READ_PHONE_STATE, ACCESS_WIFI_STATE SEND_SMS, RECIEVE_SMS, READ_SMS, GET_ACCOUNT, CALL_PHONE vb. izinlerinin zaralı yazılım tespitinde SMS işlemleri , konum bilgileri, telofon durumu hakkında ve internet bağlantısı ile ilgili izinlerin ayrıştırıcı izinler olduğu görülmektedir. Ayrıca bu izinler Android tehlikeli izinler listesinde bulunmaktadır. OnServiceConnected, attachInterface, ServisConnection API çağrıları ayrıştırıcı özelliktedir. İstemci ve servis arasında bağlantı oluşturduğunda onServiceConnected çağırılır. Belirli bir arabirimi Binder ile ilişkilendirmek için attachInterface kullanılır. Servis bağlantıları için ServisConnection kullanılmaktadır. Servis bağlantıları ile ilgili API çağrıları ayrıştırıcı özniteliklerdir.

3.2.2. Makine Öğrenmesi Yöntemleri

Veri setleri genellikle .csv formatında bulunur. CSV uzantılı dosyalar, veritabanı kullanıcıları için verileri virgüller ile ayırarak belli bir düzende yazıp kaydedilmiş olan dosyalardır. Ön işlemesi yapılan .csv formatındaki veri setimizi dataframe olarak okumak için pandas kütüphanesinin read_csv metodu kullanılmıştır.

Bu çalışmada, öznitelikleri çıkarılan veri seti üzerinde makine öğrenme sınıflandırıcılarının performansını değerlendirmek için 10 kat çapraz doğrulama (10-fold cross validation) yapmıştır. 10 katlı çapraz değerleme ilk önce veri setini 10 katmana böler ve 1. katı test için kullanırken kalan katları eğitim seti olarak kullanır. 2. adımda, 2. kat test olarak kullanılırken, geriye kalan katlar eğitim seti olarak kullanılır. 3. , 4. , 5. ve diğer katlarda aynı işlemler yapılarak veri setindeki her örneğin hem eğitim hem test için kullanılması sağlanmıştır. Hesaplama açısından pahalı olsa da, öğrenme modelinin performansını değerlendirmek ve overfittig engelenmek için çapraz doğrulama önemlidir.

3.2.2.1. <u>Makine Öğrenmesi Algoritmaların Modellenmesi</u>

a) Lojistik Regresyon (LR- Logistic Regression) Bu modelde, tek bir çalışmanın olası sonuçlarını tanımlayan olasılıklar bir lojistik fonksiyon kullanılarak modellenmiştir . Lojistik regresyon, y değişkenin (zararlı (malware) yada iyi (benign)) olma olasılığı ile veri setindeki x öznitelikleri (izin, API çağrıları, Intens öznitelikleri) arasındaki ilişkiyi anlamamıza yardımcı olur. Lojistik fonksiyon (f(x)), -∞/+∞ aralığındaki tüm değerleri girdi olarak kabul edebilir. Çıktı olarak ise 0 ile 1 arasında değerler almaktadır. Hedef değişkenin doğası gereği kategorik olduğu özel bir Doğrusal Regresyon örneğidir. (Anonim, 2019)

Doğrusal Regresyon Denklemi:

$$y = \beta\,0 + \beta\,1\;X\;1 + \beta\,2\;X\;2 + \ldots. + \beta nXn \qquad (3.1)$$

Doğrusal Regresyon Denklemi 3.1’de belirtildiği gibi , y bağımlı değişkendir ve x1, x2 ... ve Xn açıklayıcı değişkenlerdir.

Sigmoid fonksiyonu:

$$p = 1/1 + e^{-y} \tag{3.2}$$

Lojistik fonksiyon olarak da adlandırılan Deklem 3.2’de görterilmektekte olan sigmoid fonksiyonu, herhangi bir gerçek değerli sayıyı alıp 0 ile 1 arasında bir değerle eşleştirebilen 'S' şeklinde bir eğri verir. Eğri pozitif sonsuza giderse, y'nin 1 olacağı tahmin edilir ve eğer eğri negatif sonsuza gider, y'nin 0 olacağı tahmin edilir. Sigmoid fonksiyonunun çıktısı 0,5'ten fazlaysa, sonucu 1 olarak sınıflandırabiliriz ve 0,5'ten azsa, 0 dır . (Anonim, 2019)

Lojistik Regresyon modeli geliştirirken, ilk olarak scikit learn Lojistik Regresyon modülünü import edilmekte ve LogisticRegression() fonksiyonunu kullanarak bir Lojistik Regresyon sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan Logistik Regresyon nesnesinin parametreleri verilerek **LogisticRegression(penalty='l2', dual=False, tol=1e-4, C=1.0, fit_intercept=True, verbose=0, intercept_scaling=1, class_weight=None, random_state=50, solver='sag', max_iter=10, multi_class='ovr', warm_start=False, n_jobs=None, l1_ratio=None)** modeli oluşturulmaktadır. Oluşturulan model fit() metodu kullanarak eğitilmektedir.

b) Destek Vektör Makinesi (SVM- Support Vector Machine) bir sınıflandırma yaklaşımı olarak kabul edilir, ancak hem sınıflandırma hem de regresyon problemlerinde kullanılabilir. Birden fazla sürekli ve kategorik değişkeni kolayca işleyebilir. SVM, farklı sınıfları ayırmak için çok boyutlu alanda bir hiper düzlem oluşturur. SVM Hiperparametreleri kernel, regularization, gamma dır. Destek Vektör Makinesi modeli geliştirirken, ilk olarak scikit learn destek vektör makinesi modülünü import edilmekte ve **SVC(C=100, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=100, class_weight=None, verbose=False, max_iter=300,decision_function_shape='ovr', break_ties=True, random_state=None)** fonksiyonunu kullanarak bir Destek Vektör Makinesi nesnesi oluşturulmaktadır. Scikit-

learn SVC methodunun gerçeklenmesi libsvm dayanmaktadır. Oluşturulan model fit() metodu ile eğitilmektedir. SVM Sınıflandırıcılar, Naïve Bayes algoritmasına kıyasla iyi bir doğruluk sunar ve daha hızlı tahmin gerçekleştirir.

c) Naif Bayes (NB- Naïve Bayes), Bayes Teoremine dayanan istatistiksel bir sınıflandırma tekniğidir. En basit denetimli öğrenme algoritmalarından biridir. Naif Bayes sınıflandırıcı hızlı, doğru ve güvenilir bir algoritmadır. Naif Bayes sınıflandırıcıları büyük veri kümelerinde yüksek doğruluk ve hıza sahiptir. Naive Bayes'in hesaplama maliyeti çok düşüktür. Naif Bayes modeli için scikit-learn kütüphanesi kullanılmıştır. Scikit-learn `GaussianNB` sınıflandırma için Gaussian Naive Bayes algoritmasını uygular. Deklem 3.3' de verilen Gaussian Naive Bayes algoritmasında μy ve σ_y parametreleri maksimum olasılık kullanılarak tahmin edilmektedir (Anonim, 2019).

$$p(xi \vee y) = \frac{1}{\sqrt{2\,\Pi\,\sigma_y^2}} \exp\left(\frac{-(xi-\mu\,y)^2}{2\sigma_y^2}\right) \tag{3.3}$$

Naif Bayes modeli geliştirirken, ilk olarak scikit learn Naïve Bayes modülünü import edilmekte ve **GaussianNB(priors=None, var_smoothing=1e-09)** fonksiyonunu kullanarak bir Naif Bayes sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan model fit() metodu ile eğitilmektedir.

d) Doğrusal Ayrımcılık Analizi (LDA- Linear Discriminant Analysis) belirli bir veri setinde sınıflar arası varyansın sınıf içi varyansa oranını en üst düzeye çıkarır, böylece maksimum ayrılabilirliği garanti eder.(Balakrishnama,1998) Modeli geliştirirken, ilk olarak scikit learn Doğrusal Ayrımcılık Analizi modülünü import edilmekte ve **LinearDiscriminantAnalysis(solver='svd', tol=1.0e-4, shrinkage=None, priors=None, n_components=None, store_covariance=True)** fonksiyonunu kullanarak bir Doğrusal Ayrımcılık Analizi sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan model fit() metodu ile eğitilmektedir.

e) Rastgele Orman (RF- Random Forest) Rasgele Orman Algoritması denetimli bir öğrenme algoritmasıdır. Hem sınıflandırma hem de regresyon için kullanılabilir. Rastgele

Orman Algoritması, sürece katılan karar ağaçlarının sayısı nedeniyle oldukça doğru ve sağlam bir yöntem olarak kabul edilmektedir. Algoritma hem sınıflandırma hem de regresyon problemlerinde kullanılabilir. Rasgele Orman Algoritması, birden fazla karar ağacı içerdiğinden tahminler üretmede yavaştır. Ne zaman bir tahmin yaparsa, ormandaki tüm ağaçlar aynı girdi için bir tahmin yapmak ve daha sonra oy vermek zorundadır. Tüm bu süreç zaman alıcıdır. Rasgele Orman modeli geliştirirken, ilk olarak scikit learn Random Forest modülünü import edilmekte ve **RandomForestClassifier (n_estimators=120, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',verbose=0, max_leaf_nodes=None, min_impurity_decrease=0.0, ccp_alpha=0.0, n_jobs=None, min_impurity_split=None, bootstrap=True, oob_score=False, random_state=None, warm_start=False, class_weight=None, max_samples=None)** fonksiyonunu kullanarak bir Rasgele Orman sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan model fit() metodu kullanarak eğitilmektedir.

f) K en yakın komşu (K-NN- K Nearest Neighbors) KNN algoritması hem sınıflandırma hem de regresyon problemleri için kullanılır. KNN parametrik olmayan ve tembel bir öğrenme algoritmasıdır. Parametrik olmayan, altta yatan veri dağıtımı için herhangi bir varsayım olmadığı anlamına gelir. Başka bir deyişle, model yapısı veri kümesinden belirlenmiştir. KNN'de K, en yakın komşu sayısıdır. Komşu sayısı temel karar faktörüdür. Sınıf sayısı 2 ise K genellikle tek bir sayıdır. K = 1 olduğunda, algoritma en yakın komşu algoritması olarak bilinir. Algoritmanın temelde izlediği yol mesafeyi hesaplama, en yakın komşuları bulup ve etiketlere oy verme şeklindedir (Keller,1985). K En Yakın Komşu modeli geliştirirken, ilk olarak scikit learn K En Yakın Komşu modülünü import edilmekte ve **KNeighborsClassifier(n_neighbors=50, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)** fonksiyonunu kullanarak bir K En Yakın Komşu sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan model fit() metodu kullanarak eğitilmektedir.

g) Adaptif Yükseltme (AB- Adaptive Boosting) Ada-boost veya Adaptive Boosting, 1996 yılında Yoav Freund ve Robert Schapire tarafından önerilen topluluk artırıcı sınıflandırıcılardan biridir. Sınıflandırıcıların doğruluğunu artırmak için birden fazla sınıflandırıcıyı birleştirir. AdaBoost yinelemeli bir topluluk yöntemidir. AdaBoost sınıflandırıcı, düşük performanslı birden fazla sınıflandırıcıyı birleştirerek güçlü bir

sınıflandırıcı oluşturur, böylece yüksek doğrulukta güçlü sınıflandırıcı elde edersiniz (Freund, 1996). Adaptif Yükseltme modeli geliştirirken, ilk olarak scikit learn Adaptif Yükseltme modülünü import edilmekte ve **AdaBoostClassifier(base_estimator=None, n_estimators=50, learning_rate=1, algorithm='SAMME.R', random_state=100)** fonksiyonunu kullanarak bir Adaptif Yükseltme sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan model fit() metodu kullanarak eğitilmektedir.

h) Torbalama (BA- Bootstrap aggregating (Bagging)) Bir Bagging sınıflandırıcısı, temel sınıflandırıcıları her biri orijinal veri kümesinin rastgele alt kümelerine sığdıran ve daha sonra nihai tahmin oluşturmak için bireysel tahminlerini (oylayarak veya ortalayarak) toplayan bir meta tahmincisidir.(Anonim, 2019) Var olan bir eğitim setinin birden çok versiyonunu oluşturur ve sonucu tahmin ederken versiyonların ortalamasını alır ve bir sınıfı tahmin ederken çok sayıda oy kullanır (Breiman,1996). Modeli geliştirirken, ilk olarak scikit learn Bagging modülünü import edilmekte ve **BaggingClassifier(base_estimator=None, n_estimators=18, max_samples=1.0, max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None, random_state=None, verbose=0)** fonksiyonunu kullanarak bir Bagging sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan model fit() metodu kullanarak eğitilmektedir.

ı) Karar Ağaçları CART (DT- Desicion Trees (CART))sınıflandırma ve regresyon için kullanılan parametrik olmayan denetimli bir öğrenme yöntemidir. Amaç, veri özelliklerinden çıkarılan basit karar kurallarını öğrenerek bir hedef değişkenin değerini tahmin eden bir model oluşturmaktır (Scikit-learn,2019). Modeli geliştirirken, ilk olarak scikit learn karar ağaçları modülünü import edilmekte ve **DecisionTreeClassifier (criterion='gini',ccp_alpha=0.0,max_depth=None,min_samples_split=4,splitter='best' ,min_impurity_decrease=0.0, min_samples_leaf=1, min_weight_fraction_leaf=0.0, class_weight=None,max_features=None, random_state=None, max_leaf_nodes=None, presort='deprecated',min_impurity_split=None)** fonksiyonunu kullanarak bir karar ağaçları sınıflandırıcı nesnesi oluşturulmaktadır. Oluşturulan model fit() metodu kullanarak eğitilmektedir.

i) Derin İnanç Ağları (DBN- Deep Belief Networks), denetimli öğrenme için bir Derin inanç ağı oluşturulurken Kısıtlı Boltzmann Makineleri yığını kullanır. RBM iki katmandan oluşur ilk katmanı görünür katman ikincisi ise gizli katmandır. RBM de düğümler birbirine katmanlar halinde bağlanırken aynı katmandaki iki düğüm birbirine bağlanmazlar. Yani katmanlar arası iletişim yoktur. DBN modelini oluşturuken Tensorflow kütüphanesi kullanılmıştır. Ön hazırlık aşamasında iki RBM kullanılır, birincisi 784-512 ve ikincisi 512-256'dır. RBM'lerin eğitim parametreleri katman bazında belirtilebilir. Örneğin her katman için öğrenme oranını rbm_learning_rate 0.005, 0.1 şeklinde belirtilir. Düğüm çıktısı üretmek için ReLU aktivasyon fonksiyonunu kullanır. n_layers ise RBM modelimizin derinliğidir. RBM modeli oluşturulurken optimizasyon algoritması olarak sgd kullanılmıştır. Gizli Katman (hidden layer) , öğrenme oranı (learning rate), rbm öğrenme oranı (rbm learning rate), rbm eğitim tur (epoch), seyreltme (dropout), parti boyutu (batch size), aktivasyon fonksiyonu modelimizin hiper parametrelerini oluşturmaktadır. DBN modeli oluşturulurken yukarıda verilen hiperparametrelerin değerlerini atayarak en iyi doğruluk sonucunu veren model geliştirilmiştir.

j) Çok katmanlı algılayıcı (MLP- Multi Layer Perceptron), ilk adım, MLPClassifier sınıfı için sklearn.neural_network kütüphanesini içe aktarmaktır. **MLPClassifier (hidden_layer_sizes=(200,100,50),max_iter=300,batch_size=16,learning_rate_init=0.001, activation = 'relu',solver='sgd')** fonksiyonunu kullanarak bir çok katmanlı algılayıcı nesnesi oluşturulmaktadır. İlk parametre, hidden_layer_sizes gizli katmanların boyutunu ayarlamak için kullanılır. Sinir ağı için katman ve düğüm sayısını seçmek için standart bir formül yoktur. En iyi yol, farklı kombinasyonları denemek ve neyin en iyi olduğunu görmektir. İkinci parametre MLPClassifier, sinir ağınızın yürütülmesini istediğiniz yineleme sayısını belirtir. Varsayılan olarak 'relu' aktivasyon fonksiyonu kullanılır. Ancak, sırasıyla activation ve solver parametrelerini kullanarak bu işlevler değiştirebilebilir. Bu çalışmada relu, tanh, logistic ve identity aktivasyon fonsiyonları; adam, relu ve sgd optimizasyon algoritmaları kullanılarak MLP modelleri oluşturulmuştur. learning_rate_init kullanılan ilk öğrenme oranıdır. Ağırlıkları güncelleme adım boyutunu kontrol eder. Batch_size ise mini-batch boyutudur. Daha sonra oluşturulan model fit() metodu kullanarak eğitilmektedir.

3.2.2.2. Hiper Parametreler

Makine öğrenmesinde kullanılan yönteme göre model tasarlanırken modeli tasarlayan kişinin karar vermesi gereken bazı parametreler vardır. Örneğin KNN sınıflandırmasında k değerinin ne olacağı model tasarlanırken belirlenir. Modelin yüksek başarım sağlaması için en uygun hiper parametreler seçilmelidir.

a) Mini Parti Boyutu(Mini-batch): Derin öğrenme modellerinde, veri setinde bulunan tüm verileri aynı anda işlemek maliyetli bir iştir. Bu nedenden dolayı eğitim veri kümesini model hatasını hesaplamak ve model katsayılarını güncellemek için veri seti parçalar halinde işlenir. Birden fazla girdinin parçalara ayrılarak parçalar halinde işlenmesine mini-batch denir (Anonim, 2019).

b) Parti boyutu(Batch Size): Model parametrelerini güncellemeden önce üzerinde çalışılacak örnek sayısını tanımlayan bir hiperparametredir (Jason Brownlee, 2016).

c) Öğrenme Oranı (Learning Rate) : Model ağırlıkları her güncellendiğinde tahmini hataya yanıt olarak modeli ne kadar değiştireceğini kontrol eden bir hiperparametredir. Öğrenme hızı, modelin soruna ne kadar hızlı adapte olduğunu kontrol eder. Model geliştirilirken Öğrenme hızının yüksek verilmesi salınıma neden olacaktır. Küçük olması durumunda ise küçük adımlarla ilerleyeceği için eğitimin uzun sürmesine neden olacaktır. (Jason Brownlee, 2016)

d) Optimizasyon Algoritması: Bir *f(x)* algoritması verildiğinde, bir optimizasyon algoritması *f(x)* değerini en aza indirmeye veya en üst düzeye çıkarmaya yardımcı olur. Modelin eğitilmesi sırasında optimum değeri bulmak için optimizasyon algoritmaları kullanılmaktadır. Yaygın olarak kullanılan optimizasyon algoritmaları adagrad, sgd, adadelta, adam, nadam, relu, adamax' dır (Sebastian Ruder, 2017).

e) Eğitim Tur (Epoch): Tüm veri setinin modelden ağırlıkların güncellenmesi ile bir kere gidip gelmesine epoch denir. Eğitim Turunun arttırılması eğitim verisini ezberlemeye başlamasına eğitim turunun az olması ise istenilen performansa ulaşmamasına neden olabilir (Anonim, 2019).

f) Seyreltme (Dropout): Seyreltme rastgele seçilen belirli nöron setinin eğitim aşaması sırasında göz ardı edilen birimleri (yani nöronları) ifade eder. Seyreltmenin kullanılmasındaki en önemli faktör ise Aşırı Öğrenme (overfitting) engellemesidir.

3.2.3. Değerlendirme Metrikleri

Makine öğrenmesinde kullanılan modelin performansını derğerlendirmek için bazı ölçütler vardır. Tasarlanan modellerin performansını değerlendirmek için kullanılan yöntemler aşağıda açıklanmaktadır.

a)Doğruluk, modelin neredeyse her çıktısı doğruluk kullanılarak değerlendirilir. Doğruluk, modelin kaç vakanın doğru olarak tanımlandığının ölçümüdür. Doğruluk ne kadar yüksek olursa, model o kadar iyidir. Deklem 3.4 ‘de gösterildiği gibi doğruluk ölçütü, doğru tahminlerin değerlendirilen toplam örnek sayısına oranını ölçer (Hossin vd., 2015).

True positive (tp), False negative (fn) ,False positive (fp), True negative (tn)

$$\frac{tp + tn}{tp + fp + tn + fn} \tag{3.4}$$

b) Duyarlılık (Recall(r)) Deklem 3.5 de gösterilmektedir. Gerçek pozitif sınıfı ne kadar yakaladığını belirleyen bir ölçümdür (Hossin vd., 2015).

$$\frac{tp}{tp + tn} \tag{3.5}$$

c) Karışıklık matrisi (Confusion Matrix), sınıflandırma modelinin performansını değerlendirmek için kullanılan bir tablodur. Bir algoritmanın performansını da görselleştirebilirsiniz. Bir karışıklık matrisinin temeli, doğru ve yanlış tahminlerin sayısının sınıf açısından özetlenmesidir.

d) Hassasiyet (Sensitivity (sn)) Deklem 3.6 da gösterilmektedir. Pozitif tahminin ne kadarının gerçekten pozitif durumda olduğunu belirleyen bir ölçümdür (Hossin vd., 2015).

$$\frac{tp}{tp + fn} \tag{3.6}$$

e) F1-socore, Deklem 3.8 de gösterildiği gibi hassasiyet ve duyarlılığın harmonik ortalamasıdır . F1 puanı ne kadar yüksek olursa, modelin performansı o kadar iyidir (Hossin vd., 2015).

$$\frac{2 * p * r}{p + r} \tag{3.7}$$

f) Kesinlik (Precision) , Deklem 3.9 da görüldüğü şekilde pozitif bir sınıftaki toplam tahmin edilen örüntülerden doğru tahmin edilen pozitif örüntüleri ölçmek için kullanılır (Hossin vd., 2015).

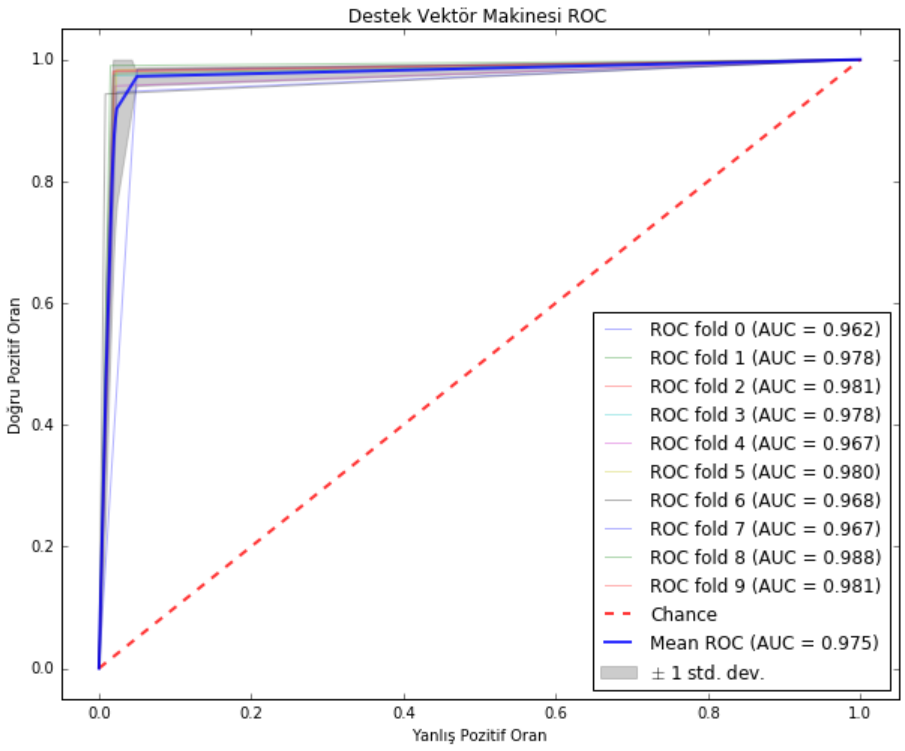
$$\frac{tp}{tp + fp} \tag{3.8}$$

g) ROC Curve, bir sınıflandırıcının olası tüm eşikler üzerindeki performansını özetleyen yaygın olarak kullanılan bir grafiktir. True Positve Rate (TPR) ile False Positive Rate (FTR) metriklerinin x ve y eksenlerine yerleştirilmesi ile çizginin altında kalan alanın hesaplanmasıdır (AUC) (Anonim,2019).

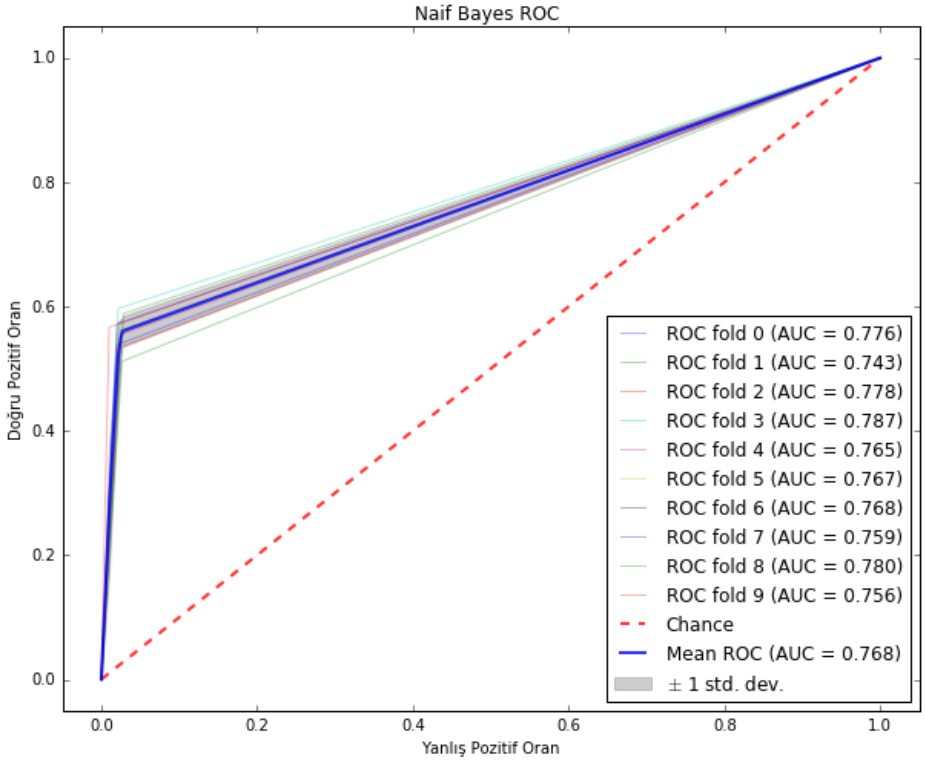
4. BULGULAR VE TARTIŞMA

4.1. Geleneksel Makine Öğrenmesi Yöntemleri Testi

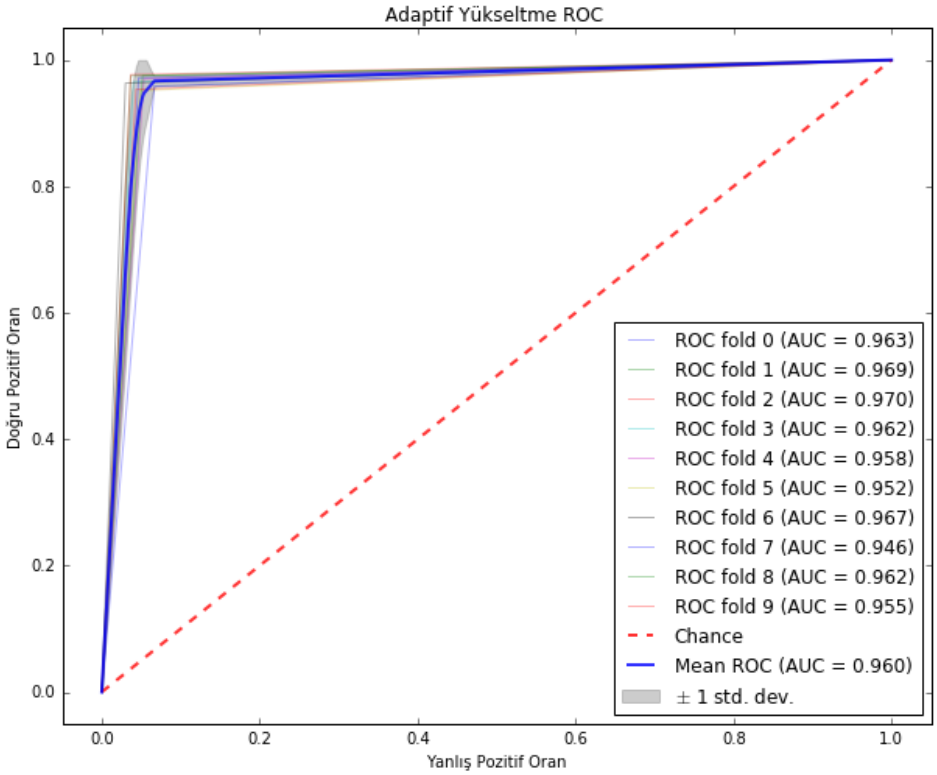
Test için ayrılan veri seti üzerinde oluşturulan modeller scikit-learn kütüphanesinin predit() metodu ile denenmiştir. Test sonuçları değerlendirilirken doğruluk (accuracy), kesinlik (precision), F1 Skoru ve duyarlılık (recall) ve ROC Curve değerlendirme metriği olarak kullanılmıştır. Şekil 4.1. 'den Şekil 4.9.'a kadar kullanılan geleneksel makine öğrenmesi yöntemlerinin ROC eğri grafikleri gösterilmektedir.



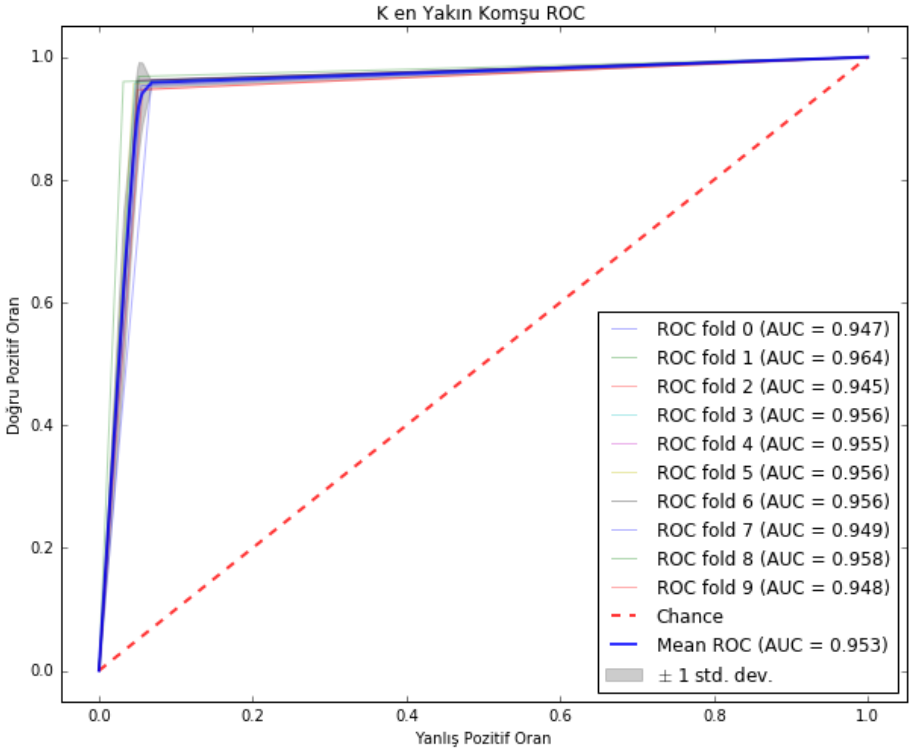
Şekil 4.1. Destek Vektör Makinesi ROC Sonuçları



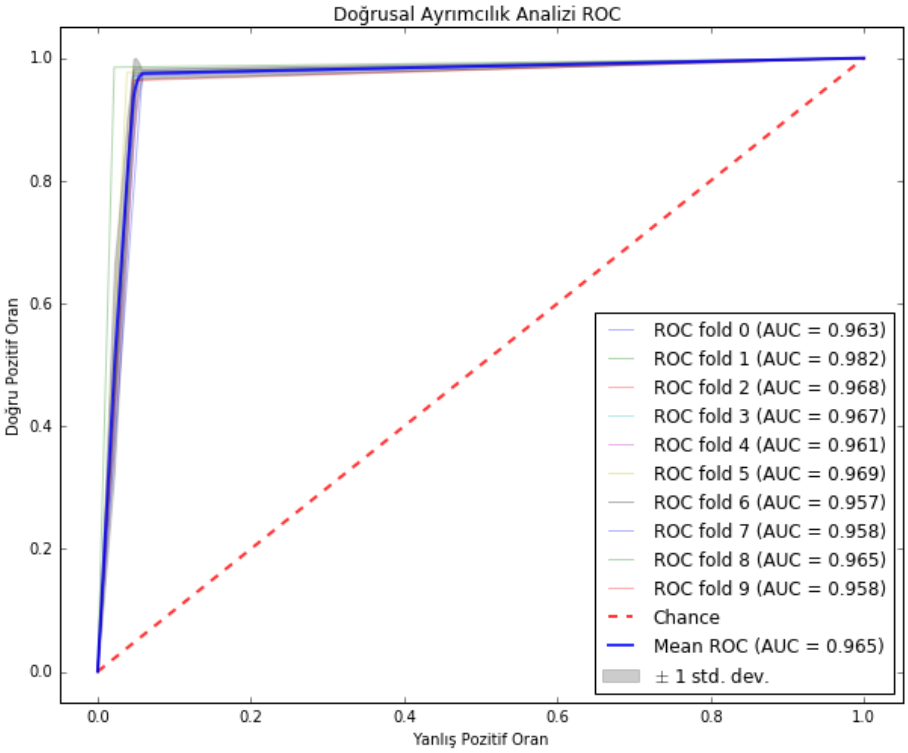
Şekil 4.2. Naif Bayes ROC Sonuçları



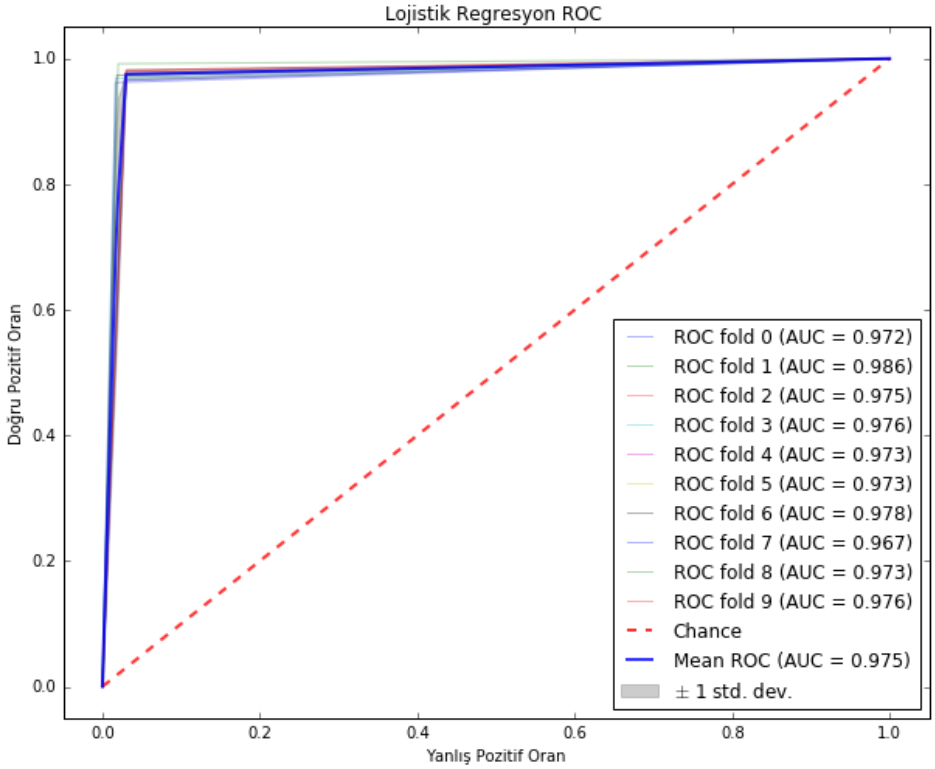
Şekil 4.3. Adaptif Yükseltme ROC Sonuçları



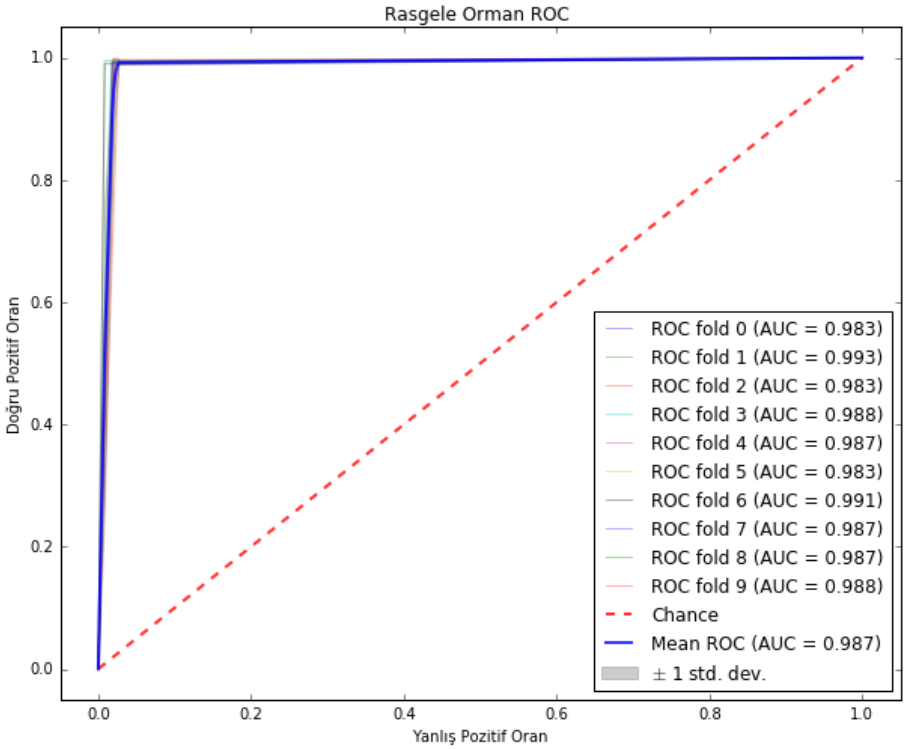
Şekil 4.4. K en Yakın Komşu ROC Sonuçları



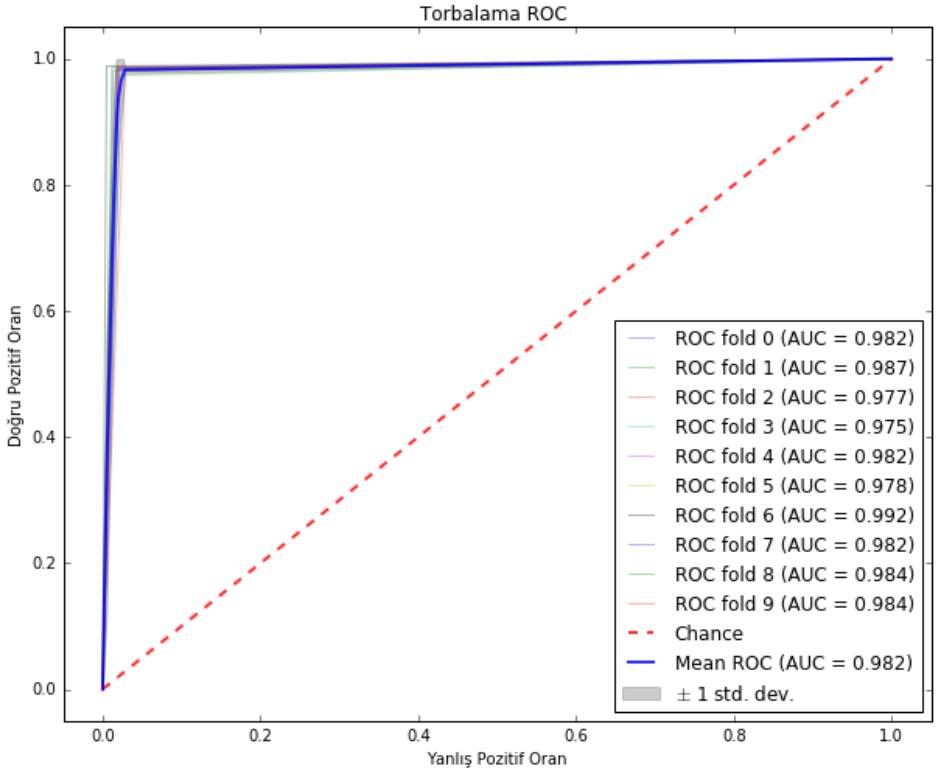
Şekil 4.5. Doğrusal Ayrımcılık Analizi ROC Sonuçları



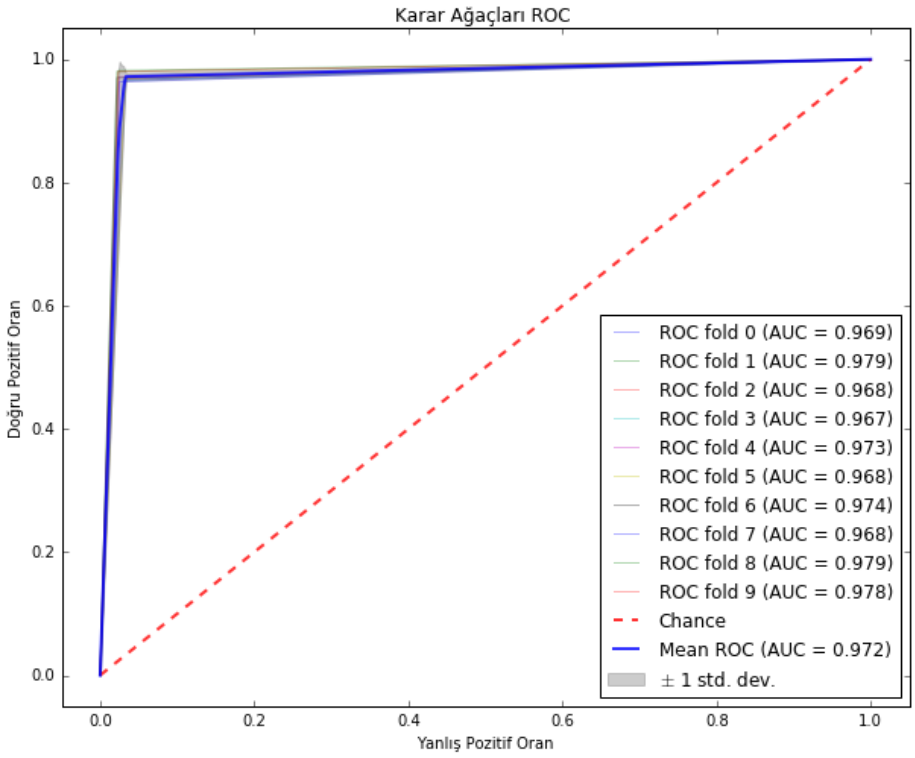
Şekil 4.6. Lojistik Regresyon ROC Sonuçları



Şekil 4.7. Rasgele Orman ROC Sonuçları



Şekil 4.8. Torbalama ROC Sonuçları



Şekil 4.9. Karar Ağaçları ROC Sonuçları

Çizelge 4.1. Geleneksel Makine Öğrenmesi Yöntem Sonuçları

Model	Doğruluk(accuracy)	Kesinlik(precision)	Duyarlılık(recall)	F1 Skor(F1-Score)
DT (CART)	%97.24	%97.31	%97.16	%97.24
BA	%98.24	%98.25	%98.23	%98.24
AB	%96.03	%95.52	%96.60	%96.03
K-NN	%95.34	%94.98	%95.75	%95.34
LDA	%96.46	%95.53	%97.50	%96.46
NB	%76.79	%96.20	%55.78	%76.79
SVM	%97.48	%97.81	%97.17	%97.48
LR	%97.49	%97.50	%97.48	%97.49
RF	**%98.70**	**%98.20**	**%99.23**	**%98.70**

Geleneksel Makine Öğrenmesi algoritmaları ile yapılan deneylerin doğruluk (accuracy) ölçüm sonuçları Çizelge 4.1. gösterilmektedir. Doğruluk (accuracy) ölçüm sonucunda yüzde 98.72 ile Rastgele Orman (Random Forest) algoritması en başarılı sonucu vermiştir.

4.2. Derin Öğrenme Yöntemleri Testi

a)Çok Katmanlı Algılayıcı (MLP- Multi Layer Perceptron)

Çizelge 4.2. MLP Sonuçları

Gizli Katman	Aktivasyon Fonksiyonu	Optimi-zasyon Yöntemi	Parti Boyutu	Öğrenme Katsayısı	F1 Skor	Duyarlılık	Kesinlik	Doğruluk
150	relu	adam	128	0.001	0.987416	0.987717	0.987126	0.987416
150	**relu**	**adam**	**64**	**0.001**	**0.988666**	**0.989647**	**0.987695**	**0.988666**
150	relu	adam	256	0.001	0.987833	0.988855	0.986829	0.987833

Çizelge 4.2. MLP Sonuçları (devam)

150	relu	sgd	256	0.001	0.9728333	0.979642	0.966476	0.972833
150	relu	sgd	128	0.001	0.978916	0.983140	0.974873	0.9789166
150	relu	sgd	64	0.001	0.9821666	0.984804	0.979586	0.982166
150	relu	adam	32	0.001	0.988249	0.9896470	0.986872	0.988240
150	**relu**	**adam**	**16**	**0.001**	**0.9887499**	**0.9893210**	**0.988142**	**0.988749**
150	**relu**	**adam**	**8**	**0.001**	**0.9896666**	**0.9916767**	**0.987701**	**0.989666**
150	relu	adam	16	0.1	0.9716666	0.9775160	0.966837	0.971666
100	relu	adam	8	0.001	0.987333	0.988898	0.985833	0.987333
200	relu	adam	8	0.001	0.988666	0.988650	0.988635	0.988666
150	tanh	adam	16	0.001	0.9874166	0.9876562	0.987135	0.987416
150	tahn	sgd	8	0.001	0.9862499	0.9876969	0.984851	0.986249
150	tahn	sgd	512	0.002	0.9716666	0.9771140	0.966565	0.971666
150	tahn	adam	8	0.002	0.987416	0.988665	0.986182	0.987416
150	relu	lbfgs	16	0.001	0.986000	0.986506	0.985482	0.986000
200	relu	lbfgs	16	0.001	0.983666	0.983674	0.983627	0.983666
150	relu	adam	4	0.001	0.9877500	0.987542	0.98795	0.987750
150	**relu**	**sgd**	**16**	**0.001**	**0.987833**	**0.988856**	**0.986857**	**0.987833**
150	**relu**	**sgd**	**8**	**0.001**	**0.987833**	**0.989344**	**0.986382**	**0.987833**
150	logistic	adam	512	0.001	0.979666	0.981346	0.978053	0.979666
150	logistic	adam	64	0.001	0.986416	0.9874928	0.985293	0.986416
150	logistic	sgd	64	0.001	0.971416	0.976441	0.966691	0.971416
150	logistic	adam	32	0.001	0.989175	0.987583	0.986013	0.987583
150	identity	adam	64	0.001	0.977416	0.978476	0.976371	0.977416
150	identity	adam	32	0.001	0.978083	0.981825	0.974481	0.978083
150	identity	adam	8	0.001	0.978583	0.980865	0.976394	0.978583
150	identity	adam	4	0.001	0.977916	0.982870	0.973185	0.977916
150	identity	adam	2	0.001	0.977500	0.977826	0.977195	0.977500

Veri seti üzerinde MLP Modelleri denenmiştir. MLP Modelinin hiperparametreler üzerinde yapılan deney sonuçları Çizelge 4.2. de gösterilmektedir. Veri seti üzerinde MLP modeli ile hiper parametreler ile yapılan deneyler sonucunda relu aktivasyon fonksiyonunun model üzerinde daha iyi çalıştığı görülmüştür. Optimizasyon algoritmalarından en iyi sonucu adam algoritması vermiştir. Mini-bach sayısı 8 olduğunda model en iyi performansı vermiştir. Test sonuçları F1 Skor, duyarlılık (recall), kesinlik (precision), doğruluk (accuracy) olmak üzere 4 değerlendirme metriği ile ölçülmüştür. En yüksek başarım oranı yüzde 98.96 doğruluk (accuracy) oranıdır.

b) Derin İnanç Ağları (DBN- Deep Belief Networks)

Çizelge 4.3. DBN Sonuçları

Gizli Katman	**Öğrenme Katsayısı**	**RBM Öğrenme Katsayısı**	**RBM Devir Sayısı**	**Seyreltme Değeri**	**Parti Boyutu**	**Doğruluk**	**F1 skor**	**Kesinlik**	**Duyarlılık**
100	0.05	0.1	4	0.1	512	0.978916	0.978916	0.973781	0.984317
150	0.1	0.05	10	0.1	128	0.984833	0.984833	0.981803	0.987955
150	0.1	0.05	10	0.1	64	0.987833	0.987833	0.985375	0.990375
150	0.1	0.05	8	0.1	64	0.984333	0.984333	0.978542	0.990359
200	0.1	0.1	8	0.1	128	0.988249	0.988249	0.98552	0.9910525
150	0.1	0.01	8	0.1	128	0.987333	0.987333	0.984684	0.990082
150	**0.1**	**0.1**	**10**	**0.1**	**128**	**0.989166**	**0.9891666**	**0.986083**	**0.992355**
200	0.1	0.1	10	0.1	128	0.988333	0.988333	0.984267	0.9925272
150	0.1	0.5	10	0.1	128	0.985083	0.9850833	0.982370	0.9877882
100	0.1	0.1	8	0.1	128	0.988749	0.9887499	0.985581	0.9920464
100	**0.1**	**0.1**	**8**	**0.1**	**64**	**0.989583**	**0.989583**	**0.986882**	**0.9923933**
100	0.1	0.1	8	0.1	512	0.984166	0.984166	0.97826	0.990363
100	0.1	0.1	4	0.1	64	0.988750	0.988750	0.985728	0.991838

Çizelge 4.3. DBN Sonuçları (devam)

100	0.1	0.1	12	0.1	64	0.9884999	0.988499	0.985859	0.991208
150	0.1	0.1	10	0.1	64	0.988916	0.9889166	0.986519	0.9913903
150	0.2	0.05	8	0.1	128	0.988250	0.988250	0.987013	0.9895933
150	0.1	0.05	12	0.5	128	0.97875	0.97875	0.964558	0.9939978
150	**0.5**	**0.05**	**10**	**0.1**	**32**	**0.989416**	**0.9894166**	**0.987995**	**0.990860**
150	0.5	0.1	10	0.01	128	0.9838333	0.9838333	0.982925	0.9848129
150	0.1	0.2	10	0.1	128	0.9878333	0.987833	0.984586	0.99122458
150	**0.1**	**0.1**	**2**	**0.1**	**64**	**0.989250**	**0.98925**	**0.986186**	**0.9923684**
150	**0.1**	**0.1**	**2**	**0.1**	**32**	**0.9899166**	**0.9899166**	**0.988235**	**0.9917081**
150	**0.1**	**0.1**	**2**	**0.1**	**16**	**0.990083**	**0.9900833**	**0.989154**	**0.991044**
150	**0.1**	**0.1**	**2**	**0.1**	**8**	**0.990250**	**0.990250**	**0.987539**	**0.993026**

Veri seti üzerinde DBN Modelleri denenmiştir. DBN Modelinin hiper parametreler üzerinde yapılan deney sonuçları Çizelge 4.3. de gösterilmektedir. Veri seti üzerinde DBN modeli ile hiper parametreler ile yapılan deneylerde relu aktivasyon fonksiyonu ve sgd optimizasyon algoritması kullanılmıştır. Öğrenme Katsayısı ve RBM Öğrenme Katsayısı 0.1 olduğunda modelin daha yüksek doğruluk oranı verdiği görülmektedir. Bach sayısı 8 olduğunda model en iyi performansı vermiştir. Test sonuçları doğruluk (accuracy) değerlendirme metriği ile ölçülmüştür. En yüksek başarı oranı yüzde 99.00 doğruluk oranıdır.

5. SONUÇ VE ÖNERİLER

Bu çalışmada uygulamaların statik analiz yöntemi ile elde edilen özniteliklerden yararlanılarak Android mobil işletim sistemi için makine öğrenmesi yöntemleriyle zararlı yazılım tespit yöntemleri geliştirilmiştir. Tezin amacı geleneksel yöntemler ile derin öğrenme yöntemlerinin başarım oranlarını karşılaştırmak ve uygulamanın zararlı olup olmadığını tespit edebilmek için en yüksek başarım oranına sahip modeli geliştirebilmektir. İlk olarak geliştirilen modellerde literatürde en çok kullanılan geleneksel makine öğrenmesi yaklaşımları kullanılmıştır. Kullanılan yöntemler DT (CART) , BA, AB, K-NN, LDA, NB, SVM, LR, RF dir. Doğruluk, kesinlik, duyarlılık ve f1 skor kullanılarak değerlendirilen test sonuçlarında %76.79 doğruluk oranı ile en kötü sonucu naif bayes algoritması vermiştir. Geleneksel yöntemlerle geliştirilen modelerde rastgele orman algoritması %98.70 doğruluk oranı ile en yüksek başarıya sahiptir.

MLP modeli geliştirilirken kullanılan hiper parametreler denenerek başarı oranı en iyi olan kombinasyonlar ile testler gerçekleştirilmiştir. Gizli katman sayısı, aktivasyon fonksiyonu, optimizasyon yöntemi, öğrenme katsayısı değişen hiper parametrelerimizi oluşturmaktadır. MLP modeli geliştirilirken relu, tahn, logistic, identity aktivasyon fonksiyonları kullanılmıştır. Yapılan eğitimler sonucunda relu aktivasyon fonksiyonunun model üzerinde daha iyi çalıştığı görülmüştür. 150 , 100, 200 gizli katman sayılarında 100 ve 200 lerde doğruluk oranının düştüğü gözlemlenmiştir. Bu yüzden gizili katman saysısı 150 de tutulmuştur. Adam, sgd, lbfgs optimizasyon yöntemlerinden adam diğerlerine oranla daha başarılı olduğu görülmüştür. Parti Boyutu olarak 256, 128, 64, 32, 16 ve 8 olarak eğitimler yapılmıştır. Parti sayısının küçültmek başarı oranını artırmaktadır fakat 8 olduğu durumda ise başarı oranı tekrar düşmektedir. Geliştirilen modelde parti sayısı 16 olarak verilmiştir. Hiper parametrelerle yapılan deneyler sonucunda doğruluk metriği ile yapılan değerlendirmede %98.96 başarı oranına sahiptir.

DBN modeli geliştirilirken kullanılan hiper parametreler; gizli katman sayısı, öğrenme katsayısı, RBM öğrenme katsayısı, aktivasyon fonksiyonu, optimizasyon algoritması, RBM devir sayısı, seyreltme değeri ve parti boyutudur. Geliştirilen modelde relu aktivasyon fonksiyonu ve sgd optimizasyon algoritması kullanılmıştır. Gizli katman

sayısı olarak 100, 120, 150, 180, 200 model üzerinde denenmiştir. Gizli katman sayısı 150 olduğunda model üzerinde daha iyi çalıştığı gözlemlenmektedir. Öğrenme katsayısı olarak 0.1 , 0.05 ,0.2 ,0.5 değerleri arasından 0.1 öğrenme katsayısı; RBM öğrenme katsayısı olarak 0.1 , 0.05 ,0.2 ,0.5 değerleri arasından 0.1 RBM öğrenme katsayısı; seyreltme değeri olarak 0.1 , 0.05 , 0.01 değerleri arasından 0.1 seyreltme değeri model üzerinde daha iyi sonuç vermiştir. RBM devir sayısı olarak 12, 10, 8, 4, 2 değerleri model üzerinde test edilmiştir. Model üzerinde yapılan hiper parametre kombinasyonlarında en yüksek başarı oranı doğruluk değerlendirme metriğine göre %99.00' dır. Çizelge 5.1. 'de gösterildiği gibi zararlı yazılım tespit sistemlerinde derin öğrenme yöntemi ile geliştirilen sistemlerin daha yüksek başarı oranına sahip olduğu görülmektedir.

Çizelge 5.1. Zararlı Yazılım Tepit Sistemleri Test Sonuçları

Model	**Doğruluk(accuracy)**
DT (CART)	%97.24
BA	%98.24
AB	%96.03
K-NN	%95.34
LDA	%96.46
NB	%76.79
SVM	%97.48
LR	%97.49
RF	%98.70
DBN	**%99.00**
MLP	**%98.96**

Önerilen yöntemin başarı oranı yüksek olmasına rağmen dezavantajı ise bütün statik analiz yöntemlerinde olduğu gibi uygulama davranışlarını gözlemleyememesinden kaynaklı sonradan kod yükleme yapan zararlı yazılımlara karşı savunmasız olmasıdır.

Daha sonraki çalışmalarımızda, dinamik analiz yöntemi ile elde edilen özniteliklerde eklenerek hibrit analiz yöntemi ile sistemin dinamik davranışlarada duyarlı hale getirerek başarısı artırılmaya çalışılacaktır. Dinamik analiz ayrıca mobil uygulamalarda, özellikle Android kötü amaçlı yazılım algılamasında veri akışının izlenmesi amacıyla da önerilmektedir.

KAYNAKLAR DİZİNİ

Bernd van der Wielen ,2018 ,Insights into the 2.3 billion Android smartphones in use, https://newzoo.com/insights/articles/insights-into-the-2-3-billion-Android-smartphones-in-use-around-the-world/, erişim tarihi : 01/12/2018

J. Clement, 2018, Google play store: number of apps 2018 report — statista, https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ , erişim tarihi : 01/12/2018

Anonim, 2018, Mcafee mobile threat report q1, 2018, https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2018.pdf, erişim tarihi : 03/12/2018

Anonim, 2018 “Mcafee labs threats report september 2018”https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-sep-2018.pdf, erişim tarihi: 03/12/2018

Schmeelk, S., Yang, J., & Aho, A., 2015, Android malware static analysis techniques. In Proceedings of the 10th Annual Cyber and Information Security Research Conference (p. 5). ACM.

Balakrishnama, S., & Ganapathiraju, A., 1998, Linear discriminant analysis-a brief tutorial. Institute for Signal and information Processing, 18, 1-8.

G. E. Hinton, S. Osindero, and Y.-W. Teh, 2006, A fast learning algorithm for deep belief nets,” Neural computation, vol. 18, no. 7, pp. 1527–1554,

M.el Khamlichi, 2015, The History Of Android, https://www.unixmen.com/the-history-of-Android/ ,erişim tarihi : 01/08/2019

Jason Brownlee, 2016, How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras, https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/, erişim tarihi: 25.12.2019

Anonim, 2019, Android Licenses ,https://source.Android.com/setup/start/licenses, erişim tarihi 11/010/2019

Anonim, 2019, Distribution dashboard , https://developer.Android.com/about/dashboards , erişim tarihi 11/010/2019

Anonim, 2019, Platform Architecture, https://developer.Android.com/guide/platform, erişim tarihi 11/010/2019

Sebastian Ruder, 2017, “An overview of gradient descent optimization algorithms”, erişim tarihi: 21.12.2019

KAYNAKLAR DİZİNİ (devam)

Anonim, 2019, Permissions overview, https://developer.Android.com/guide/topics /permissions/overview, erişim tarihi: 21/05/2019

Anonim, 2019, Google Inc. Application Fundamentals - Android Developers. https://developer.Android.com/guide/components/fundamentals.html, 2019. erişim tarihi: 21/05/2019

Anonim, 2019 ,Dalvik bytecode, https://source.Android.com/devices/tech/dalvik /dalvik-bytecode , erişim tarihi: 21/05/2019

Moubayed, A., Injadat, M., Nassif, A. B., Lutfiyya, H., Shami, A., 2018, E-Learning: Challenges and Research Opportunities Using Machine Learning & Data Analytics. IEEE Access, 6, 39117–39138, doi: 10.1109/access.2018.2851790

Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M., 2014, Android security: a survey of issues, malware penetration, and defenses. IEEE communications surveys & tutorials, 17(2), 998-1022.

Breiman, L., 1996, Bagging predictors. Machine learning, 24(2), 123-140.

N. Peiravian and X. Zhu, 2013, Machine learning for Android malware detection using permission and api calls, in Tools with Artificial Intelligence (ICTAI), IEEE 25th International Conference on. IEEE, pp. 300–305.

D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, 2012, Droidmat: Android malware detection through manifest and api calls tracing, in Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on. IEEE, pp. 62–69.

Keller, J. M., Gray, M. R., & Givens, J. A., 1985, A fuzzy k-nearest neighbor algorithm. IEEE transactions on systems, man, and cybernetics, (4), 580-585.

Anonim, 2019, contagio mobile, http://contagiominidump.blogspot.com/ , erişim tarihi: 21/12/2018

N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupe, 2017, Deep Android malware detection, in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. ACM, pp. 301–308.

M. Tim Jones, 2017, Deep learning architectures, https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/ , erişim tarihi: 01/12/2019

Anonim, 2019, Matplotlib , https://matplotlib.org/index.html , erişim tarihi: 01/12/2019

KAYNAKLAR DİZİNİ (devam)

Shen, T., Zhongyang, Y., Xin, Z., 2014, Detect Android Malware Variants using Component Based Topology Graph, 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, 406-413.

Anonim, 2015, Android malware genome project, http://www.malgenomeproject.org/, erişim tarihi: 01/12/2018

Anonim, 2015, Neural Network Hyperparameters, http://colinraffel.com/wiki/neural_ network_hyperparameters, erişim tarihi: 21.09.2019

S. Hou, A. Saas, Y. Ye, and L. Chen, 2016, Droiddelver: An Android malware detection system using deep belief network based on api call blocks, in International Conference on Web-Age Information Management. Springer, pp. 54–66.

D. Zhu, H. Jin, Y. Yang, D. Wu, and W. Chen, 2017, Deepflow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data, in Computers and Communications (ISCC), 2017 IEEE Symposium on. IEEE, pp. 438–443.

X. Su, D. Zhang, W. Li, and K. Zhao, 2016 ,A deep learning approach to Android malware feature learning and detection," in Trustcom/BigDataSE/I SPA, 2016 IEEE. IEEE, pp. 244–251.

A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, 2014, Mobile malware detection through analysis of deviations in application network behavior, Computers & Security, vol. 43, pp. 1–18.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams, 2016, Learning internal representations by error propagation, California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep.

Z. Yuan, Y. Lu, and Y. Xue, 2016, Droiddetector: Android malware characterization and detection using deep learning, Tsinghua Science and Technology, vol. 21, no. 1,pp. 114–123.

Anonim, 2019, App manifest overview Android developers, https://developer.Android.com /guide/topics/manifest/manifest-intro, erişim tarihi: 14/12/2019

Anonim, 2019, Intents, https://developer.Android.com/reference/Android/content/Intent erişim tarihi: 14/12/2019

Anonim, 2019, Keras, https://keras.io/ , erişim tarihi 01/12/2019

KAYNAKLAR DİZİNİ (devam)

D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, 2014, Drebin: Effective and explainable detection of Android malware in your pocket. in Ndss, vol. 14, 2014, pp. 23–26.

Nikoli´c and F. Spoto., 2014, Reachability analysis of program variables. ACM Trans. Program. Lang. Syst., 35(4),

Hossin, M., & Sulaiman, M. N., 2015, A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process, 5(2), 1.

Freund, Y., & Schapire, R. E., 1996, Schapire R: Experiments with a new boosting algorithm. In In: Thirteenth International Conference on ML.

Anonim, 2019, Receiver Operating Characteristic (ROC) https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html?highlight=roc%20curve, erişim tarihi:14/12/2019

Anonim, 2019, App manifest overview Android developers, https://developer.Android.com /guide/topics /manifest/manifest-intro,

Anonim, 2019 ,Dex2jar, https://github.com/pxb1988/dex2jar, erişim tarihi: 01/12/2019

Anonim, 2019, Tensorflow, https://www.tensorflow.org/about/, erişim tarihi: 01/12/2019

Anonim, 2019, Numpy, https://numpy.org, erişim tarihi: 01/12/2019

Anonim, 2019, Pandas, Python Data Analysis Library , https://pandas.pydata.org/, erişim tarihi: 01/12/2019

Anonim, 2012, Drebin dataset , https://www.sec.cs.tu-bs.de/~danarp/drebin/, erişim tarihi: 01/03/2018

Anonim, 2019, scikit-learn , https://scikit-learn.org/stable/ , erişim tarihi:01/12/2019

S. O'Dea, 2020, https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/

J. Clement, 2020, https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/