T.C.
YALOVA ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DİSİPLİNLER ARASI
ADLİ BİLİŞİM ANABİLİM DALI

ANDROID YAZILIMLARDA YAPAY ZEKA DESTEKLİ ZARARLI YAZILIM TESPİTİ ve PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

FATİH BULDUR

DANIŞMAN: PROF.DR. MURAT GÖK

YALOVA
TEMMUZ 2023

T.C.
YALOVA ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ADLİ BİLİŞİM ANABİLİM DALI
DİSİPLİNLER ARASI

ANDROID YAZILIMLARDA YAPAY ZEKA DESTEKLİ ZARARLI YAZILIM TESPİTİ ve PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

FATİH BULDUR
205113001

DANIŞMAN: PROF.DR. MURAT GÖK

YALOVA
TEMMUZ 2023

ETİK BEYAN

Yalova Üniversitesi Lisansüstü Eğitim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım " Android Yazılımlarda Yapay Zeka Destekli Zararlı Yazılım Tespiti ve Performans Analizi" başlıklı bu tez çalışmasında; tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi, tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu, tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu tezde sunduğum çalışmanın özgün olduğunu bildirir, aksinin tespiti halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi taahhüt ve beyan ederim.

İmza

FATİH BULDUR

ÖNSÖZ

Araştırma sürecim boyunca beni yönlendirdiği ve desteklediği için tez hocam Prof. Dr. Murat Gök' e teşekkür ederim. Tez yazım sürecinde bana maddi ve manevi her türlü desteği sağlayan eşim Merve Buldur' a ve kızım Neva Buldur' a şükranlarımı sunarım.

Temmuz- 2023 Fatih BULDUR

İÇİNDEKİLER

KISALTMALAR

AÇZ	:Android Çalışma Zamanı
API	:Uygulama Programlama Arayüzü
APK	:Android Paket Kiti
ÇKA	:Çok Katmanlı Algılayıcı
DSK	:Donanım Soyutlama Katmanı
DVM	:Destek Vektör Makineleri
GA	:Genetik Algoritma
k-EYK	:K-En Yakın Komşuluğu
ML	:Makine Öğrenmesi
NB	:Naive Bayes
PID	:İşlem Kimliği
RO	:Rasgele Orman
UID	:Kullanıcı Kimliği

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

ANDROID YAZILIMLARDA YAPAY ZEKA DESTEKLİ ZARARLI YAZILIM TESPİTİ ve PERFORMANS ANALİZİ

ÖZET

Günümüzde, akıllı cihazların hayatımızın ayrılmaz bir parçası haline gelmesi ile birlikte, bu cihazlarda ağırlıklı olarak kullanılan Android işletim sistemi de giderek yaygınlaşmaktadır. Bu duruma paralel olarak siber saldırganların ilgisi Android tabanlı uygulamalara yönelmektedir. Kaspersky firmasının 2022 mobil tehdit raporuna göre, Android ortamında son bir yıl içerisinde 1,5 milyondan fazla zararlı yazılım tespit edilmiştir. Bu nedenle zararlı yazılım tespiti, siber suçlar ve siber güvenlik alanlarında en önemli konulardan biri haline gelmiştir. Makine öğrenmesi temelli çalışmalar, zararlı yazılım tespitinde büyük önem taşımaktadır. Bu tez çalışmasının amacı güncel tehditleri içeren özgün bir veri seti üzerinde statik analiz yöntemi kullanarak elde edilen özelliklerin, makine öğrenmesi yöntemleri ile zararlı yazılım tespitini yapmak ve performans çıktılarını kıyaslamaktır. Kullandığımız veri seti 2022 yılının 3. çeyreğinde yayınlanmış 3098 adet zararlı, 11000 adet tehdit içermeyen Android kurulum dosyasından oluşmaktadır. Toplamda 14098 adet kurulum dosyasından statik analiz yöntemleri ile 528 adet Android işletim sistemine özel ayırt edici izin özelliği çıkarılarak veri matrisi elde edilmiştir. İlk olarak veri setimiz üzerinde destek vektör makineleri, k-en yakın komşuluğu, rasgele orman, naif bayes, çok katmanlı algılayıcı ağ, LightGBM ve XGBoost sınıflandırıcıları ile zararlı yazılım tespit işlemi gerçekleştirilmiştir. İkinci aşamada tespit başarımını artırabilmek için öznitelik seçim yöntemleri (genetik algoritma, karınca kolonisi, tepe tırmanma) ile belirtilen sınıflandırma algoritmaları test edilerek performans çıktıları incelenmiştir. LightGBM algoritması 0,98 doğruluk oranı 0,96 kesinlik, 0,90 duyarlılık değerleri ile en yüksek performansı sergilemiştir.

Anahtar kelimeler: Android, Zararlı Yazılım Tespiti, Sınıflandırma, Makine Öğrenmesi

AI-ASSISTED MALWARE DETECTION AND PERFORMANCE ANALYSIS IN ANDROID SOFTWARE

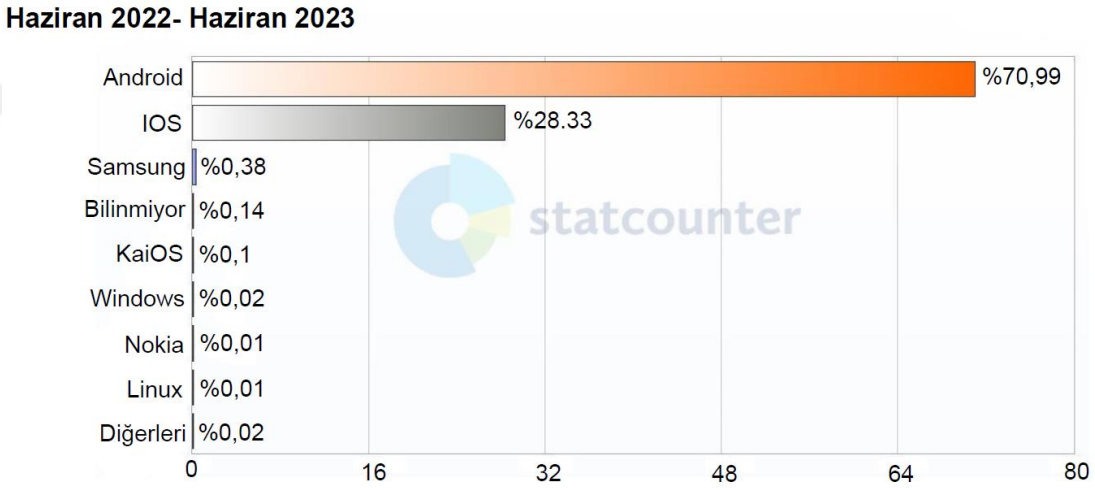
ABSTRACT

Nowadays, as smart devices have become an integral part of our lives, the android operating system, which is predominantly used in these devices, is becoming increasingly widespread. In parallel with this situation, the interest of cyber attackers is turning towards android-based applications. According to Kaspersky's 2022 mobile threat report, more than 1.5 million malware have been detected in the android environment in the last year. Therefore, malware detection has become one of the most important issues in the fields of cybercrime and cyber security. Machine learning based studies are of great importance in malware detection. The aim of this thesis is to perform malware detection using the features obtained through static analysis on an original dataset containing current threats and compare the performance outputs of various machine learning methods. The dataset used in this study consists of 3,098 malicious and 11,000 benign android installation files published in the 3rd quarter of 2022. In total, 528 distinctive permission features specific to the android operating system were extracted from 14098 installation files with static analysis methods and a data matrix was obtained. First, malware detection was performed on our dataset with support vector machines, k-nearest neighbor, random forest, naive bayes, multilayer perceptron network, LightGBM and XGBoost classifiers. In the second stage, in order to improve the detection performance, feature selection methods (genetic algorithm, ant colony, hill climbing) and the specified classification algorithms were tested and their performance outputs were analyzed. The LightGBM algorithm showed the highest performance with an accuracy of 0.98, precision of 0.96 and sensitivity of 0.90.

Keywords: Android, Malware Detection, Classification, Machine Learning

1. GİRİŞ

Mobil cihazların hayatımızın her alanında yoğun olarak kullanılması neticesinde Android işletim sistemine sahip cihazlarda son yıllarda büyük bir artış olmuştur. Statcounter isimli şirketin dünya genelinde Nisan 2022 – Nisan 2023 tarihlerini içeren mobil işletim sistemlerinin marketteki pazar payına bakıldığında her 10 mobil işletim sisteminden yaklaşık olarak 7'sinin Android işletim sistemli cihaz olduğu görülmektedir [1]. Bu artış kötü niyetli yazılımların büyük oranda Android platformunu hedef almasına neden olmuştur.



Şekil 1.1 Mobil işletim sistemlerinin pazar dağılımı

Zararlı yazılımlar bulunduğu cihazda izinsiz erişim elde etmek, kullanıcıya ait bilgileri çalmak, bankacılık işlemlerinde kullanılan finansal verileri ele geçirmek, kötü amaçlı sitelere ait reklamları göstermek, kullanıcıya ait verileri şifreleyerek erişilmez duruma getirmek gibi bir dizi zararlı eyleme neden olan yazılımlardır.

Android cihazların resmi uygulama marketi olan Google Uygulama Mağazası, otomatik zararlı yazılım taraması, insan gözüyle inceleme, geliştirici doğrulaması, güncelleme kontrolleri gibi önlemler ile marketinde bulunan yazılımları tarayarak zararlı yazılımlarla mücadele etmektedir. Ancak bu önlemler bile bazı zararlı yazılımların gözden kaçmasını engelleyememektedir. 2022 yılında Bitdefender güvenlik şirketinin yapmış olduğu incelemede Google Uygulama Mağazasında bulunan ve 2 milyona yakın indirilme sayısına sahip 35 uygulamanın kullanıcı verilerini ele geçirebilecek zararlı yazılım içerdiği ve reklam yayınladığını tespit

edilmiştir [2]. Ayrıca Android işletim sistemi doğası gereği uygulama mağazası dışında birçok marketten uygulama indirip yüklemeye imkân tanımaktadır. Bu nedenle popüler olarak kullanılan apkmirror , apkpure, aptoide,f-droid vb. birçok mobil uygulama sitesinden kullanıcı cihazlarına zararlı yazılımlar bulaşabilmektedir.

Son zamanlarda yapılan çalışmalar zararlı yazılımların tespitinde makine öğrenmesi yöntemlerinin önemli katkılar sağladığını ve tespit oranlarında ciddi iyileştirmeler sağladığını göstermiştir.

Bir Android kurulum dosyasını hangi kaynaktan elde etmiş olursak olalım, bu programın zararlı bir yazılım içerip içermediği konusunda bilgi sahibi olabilmemiz için analiz yöntemlerinden birini kullanmalıyız. Tespit aşamasında ise makine öğrenmesi destekli yaklaşımlar yüksek doğruluk ve performans sağlayacaktır.

1.1 Tezin Amacı

Bu tez çalışmasının amacı güncel tehditleri içeren Android kurulum dosyalarından tersine mühendislik yöntemleri ile elde edilen izin özellikleri değerlendirilerek, zararlı yazılım içerip içermediğini makine öğrenmesi yöntemleri kullanarak yüksek doğrulukla tahmin etmektir. Çalışmamızda özgün ve güncel veri setimiz üzerinde statik analiz metodu ile elde etmiş olduğumuz izin özelliklerini Genetik algoritma, karınca kolonisi optimizasyon tekniği ve tepe tırmanma algoritması kullanarak özellik sayısını optimize ettik. Daha sonra rasgele orman (RO), çok katmanlı algılayıcı ağ (ÇKA), k-En yakın komşuluk(k-EYK), destek vektör makineleri (DVM), naif bayes (NB) yöntemlerini kullanarak sınıflandırma çalışmalarını yaptık. Elde ettiğimiz sonuçların performanslarını iyileştirmek için son zamanlarda yaygın kullanıma sahip gradyan artırma tekniklerinden olan LightGBM ve XGBoost tekniklerini kullanarak performans çıktılarını kıyasladık.

1.2 Literatür Araştırması

Ünver ve arkadaşları (2020), her biri 9700 örneği (4850 zararsız ve 4850 kötücül örnek) içeren üç gri tonlamalı görüntü veri seti, Android kurulum dosyası arşivlerinin içeriğinden meydana getirmişlerdir. İlk veri seti, her Android uygulamasının Manifest.xml dosyasının bir gri tonlamalı görüntüye dönüştürülmesiyle oluşturulmuştur. İkinci görüntü veri seti, her Android uygulamasının DEX bayt kodu

dosyalarının bir gri tonlamalı görüntüye dönüştürülmesiyle oluşturulmuştur. Son veri seti görüntüleri, her Android uygulamasının Manifest.xml, DEX ve Resource.ARSC dosyalarının bir gri tonlamalı görüntüye dönüştürülmesiyle oluşturulmuştur. SIFT, SURF, KAZE ve ORB gibi dört farklı görüntü tabanlı yerel özellik ve Renk Histogramı, Haralick Doku ve Hu Momentleri gibi üç farklı görüntü tabanlı genel özellik de dahil olmak üzere, çeşitli makine öğrenme sınıflandırıcılarını eğitmek için çıkarılan farklı özellikler kullanmışlardır. Belirtilen bu özellikler ilk kez Android kötü amaçlı yazılım tespit alanında kullanılmıştır. Görüntü tabanlı yerel ve genel özellikler, çanta görsel kelimeler (BOVW) algoritması kullanılarak birden çok yerel özellik tanımlayıcısı vektöründen bir özellik vektörü elde etmek için kullanarak makine öğrenme sınıflandırıcılarına kaynak sağlamışlardır. Elde edilen yerel ve genel özellikler, Rastgele Orman, K-en Yakın Komşu, Karar Ağacı, Bagging, AdaBoost ve Gradyan Hızlandırma gibi altı farklı makine öğrenme sınıflandırıcısını eğitmek için kullanılmışlardır [3].

Sangal ve arkadaşları yaptıkları çalışmada CICInvesAndMal2019 veri kümesi üzerinde çeşitli makine öğrenimi tekniklerini analiz etmektedirler. Veri kümesi, 1522 uygulamanın izinler ve amaçlar gibi statik özelliklerini içermektedir. 396 kötü amaçlı yazılım ve 1126 zararsız uygulama içermektedir. Kötü amaçlı yazılımlar, reklam yazılımları, SMS Truva atları ve fidye yazılımları gibi farklı kategorilere ayrılmaktadır. Araştırma, veri ön işleme, özellik seçimi, sınıflandırma ve değerlendirme olmak üzere dört aşamayı içermektedir. Veri ön işleme aşamasında, eksik verilerin ve aykırı değerlerin ele alındığı, gereksiz verilerin çıkarıldığı belirtilmektedir. Özellik seçimi aşamasında, Android uygulama izinleri ve amaçları seçilerek analiz için kullanılan özellikler belirlenmektedir. Principal Component Analysis (PCA) adı verilen bir boyut indirgeme tekniği kullanılarak özellik sayısı azaltılmaktadır. Sınıflandırma aşamasında, Naif Bayes, Rasgele Orman, Karar Ağacı, K-en Yakın Komşuluk ve DVM gibi makine öğrenimi algoritmaları kullanılmaktadır. Son olarak, değerlendirme metrikleri olan doğruluk, hassasiyet, geri çağırma ve F1 skoru gibi performans ölçütleri kullanılarak sınıflandırma modellerinin etkinliği değerlendirilmektedir. Yaptıkları araştırmada Android sistemlerde kötü amaçlı yazılımlarının tespiti için makine öğrenimi tekniklerinin etkin bir şekilde kullanılabileceğini göstermektedir. Özellik seçimi ve boyut indirgeme teknikleri

kullanılarak daha az işlem gücüyle daha yüksek doğruluk elde edilebileceği sonucuna varılmıştır [4].

Elayan ve arkadaşları CICAndMal2017 veri kümesinin bir bölümünü kullanarak, derin ve makine öğrenme modelleriyle Android kötü amaçlı yazılım tespit sistemlerini test etmek ve doğrulamak için güvenilir bir veri kümesine olan ihtiyacı ele almaktadır. Veri kümesi Kanada Siber Güvenlik Enstitüsü web sitesinde mevcuttur. Kullandıkları veri kümesi, 347 zararsız örnek ve 365 kötü amaçlı yazılım örneği içermektedir. Kötü amaçlı yazılım örnekleri, reklam yazılımı, fidye yazılımı, korkutma yazılımı ve sms aldatma yazılımı olarak adlandırılan gruplara ayrılmıştır. Her bir grup, adından da anlaşılacağı gibi belirli saldırıları gerçekleştirir. Reklam ve SMS yazılımı uygulama çalışırken rahatsız edici reklamlar ve SMS mesajları gönderirken, Korkutma ve Fidye yazılımları ise kullanıcılardan ücret veya fidye talep ederek sistem çökmesini veya özel verilerin ifşa olmasını engellemeye çalışır. Bu veri kümesini kullanarak, çalışma derin öğrenme ve makine öğrenmesi tekniklerini kullanarak Android kötü amaçlı yazılım tespit sistemleri geliştirmeyi ve değerlendirmeyi amaçlamaktadır. Önerilen sınıflandırıcılar, CICAndMal2017 veri kümesinden alınan bir veri kümesi üzerinde eğitilmiştir. Veri kümesi, kötü amaçlı ve zararsız Android uygulamalarının statik analiziyle incelenmiştir. Analiz hem izinler hem de API çağrıları için yapılmış olup, her ikisinin de kullanılması Android kötü amaçlı yazılımların tespitinde daha güvenilir olduğunu göstermiştir. Derin öğrenme sınıflandırıcısı, izinler ve API çağrıları gibi Android kötü amaçlı yazılımların statik özelliklerini kullanarak %98,2 doğruluk seviyesine ve %98,0 f1 skoruna ulaşarak diğer tüm sınıflandırıcılardan daha iyi bir performans sergilemiştir [5].

Urooj ve arkadaşları Android kötü amaçlı yazılımların statik analiz yöntemi ile tespiti için yedi farklı ek özellik setinden oluşan yeni bir alt küme sunmuştur. Bu özellik setleri izinler, uygulama bileşenleri, yöntem etiketleri, intentler, paketler, API çağrıları ve hizmetler/alıcılar gibi kategorilerden yaklaşık 56000 özelliği kullanmaktadır. 500 binden fazla zararsız ve kötü amaçlı Android uygulaması üzerinde, diğer mevcut yöntemlere göre en yüksek kötü amaçlı örnek setine sahip olan bu özelliklerin kararlılığını değerlendirmişlerdir. Sonuçlar, yanlış pozitif oranı %0,3 olan %96,24 doğrulukla sonuçlarda ciddi bir iyileştirme sağlamıştır. Önerilen modelde ilk olarak Android uygulamalarının davranışını yakalamak ve analiz etmek için işlevleri

tanımlanmış ve seçilmiştir. Tersine mühendislik ve AndroGuard aracından yararlanarak özellikleri ikili vektörlere çıkarılmıştır. Ardından, Python yapısı ve bölme karıştırma işlevlerini kullanarak, modeli zararsız ve kötü amaçlı veri kümeleriyle eğitilmiştir. Ayrıca, sınıflandırmalar ve yüksek boyutlu verilerle uğraşırken, ensemble ve güçlü öğrenici algoritmalarının görece daha iyi performans gösterdiğini keşfetmişlerdir. Önerilen yaklaşım, statik analiz açısından sınırlıdır, sürdürülebilirlik sorunlarına dikkat etmez ve önemli bir çoklu doğrusallık engelini ele almaz. Ek özelliklerle birlikte, altı farklı sınıflandırıcı modeli veya makine öğrenme algoritması eğitilmiştir ve tahmin oranını artırmak için bir Boosting ensemble öğrenme yaklaşımı (AdaBoost) ile karar ağacına dayalı ikili sınıflandırma uygulanmıştır [6].

Keyvanpour ve arkadaşları yapmış oldukları çalışmada Drebin veri seti üzerinde deneyler gerçekleştirmiştir. Bu veri seti, 123.453 gerçek uygulama örneği ve 179 farklı kötü amaçlı yazılım türünden 5.560 örnek içermektedir. Bu uygulamalardan çıkarılan toplam özellik sayısı 11.381'dir ve bu özellikler, Androidmanifest.xml ve Classes.dex dosyalarından altı özellik grubundan elde edilmiştir. Bu veri setinin örnekleri Ağustos 2010 ile Ekim 2012 tarihleri arasında toplanmıştır. Bu veri seti, Google Uygulama Mağazası, farklı Çin ve Rus pazarları ve diğer kaynaklardan uygulamalar içermektedir. Uygulamalar VirusTotal hizmeti tarafından test edilmiştir. Özellik seçimi teknikleri için deneyler veri setinin tamamında gerçekleştirilmiş, eğitimi amacıyla 10 kat çapraz doğrulama kullanılmıştır. Veri seti 10 parçaya bölünmüştür. Model her adımda dokuz parça üzerinde eğitilirken, son parça test veri seti olarak kullanılmıştır. Bu çalışmada Keyvanpour ve arkadaşları özellik seçimi teknikleri ve makine öğrenimi kullanarak Android kötü amaçlı yazılımlarını tespit etmek için bir rasgele orman algoritması kullanmıştır. Makalede üç farklı özellik seçimi tekniği uygulanmıştır: etkili özellik seçimi, yüksek ağırlıklı özellik seçimi ve etkili grup özellik seçimi. Bu seçilen özellikler daha sonra rasgele orman algoritması tarafından kötü amaçlı yazılımları tespit etmek için kullanılmıştır. Drebin veri setinde yapılan deneyler, özellik seçimi yöntemlerinin rasgele orman değerlendirme ölçütleri ve gereken süre açısından doğruluğunu artırdığını göstermiştir [7].

1.3 Hipotez

Makine öğrenmesi yöntemleri, Android işletim sisteminde zararlı yazılımların tespiti

için kullanılan izin tabanlı statik analiz metodunun performansını ve doğruluğunu büyük veri setleri üzerinde iyileştirebilir. Bu hipotez çerçevesinde çeşitli makine öğrenmesi sınıflandırıcılarının performansları özgün ve güncel veri seti üzerinde test edilerek sonuçlar kıyaslandı. Çıkan sonuçları iyileştirmek için son zamanlarda daha popüler olan gradyan artırma tekniklerinden olan LightGBM ve XGBoost kullanarak sonuçları iyileştirmeyi hedefledik. Elde ettiğimiz çıktılar hipotezin doğruluğunu göstermiştir.

1.4 Tez Organizasyonu

Tez metni altı bölümden oluşmaktadır:

- ✓ Bölüm 1'de, tez çalışmasının amacı, literatür taraması ve hipotezi açıklanmıştır.
- ✓ Bölüm 2'de, Android işletim sistemi ve dosya yapısı açıklanmıştır. Ayrıca bu bölümde zararlı yazılımlar ve çeşitleri ile tersine mühendislik metodu kullanarak zararlı yazılım analiz yöntemleri açıklanmıştır.
- ✓ Bölüm 3'te çalışmada kullanılan özgün ve güncel veri seti detaylı olarak açıklanmıştır.
- ✓ Bölüm 4'de kullanılan makine öğrenmesi modelleri ve başarım metrikleri açıklanmıştır.
- ✓ Bölüm 5 de, bulgular ve deneysel sonuçlar açıklanmıştır.
- ✓ Bölüm 6'da tezin sonuçları hakkında yorumlar yapılarak ileride planlanan çalışmalardan bahsedilmiştir.

2. YÖNTEMLER

2.1 Android İşletim Sistemi

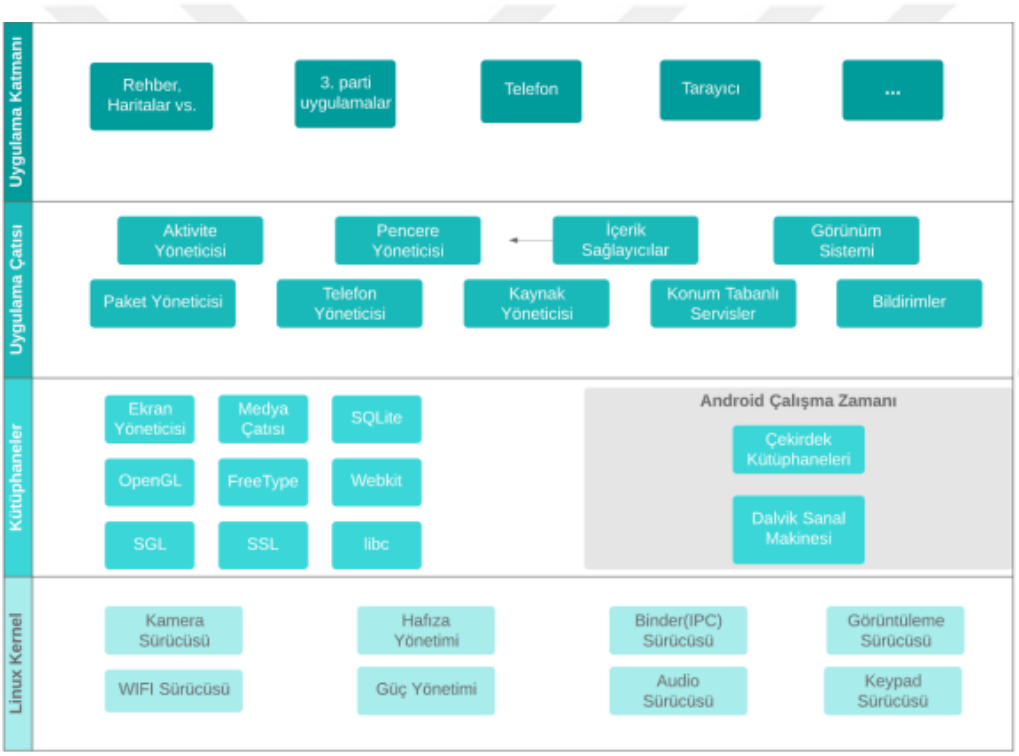
Günümüzde birçok mobil cihazda ya da tablet cihazlarda yaygın olarak kullanılan Android işletim sistemi açık kaynaklı bir işletim sistemidir ve Linux çekirdeği üzerine inşa edilmiştir. OHA (Open Handset Alliance), Linux çekirdeğinin modifiye edilmiş bir sürümü ve diğer açık kaynaklı uygulamalar üzerinde ilk Android'i oluşturmuştur. 2005 yılında, Google bu girişimi desteklemiş ve şirketi tamamen satın almıştır [8]. Eylül 2008'de, ilk Android akıllı telefon piyasaya sürülmüş ve kullanıcı dostu arayüzü, büyük bir topluluk desteği, özelleştirilebilirliği ve büyük firmalar tarafından üretilmesi sayesinde mobil endüstrinin lideri olmuştur. Sonuç olarak, pazarda Android destekli cihazlar yaratmak için yetenekli geliştiricilere olan ihtiyaç değerlendirilmiştir. Böylece, Android işletim sistemi, giyilebilir cihazlardan mobil cihazlara, dizüstü bilgisayarlardan akıllı televizyonlara, tabletlerden set üstü kutulara kadar çeşitli cihazlar için kapsamlı bir işletim sistemleri paketine dönüşmüştür [9].

2.2 Android İşletim Sistemi Güvenlik Modeli

Android mimarisi, güvenli bir işletim sistemi sağlamak için bir izolasyon yaklaşımını benimser. Android Mimarisi, Android işletim sistemi yüklü herhangi bir cihazda çalışabilen mobil uygulamalar oluşturmak için geliştiricilere birleşik bir yöntem sunar. Program bileşenlerinin yeniden kullanılmasını sağlayarak yeniden inşa ihtiyacını ortadan kaldırır [10]. Birden çok web sitesi, açık kaynak lisansı başlığı altında Android kaynak kodlarını sağlar. Linux çekirdeği, Android platformunun temelini oluşturur. Android çalışma zamanı (AÇZ) çoklu görev ve düşük seviye bellek yönetimi gibi temel özellikler için Linux çekirdeğine dayanır [11].

Uygulama Sandbox (Kum Havuzu), Android programlarının kendi "alanlarında" çalışmasına ve gerekli izinler olmadan diğer çalışan uygulamalar veya Android işletim sistemiyle etkileşime geçmesini önleyen bir güvenlik tekniğidir [12]. Android'in başlangıçtaki DEK (Disk Erişim Kontrolü) uygulama kum havuzu, her programa benzersiz bir UNIX kullanıcı kimliği (UID) ve uygulamanın kontrol ettiği bir dizin atayarak uygulamaları birbirinden ve sistemden izole etmiştir. Bu yöntem, gerçek kullanıcının UID' sini kullanarak uygulamaları yürütme konusundaki geleneksel masaüstü yönteminden önemli ölçüde farklılık göstermiştir [13]. Uygulama başına

UID (Kullanıcı Kimliği) ataması, bir uygulamanın kaynakları diğer uygulamalardan ve işletim sisteminden izole bir şekilde kullanabilmesini sağlar. Bu açıklamalar, Android işletim sisteminin güvenlik mimarisinin temel yönlerini ve uygulama kum havuzunun önemini vurgulamaktadır. Uygulama başına kullanıcı kimliği UID, izin kontrollerini kolaylaştırır ve zaman alıcı olan her işlem kimliği (PID) doğrulamasını ortadan kaldırır. Bir uygulamaya verilen izinler merkezi olarak kaydedilir (/data/system/packages.xml), böylece diğer hizmetler bunları sorgulayabilir. Örneğin, bir uygulama konum servisinden adres istediğinde, konum servisi, talep edilen UID' nin konum izni verilip verilmediğini kontrol etmek için yetkilendirme servisini kontrol eder.



Şekil 2.1 Android işletim sistemi bileşenleri

Android işletim sistemi, aşağıdaki beş bölüme ve dört ana katmana gevşek bir şekilde ayrılmış yazılım bileşenlerinin bir yığındır:

- Linux çekirdeği: Bu, cihazın donanımı arasında bir soyutlama düzeyi sağlar ve kamera, klavye, ekran vb. gibi tüm gerekli donanım sürücülerini içerir. Ayrıca, çekirdek Linux'un üstün olduğu bağlantı ve çevre birimleriyle basit bağlantı sağlayan çok sayıda cihaz sürücüsünü yönetir.

- Kütüphaneler: WebKit, libc, SQLite, ses ve video oynatma ve kaydetme kütüphaneleri, İnternet güvenliği için SSL kütüphaneleri vb. gibi, Linux çekirdeğinin bir parçası olan kütüphaneler Linux çekirdeğinin üstünde bulunur. Kütüphane bileşenleri Tablo 2.1'de listelenmiştir.

Tablo 2.1 Kütüphane içindeki çekirdek kütüphaneler

Kütüphane	**Açıklama**
SQLite	Bu kütüphane, içerik sağlayıcılar tarafından yayınlanan verilere erişmek için kullanılır ve SQLite veri tabanı yönetimi sınıflarını içerir.
SSL	İnternet güvenliğini sağlamak için kullanılır.
OpenGL	OpenGL/ES 3D grafik renderlama API'sine Java arayüzü sağlamak için kullanılır.
Medya Çerçevesi	Farklı medya formatlarının kaydedilmesine ve oynatılmasına izin veren farklı medya kodeklerini sağlar.
WebKit	İnternet içeriğini veya HTML içeriğini görüntülemek için kullanılan tarayıcı motorudur.
Web Tarayıcısı	Açık kaynak WebKit düzen motoruna dayanan ve HTML5 ve CSS3'ü destekleyen Chrome'un V8 JavaScript motoruyla birleştirilmiştir.

- Uygulamalar: Bu kategori, Android geliştirme sürecinde kullanılan Android'e özgü Java kütüphanelerini içerir. Uygulama çerçevesi kütüphanelerine ek olarak, bu kategori, kullanıcı arayüzü oluşturma, grafik çizimi ve veri tabanı erişimi gibi işlevleri sağlayan kütüphaneleri içerir.
- Android Çalışma Zamanı (AÇZ): Bu bölüm, Android için özel olarak geliştirilmiş ve optimize edilmiş bir Java Platformu olan Dalvik Sanal Makinesini kapsar. Dalvik sanal makinesi, Java programlama dilinin temel özelliklerinden olan bellek yönetimi ve çoklu iş parçacığı kullanımını temel alır. Dalvik sanal makinesi, her Android uygulamasının kendi versiyonuyla oturumunu gerçekleştirmesini sağlar. Android çalışma zamanı ayrıca, Android

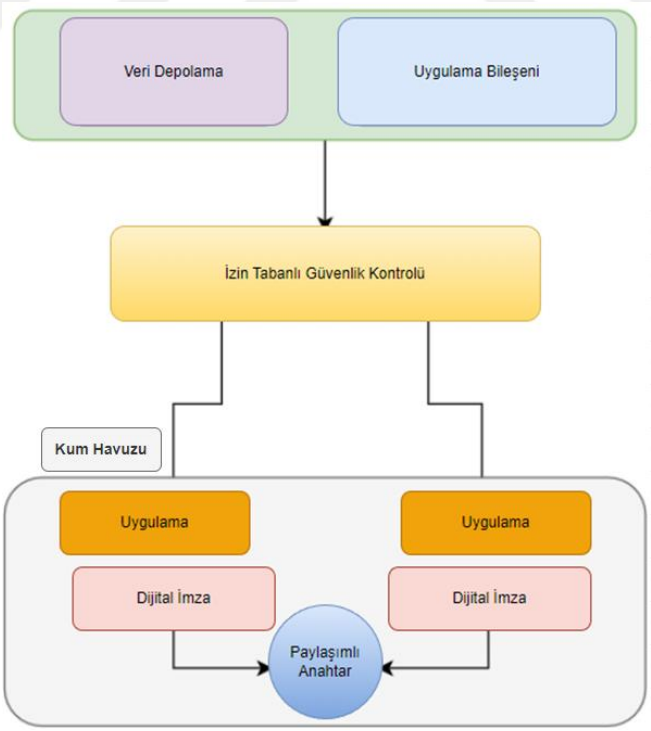
uygulama geliştiricilerinin Java dilinde Android uygulamaları oluşturmasına olanak tanıyan bir dizi sanal kütüphaneyi içerir [14].

- Uygulama Çerçevesi: Android uygulamaları doğrudan uygulama çerçevesiyle etkileşime geçer. Uygulama çerçevesi, kaynak yönetimi ve sesli arama yönetimi gibi temel Android cihaz özelliklerini destekler. Java sınıfları şeklinde, uygulama çerçevesi katmanı uygulamalara çeşitli yüksek seviyeli hizmetler sunar. Uygulama geliştiriciler, bu hizmetleri kendi uygulamalarının içinde kullanabilir.

Tablo 2.2 Uygulama çerçevesi bileşenleri

Uygulama çerçevesi bileşenleri	**Açıklama**
Aktivite Yöneticisi	Aktivitelerin oluşturulması, başlatılması, durdurulması ve sonlandırılması gibi işlemleri gerçekleştirir.
İçerik Sağlayıcı	İki uygulama arasında veri paylaşımını yönetmek için kullanılır. Bir uygulamanın diğer bir uygulamanın verilerine erişmesine ve bu verileri paylaşmasına olanak tanır.
Telefon yöneticisi	Tüm sesli aramaların yönetimini sağlar. Telefon çağrılarının yapılması, alınması ve yönlendirilmesi gibi telekomünikasyon işlevlerini yönetir.
Konum Yöneticisi	GPS veya baz istasyonu kullanılarak elde edilen konumların yönetimini sağlar. Konum hizmetlerini kullanarak uygulamaların konum bilgilerini almasını ve işlemesini sağlar.
Kaynak Yöneticisi	Android uygulamalarında kullanılan farklı türdeki kaynakları yönetmek için kullanılır. Bu kaynaklar, resimler, metinler, renkler, stil dosyaları, dize kaynakları ve diğer varlıkları içerir. Kaynaklara erişim, yönetim ve geri yükleme gibi işlemleri kolaylaştırır.

Güvenlik yapısı açısından Android, bir izolasyon yaklaşımı kullanır. Android işletim sistemi, Linux çekirdeği üzerine inşa edilmiştir ve bellek konumları, süreç yönetimi, güç yönetimi ve sürücüler aracılığıyla fiziksel cihazlara ve ağ yönetimine kadar önemli sistem işlevlerini gerçekleştirir [15]. Bu, temel bir bileşen olan dalvik sanal makinesi aracılığıyla gerçekleştirilir. Ayrıca, Android cihazlarının verimli bir şekilde birçok sanal makine çalıştırmasına izin vermek amaçlanır. Sonuç olarak, Android uygulaması, işlemini Dalvik sanal makinesi içinde çalıştırabilir ve kendi sanal makinesini kullanabilir. Şekil 2.2'de görüldüğü gibi, her Android uygulaması için bir kum havuzu ortamı açıkça belirgindir (Dalvik Sanal Makinesi). Bu nedenle, çalışan bir Android programı, diğer çalışan Android uygulamalarını etkilemez veya değiştirmez çünkü her iki uygulama da ayrı kum havuzlarında çalışır [16]. Linux tabanlı erişim kontrol önlemleri, her Android işletim uygulamasının diğerlerinden izole edildiğini sağlar. Android uygulamaları, güvenli bir ortamda Android Güvenlik paradigmaları kullanılarak yürütülebilir. Bu anlamda izolasyon yapısı Android uygulamalarının işlevselliğini sınırlasa da her sanal makine birbirinden ayrı olduğu için bu bir gerekliliktir.



Şekil 2.2 Android yığını kum havuzu yapısı

Linux çekirdeği, Android'in temel güvenlik mekanizmalarını kullanmasını sağlar ve cihaz üreticilerinin tanınmış bir çekirdek için cihaz sürücüleri yazmasına olanak tanır. Donanım soyutlama katmanı (DSK), cihaz donanım özelliklerine erişim sağlayan standart arayüzlere sahip Java API çerçevelerini sunar. DSK, kamera veya iletişim modülü gibi belirli bir donanım cihazı için bir arayüz sağlayan birkaç kütüphane modülünü içerir. Bir çerçeve API'si cihaz bileşenlerine erişim istediğinde, Android işletim sistemi ilgili kütüphane modülünü içe aktarır [17].

Android'in sistem bileşenleri ve AÇZ ve DSK gibi hizmetleri, doğru çalışabilmesi için C ve C++ dillerinde yazılmış kütüphanelere ihtiyaç duyan yerel kod kullanılarak oluşturulur. Bu yerel kütüphanelere, Android platformunda Java çerçeve API'leri kullanılarak erişilebilir. Android'in Java OpenGL API'si aracılığıyla oluşturulacak programlara 2D ve 3D grafik yetenekleri eklemek mümkündür [18]. Android ile birlikte e-posta, SMS, takvimler, internet gezintisi, kişiler ve daha fazlası için temel uygulamalar sunulmaktadır. Platform destekli uygulamalar, kullanıcının cihazına yüklenmiş diğer uygulamalar gibi işler. Örneğin, üçüncü taraf bir yazılım, kullanıcının varsayılan tarayıcısı, SMS mesajlaşması (belirli koşullarla) veya hatta varsayılan klavyesi olabilir.

Kullanıcılar ve programcılar birçok yerleşik uygulamadan faydalanabilir. Kendi uygulamanızda işlevselliği uygulamak zorunda kalmadan bir SMS mesajı gönderebilirsiniz ya da bunun yerine mesajı alıcıya göndermek için mevcut bir SMS uygulamasını kullanabilirsiniz [19].

Android'in güvenlik mimarisi kapsamında, hiçbir uygulama diğer uygulamalara, işletim sistemine veya kullanıcıya zarar verebilecek herhangi bir eylem gerçekleştirme yetkisi verilmez. Bu kategoriye kullanıcının verilerini (kişiler veya e-postalar gibi) okuma veya yazma, ağa erişim, cihazın uyanık kalması vb. gibi çeşitli eylemler girer.

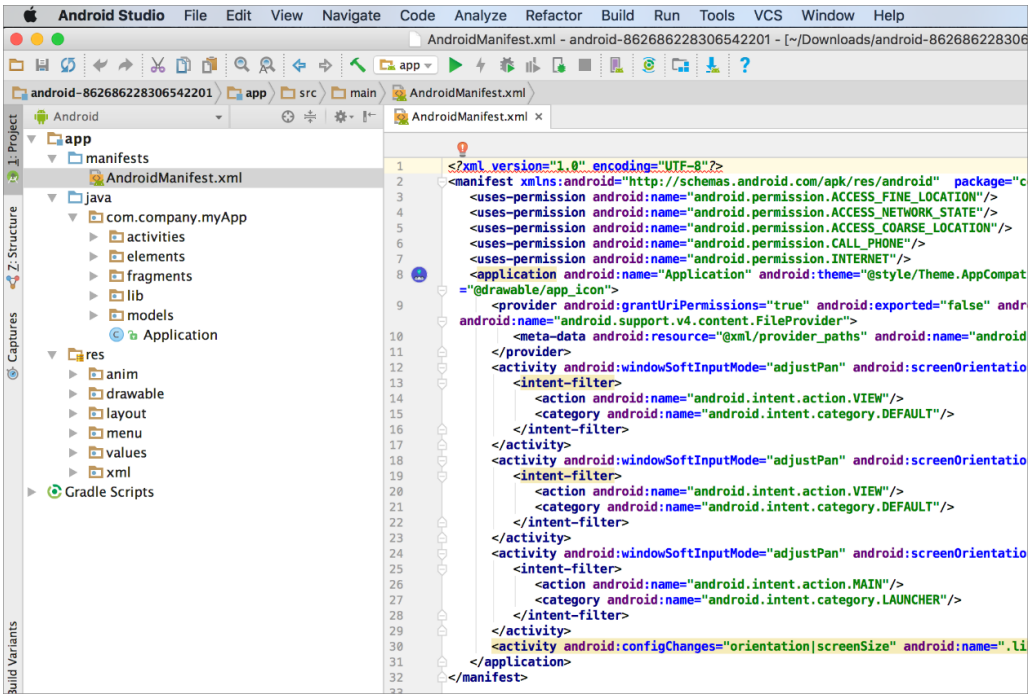
Bir uygulamanın işlemi için kum havuzu içinde korunur. Temel kum havuzu tarafından sağlanmayan ek yetenekler için gereken izinleri açıkça belirterek diğer programları etkileyebilir [20]. İşletim sistemi, bu hakları sertifikalara dayanarak onaylar ya da reddeder veya kullanıcıdan izin isteyerek yönetebilir. Uygulamalar izinlerini statik olarak tanımlarlar, bu nedenle yüklemelerden sonra değiştirilmezler [21].

2.3 Android Güvenlik Mekanizması

Android güvenliği, kullanıcının verilerini ve telefonun donanımını, sistemini ve yazılımını korumayı hedefler. Android'in güvenliğinin temel unsurları aşağıdaki gibidir: Linux çekirdeğine güvenme, kum havuzu uygulama imzalama, uygulama tarafından tanımlanan ve kullanıcı tarafından verilen izinler. Linux çekirdeği, genellikle yüksek hassasiyetli güvenlik ortamlarında kullanılması için güvenilir kabul edilir. Linux güvenliği, açık bir ortam olması nedeniyle sürekli olarak güvenlik uzmanları ve güvenlik hatalarını düzeltip yamalayan geliştiriciler tarafından geliştirilmektedir ve saldırganlar tarafından istismar edilebilecek zayıflıklar bulunabilir. Aynı zamanda gereksiz ve güvensiz bölümlerin çekirdekten çıkarılabilmesine olanak sağlar. Uygulamaların kum havuzu özelliği, uygulamaların işlemlerini ve verilerini benzersiz bir kullanıcı kimliği (UID) altında diğerlerinden izole eder. Android, her uygulamaya kurulum sırasında ayrı bir Linux kullanıcı kimliği verir. Başka bir deyişle, her uygulamanın tüm ömrü boyunca sabit bir benzersiz UID'si vardır. Aynı uygulama başka cihazlarda farklı bir UID'ye sahip olabilir, önemli olan iki farklı uygulamanın aynı UID'ye atanamamasıdır. Her uygulamanın kendi UID'si olduğu için, aynı işlemde çalışamazlar, bunun yerine UID'si altında ayrı ayrı çalışmaları gerekir. Bu, çalışan işlemi birbirinden güvenli uygulamalara karşı izole eder. Ayrıca, herhangi bir uygulamanın verileri kendi UID'si altında depolanır ve diğer uygulamalar tarafından erişilemez hale gelir. Uygulama imzalama özelliği, herhangi bir uygulamanın .apk dosyasının geliştirici sertifikasıyla imzalanması gerektiğini gerektirir ve bu şekilde uygulamanın yazarını tanır. Bu özellik, aynı sertifika ile imzalanmışsa uygulamaların aynı UID'yi paylaşmasına olanak sağlar. Ayrıca, sistem imza düzeyinde izinleri kabul veya reddeder; sistem, istenen uygulamaya aynı izinleri bildiren diğer uygulamaların sertifikasıyla imzalanmışsa, imza düzeyinde izinleri sağlar. Son olarak, Android tarafından benimsenen izin modeli, telefonun kaynaklarını ve işlevlerini yalnızca ilgili yetkilere sahip uygulamalara erişilebilir kılarak korur. Varsayılan olarak, hiçbir uygulamanın telefonun donanımı, yazılımı, işlevleri ve verileriyle ilgili izinleri yoktur. Uygulama geliştiricilerinin uygulama işlevselliği için gereken izinleri bildirmesi gerekmektedir. Kurulum sırasında kullanıcılar istenen izinleri vermelidir, aksi takdirde sistem tarafından kurulum sonlandırılır.

2.4 Android İzin Modeli

İzin modeli, Android güvenliğinin temel güvenlik kavramıdır. Android, uygulamaları ayrı ayrı kum havuzlarında çalıştırır. Her uygulama, sistem kaynaklarına erişimi olmayan izole bir ortamda çalıştırılır. Uygulamanın işlevselliği için gereken sistem kaynaklarına erişebilmesi için izinlerin uygulamaya verilmesi gerekmektedir. İzinler, geliştiricinin geliştirme aşamasında AndroidManifest.xml dosyasında bildirmesi gereken izinlerdir. Bu izinler, kurulum sırasında sistem veya kullanıcı tarafından verilmelidir; bir kez verildikten sonra uygulama kaldırılmadığı sürece değiştirilemezler. İzinler, AndroidManifest.xml dosyasındaki bir veya daha fazla <permission> etiketiyle bildirilebilir; ayrıca gerekli ve isteğe bağlı özniteliklerle tanımlanmalıdır. <label> ve <description> öznitelikleri, kurulum sırasında kullanıcıya görüntülenen metinlerdir. Bu öznitelikler, iznin gösterdiği ayrıcalıkları anlamak için kullanıcıya yardımcı olacak şekilde net bir şekilde tanımlanmalıdır. <permissionGroup> özniteliği isteğe bağlıdır ve iznin kategorisini kullanıcıya göstermek için sistem tarafından kullanılır. <protectionLevel> özniteliği, iznin güvenlik düzeyini tanımlamak için gereklidir; uygulama ayrıcalıklarının önemini belirler.



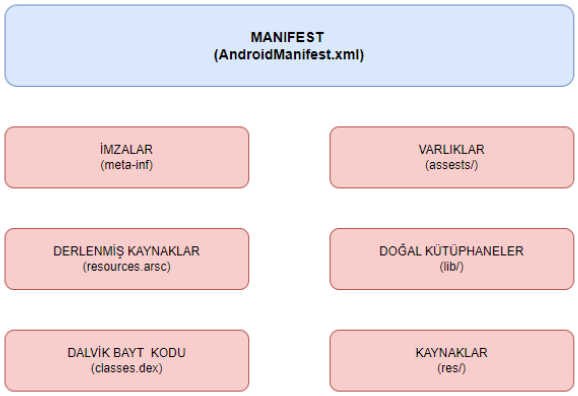
Şekil 2.3 AndroidManifest.xml dosyası örneği

Şekil 2.3 de örnek bir manifest dosyasındaki izin ve özniteliklerini gösterilmektedir. İzin modeli, sistem kaynaklarına ve diğer uygulamaların verilerine erişimi kontrol etmek için <uses-permission> ve <permission> etiketlerini kullanır. <uses-permission> etiketi, uygulamanın belirli verilere, donanımlara, yazılımlara ve diğer sistem kaynaklarına erişmek için gereken izinleri tanımlar. Öte yandan, <permission> etiketi, diğer uygulamaların uygulamanın verilerine ve bileşenlerine erişmesi gereken izinleri tanımlar. Başka bir deyişle, uygulamanın bileşenlerinin diğer uygulamalar tarafından nasıl erişilebilir olacağını belirler.

2.5 Android Dosya Sistemi

Android uygulamalarının çoğunu oluşturan sıkıştırılmış dosyalara "Android Paket Kitleri" (APK'lar) denir. APK, bir Android cihaza dağıtılmak için uygulama kodunu, geliştirici sertifikalarını ve manifest dosyasını içermelidir [22]. İzin istekleri ve uygulamanın bileşenleri, Manifest adı verilen bir meta veri dosyasında listelenir. Manifest dosyası, Android uygulamalarının belirli kaynaklara erişmek için izin talep edebileceği tek yerdir. Kullanıcı, bu ayrıcalıkları uygulamanın yükleme veya çalışma aşamalarında vermelidir. Bir uygulamanın manifesti, belirli bir izin setiyle aşağıdaki bileşenler hakkında bilgi içerir:

- Aktiviteler: kullanıcı arayüzünü kontrol eder ve uygulama ekranlarıyla ilişkilidir.
- İçerik sağlayıcıları: Uygulamalar arasında veri transferini sağlar.
- Intent: Niyet adı verilen mesajlaşma nesneleri aracılığıyla bir bileşen diğerine doğru eylemi başlatma veya veri paylaşımını sağlamayı amaçlar.
- Yayın alıcıları: Uygulamalar ve işletim sistemi, yayın mesajları göndermek ve almak için yayın alıcılarını kullanır.
- Hizmetler: Arka planda çalışan bileşenlere denir.



Şekil 2.4 Android kurulum dosyası bileşenleri

- Manifest dosyası: Uygulamanın adını, sürümünü, erişim izinlerini ve referans alınan kütüphane dosyalarını içeren dosyadır. Bu dosya Android ikili XML formatında olabilir ve insan tarafından okunabilir metin tabanlı XML' ye dönüştürülebilir.
- İmzalar (META-INF): Bu bölüm, sertifikaları ve imzaları içerir ve APK' nın bütünlüğünü ve doğruluğunu doğrulamak için kullanılır.
- Varlıklar (Assets): Asset Yöneticisi tarafından erişilebilen uygulama varlıklarını içeren bir dizin.
- Derlenmiş kaynaklar (resources.arsc): Önceden derlenmiş kaynakları içeren bir dosya, örneğin ikili XML.
- Doğal Kütüphaneler (lib): Platforma bağımlı derlenmiş kodu içeren bir dizin; bu dizin ilgili hedef mimarilerine göre ek dizinlere ayrılabilir.
- Dalvik Byte Kodu: Android uygulamaları Java dilinde geliştirilir ve daha sonra .class dosyaları olarak derlenir (Dalvik byte kodu) .class dosyaları tek bir Dalvik yürütülebilir dosyaya, yani classes.dex dosyasına sıkıştırılır ve sonunda dalvik sanal makinesinde yürütülür.
- Kaynaklar: Uygulamanın gereksinim duyduğu tüm gerekli kaynakları içerir. Bu kaynaklar arasında resimler, animasyonlar, düzenler ve kullanıcı arayüzü öğeleri bulunur. Derleme zamanında, tüm bu kaynaklar uygulamaya dahil edilir.

Bu bileşenler, Android uygulama paketinin yapısına ve işlevselliğine katkıda bulunur.

2.6 Zararlı Yazılımlar

Zararlı yazılımlar bilgisayar sistemlerini, ağları veya kullanıcıları zarar vermek veya sömürmek amacıyla tasarlanmış herhangi bir yazılımı ifade eder. Kötü niyetle oluşturulan bu yazılımlar, izinsiz erişim elde etmek, hassas bilgileri çalmak, işlemleri bozmak veya bilgisayarları ve ağları zarar vermek için kullanılır. Zararlı yazılımlar çeşitli şekillerde ortaya çıkabilir, her biri belirli bir amaç için kullanılır ve farklı davranışlar sergiler. Yaygın olarak bilinen zararlı yazılım türleri şunlardır:

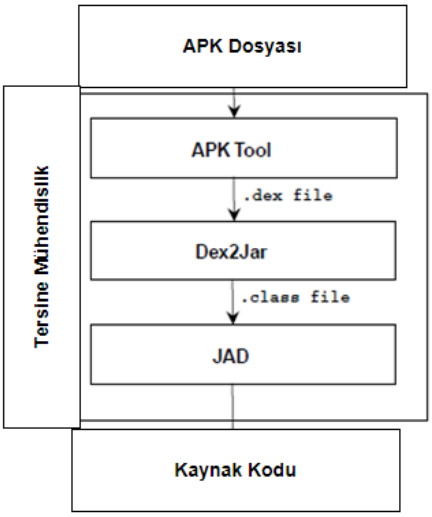
- Virüsler: Kendini çoğaltabilen programlardır ve genellikle meşru dosyalara veya programlara bulaşırlar. Kendilerini diğer dosyalara veya sistemlere bulaştırarak yayılırlar ve veri değiştirme, silme, dosya bozulması veya bilgisayarın performansını yavaşlatma gibi zararlar oluşturabilirler.
- Solucanlar: Başka dosyalara veya programlara bağlı olmadan bağımsız olarak çoğalan programlardır. Sistemleri enfekte etmek için genellikle güvenlik açıklarından faydalanırlar ve ağ bağlantılarını kullanarak yayılırlar. Solucanlar, sistem kaynaklarını tüketebilir, ağ trafiğini tıkayabilir ve veri hırsızlığı veya kötü amaçlı aktiviteler gibi zararlı eylemleri gerçekleştirebilirler.
- Truva Atları: Kötü niyetli yazılımların meşru yazılımlar veya dosyalar gibi görünerek kullanıcıları kandırması için tasarlanmıştır. Aktive edildiklerinde hassas bilgileri çalabilir, sistem ayarlarını değiştirebilir veya uzaktan saldırganlara yetkisiz erişim sağlayabilirler.
- Fidye yazılımı: Kurbanın bilgisayarında veya ağında bulunan dosyaları şifreleyerek erişimi engeller ve fidye ödenene kadar dosyaları kullanılamaz hale getirir. Genellikle kötü niyetli e-posta ekleri, tehlikeli web siteleri veya yazılım güvenlik açıklarından yayılır. Fidye yazılımı saldırıları ciddi sonuçlar doğurabilir, çünkü kurbanlar önemli verilere erişimlerini kaybedebilir veya maddi kayıplarla karşılaşabilirler.
- Casus yazılımlar: Kullanıcının veya kuruluşun haberi veya izni olmadan bilgi toplamak amacıyla tasarlanmıştır. Klavye girişlerini takip edebilir, ekran görüntüleri yakalayabilir, web tarama alışkanlıklarını izleyebilir ve şifreler

veya kredi kartı bilgileri gibi hassas verileri toplayabilir. Toplanan bilgiler genellikle kötü niyetli kişilere gönderilir ve zararlı amaçlarla kullanılır.

- Reklam yazılımları: Kullanıcılara istenmeyen reklamlar sunmayı amaçlayan yazılımlardır. Genellikle meşru yazılımlarla birlikte gelir ve kullanıcılara zorlayıcı reklamlar gösterir veya web tarayıcılarını belirli web sitelerine yönlendirir. Reklam yazılımları diğer zararlı yazılım türleri kadar yıkıcı olmayabilir, ancak sistem performansını ciddi şekilde düşürebilir ve kullanıcı gizliliğini tehlikeye atabilir.
- Tuş kaydediciler: Saldırıya uğramış bir sistemdeki tuş vuruşlarını kaydederek şifreler, kredi kartı bilgileri veya kişisel mesajlar gibi hassas bilgileri yakalar. Donanım veya yazılım tabanlı olabilirler ve toplanan veri genellikle kötü niyetli kişilerin eline geçer ve bunu kendi çıkarları için kullanabilirler.
- Bot ağları: "Bot" veya "zombi" adı verilen enfekte bilgisayarların merkezi bir komuta kontrol sunucusu tarafından kontrol edildiği ağlardır. Bot ağları genellikle Dağıtık Hizmet Engelleme (DDoS) saldırıları, istenmeyen e-posta kampanyaları veya diğer türdeki kötü amaçlı yazılımların dağıtımı için kullanılır.

2.7 Tersine Mühendislik

Android uygulama paketi (APK), classes.dex dosyasını, Android-Manifest.xml dosyasını, kaynakları ve kütüphaneleri sıkıştırılmış şekilde içeren bir zip arşividir. Java sınıfları tek bir Dalvik yürütülebilir (.dex) dosyasına sıkıştırıldığından, zip arşivi içeriği insan tarafından okunamaz. Bununla birlikte, birkaç araç kullanarak bir APK dosyasını tersine mühendislik yaparak Java kaynak kodunu ve ilgili dosyaları çıkarmak mümkündür. Şekilde, bir APK'nın tersine mühendislik için gereken adımları açıklar. APK aracı, APK dosyasını açmak için kullanılır ve classes.dex dosyasını çıkarmak için kullanılır. Daha sonra classes.dex dosyası, dex2jar aracı kullanılarak java-archive .jar dosyası şeklinde decompile edilir. Dex2jar aracı tarafından üretilen .jar dosyası, hala insan tarafından okunamayan .class dosyaları şeklinde java byte kodunu içerir. Son olarak, JAD gibi bir java decompiler aracı kullanılarak .class dosyaları java kaynak kodu şeklinde decompile edilir.



Şekil 2.5 APK dosyalarının tersine kaynak kodunun elde edilmesi

Bir Android uygulamasının tersine mühendisliği genellikle bir Android uygulamasının statik analizi yapılırken gerçekleştirilir. Uygulama tersine mühendislik yapıldıktan sonra, kötü amaçlı yazılım analisti uygulamanın kaynak koduna ve manifest.xml dosyasına erişebilir ve çeşitli uygulama özelliklerini alabilir. Bir Android uygulamasının kaynak kodu genellikle izinler, ağ adresleri, API çağrıları ve işlev çağrı grafikleri gibi verileri ve özellikleri elde etmek için ayrıştırılır. Bununla birlikte, bir Android uygulamasının manifest.xml dosyasından izinler, filtrelenmiş intentler, paket adları, hizmetler, alıcılar ve donanım bileşenleri gibi statik özellikler çıkarılabilir.

APK dosyalarını decompile etmek için kullanılan bazı popüler araçlar şunlardır:

- JADX: JADX, Java kaynak kodunu dönüştürmek için popüler bir araçtır. APK dosyasını açar ve içindeki Java kodunu anlaşılabilir bir şekilde çıkarır. Android uygulamalarının kaynak kodlarına ulaşmak için yaygın olarak kullanılır.
- APKTool: APKTool, APK dosyalarını açmak ve içindeki kaynak dosyaları, resimler ve XML dosyalarına erişmek için kullanılan popüler bir araçtır. Uygulamanın iç yapısını anlamak ve kaynakları incelemek için kullanışlıdır.
- Jadx-gui: jadx-gui, JADX'in grafik arayüzüne sahip bir sürümüdür. Kullanıcı dostu bir arayüze sahip olması sayesinde APK dosyalarını daha rahat bir şekilde decompile etmek için tercih edilebilir.
- dex2jar: dex2jar, Android uygulamalarının dex dosyalarını jar dosyalarına

dönüştürür. Bu sayede APK dosyalarındaki bytecode' ları daha kolayca incelemek mümkün olur.

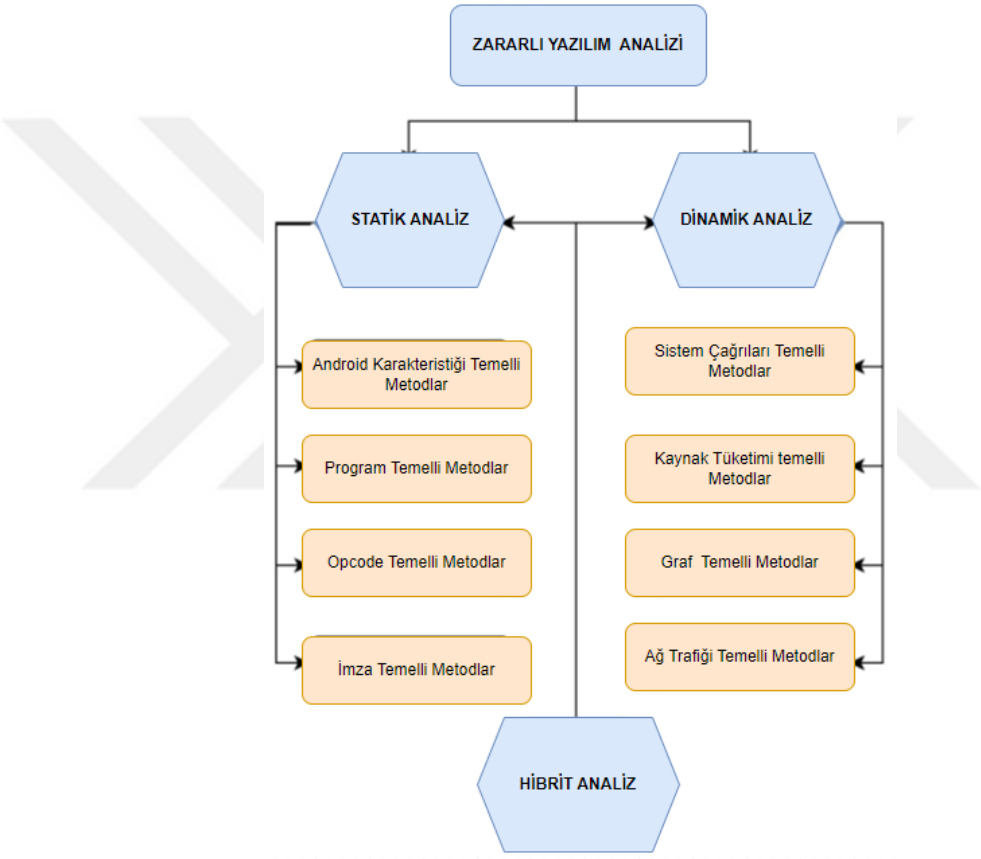
- JADX-GUI: Bu araç, jadx-gui' nin geliştirilmiş bir sürümüdür ve Java kaynak kodlarını, XML dosyalarını ve diğer uygulama kaynaklarını daha anlaşılır bir şekilde sunar. Arayüzü kullanıcı dostu ve basittir.
- Androguard: Android uygulamalarını analiz etmek ve incelemek için kullanılan bir açık kaynaklı araçtır. Android APK dosyalarını geri mühendislik yapmak, kötü amaçlı yazılımları tespit etmek, güvenlik açıklarını bulmak ve uygulama davranışını anlamak için kullanılabilir. Androguard, Python programlama diliyle yazılmıştır ve bir dizi özellik sunar. Androguard aracının başlıca özellikleri şunlardır:
  - ✓ APK Analizi: Androguard, APK dosyalarını açabilir, içerisindeki kaynak kodu, kaynak dosyaları, manifest dosyası, XML dosyaları, resimler ve diğer bileşenleri inceleyebilir.
  - ✓ Kod İnceleme: Androguard, APK dosyasının içindeki Java sınıflarını decompile edebilir ve elde edilen kodu analiz etmek için kullanılabilir. Bu, uygulama mantığını anlamak, kod tabanını araştırmak ve potansiyel güvenlik açıklarını tespit etmek için kullanışlı olabilir.
  - ✓ İz izleme: Androguard, uygulamanın davranışını izlemek ve potansiyel kötü amaçlı faaliyetleri tespit etmek için kullanılabilir. Bu, ağ trafiğini izlemek, API çağrılarını analiz etmek, dosya sistemi etkileşimlerini takip etmek ve diğer dinamik faaliyetleri incelemek anlamına gelir.
  - ✓ Sertifikaları İnceleme: Androguard, APK dosyasının içindeki sertifikaları analiz edebilir ve bu sayede uygulamanın kimlik doğrulaması ve güvenilirliği hakkında bilgi sağlayabilir.
  - ✓ Statik Analiz: Androguard, APK dosyasının içindeki izinleri, bileşenleri, servisleri, alıcıları ve diğer özellikleri analiz ederek potansiyel güvenlik tehditlerini tespit etmek için kullanılabilir.

Androguard, Android uygulama güvenliği araştırmacıları, kötü amaçlı yazılım analistleri ve uygulama geliştiricileri tarafından yaygın olarak kullanılan güçlü bir

araçtır. Açık kaynaklı olması sayesinde kullanıcılar, aracı kendi ihtiyaçlarına göre özelleştirebilir ve geliştirebilir. Bu çalışmada APK dosyalarından tersine mühendislik yöntemi ile uygulama izinlerini elde etmek için python programlama dili içinde bulunan androguard kütüphanesi kullanılmıştır.

2.8 Zararlı Yazılım Analiz Teknikleri

Zararlı yazılım analiz teknikleri temel olarak üç ana grupta sınıflandırılmıştır. Bunlar Statik analiz, dinamik analiz ve hibrit analiz yöntemleridir.



Şekil 2.6 Zararlı yazılım analiz yöntemleri

2.8.1 Statik analiz

Statik analiz, uygulamayı çalıştırmadan kötü amaçlı dosyaların analiz edilmesi tekniğidir. Statik analiz, bir uygulamanın kaynak kodunu veya ikili kodunu analiz ederek potansiyel olarak kötü amaçlı faaliyetleri tespit etmeyi içerir [23]. Statik analiz metodu inceleyen çözümleyiciye, bilinen kötü amaçlı yazılımlara benzer desenleri bulmak için kötü amaçlı bir dosyanın statik özelliklerini, dize, karma değerleri,

imzaları ve meta verilerini değerlendirmek için imkân sağlar. Statik analiz, bir uygulamanın çalıştırılmasını gerektirmediği için hesaplama maliyeti ve zaman açısından ucuzdur. Statik analiz, makine öğrenmesi tabanlı Android kötü amaçlı yazılım tespiti alanında yaygın olarak kullanılmaktadır. Kötü amaçlı ve zararsız Android uygulamalarından çeşitli ayırt edici özellikler çıkarılır ve bunlar makine öğrenmesi tabanlı algoritmaların eğitimi ve testi için kullanılır [24]. Tablo 2.3 'de, Android kötü amaçlı yazılım tespiti için statik analiz yoluyla elde edilebilecek çeşitli özelliklerin bir listesini sunmaktadır.

Tablo 2.3 Statik analiz yoluyla elde edilebilecek özellikler

Özellik	**Kaynak**	**Açıklama**
İzinler	AndroidManifest.xml	Bir uygulamanın çalışması için ihtiyaç duyduğu özel izinler
API Çağrıları	Source/byte code	Statik analiz yoluyla çıkarılan uygulama tarafından çağrılan API çağrıları
URL'ler	Source code	Bir uygulamanın kaynak koduna string olarak yerleştirilmiş URL'ler
Intent	AndroidManifest.xml	Intent nesneleri belirli eylemlerin gerçekleştirilmesini tanımlar.
OpCode	Byte code	Yapılacak işlemi tanımlayan talimatları içerir
Strings	Source Code	Fonksiyonların, sınıfların, nesnelerin ve değişkenlerin isimleri gibi stringleri içerir
Paket Adı	AndroidManifest.xml	Android uygulamasını tanımlamak için kullanılan paket ismi
Uygulama Bilişenleri	AndroidManifest.xml	Her bir uygulama bileşeni için etkinliklerin, hizmetlerin, alıcıların ve sağlayıcıların listesi.

2.8.2 Dinamik analiz

Dinamik analiz, potansiyel kötü amaçlı faaliyetleri tespit etmek için bir uygulamanın çalıştırılmasını içerir. Kötü amaçlı bir uygulamanın çalıştırılması, ana cihaza zarar

verebileceği için genellikle izole edilmiş sanal bir ortamda gerçekleştirilir [25]. Dinamik analiz, çözümleyicinin bir uygulamanın davranışını güvenli bir modda kaynak tüketimi, işlemler, ağ trafiği gibi kaynaklar üzerinden incelemesine olanak tanır. Ayrıca, dinamik analiz, talimat dizileri, API çağrıları ve sistem çağrıları gibi Android uygulamasının önemli özelliklerini yakalayabilir [26]. Bu özellikler, makine öğrenmesi tabanlı sınıflandırıcıları eğitmek için özellik vektörlerine gömülür. Tablo 2.4 'de Android kötü amaçlı yazılım tespiti için önemli dinamik analiz özelliklerinin bir listesini sunmaktadır. Dinamik analiz, bir uygulamanın kontrol edilen bir ortamda çalıştırılmasını gerektirdiği için kaynak ve zaman tüketimi açısından statik analize göre daha pahalıdır.

Tablo 2.4 Dinamik analiz yoluyla elde edilebilecek özellikler

Özellik	**Açıklama**
Kullanılan izinler ve API çağrıları	Dinamik analiz yoluyla çıkarılan istenen izinler ve API çağrıları izleri.
Ağ trafiği	Bu özellikler, ağ trafiği günlüklerinden çıkarılan verileri, HTTP iletişimlerini ve ağ paketlerinden gelen verileri içerir.
Sistem çağrıları	Sistem çağrıları, çekirdeğin temel işlevselliğine erişmek için çağrılır. Sistem çağrısı izleri dinamik analiz yoluyla çıkarılabilir.
Batarya ömrü	Uygulamaların batarya tüketimi çalışma zamanında profil olarak kaydedilir ve kötü niyetli desenler bulmak için kullanılır.
CPU kullanımı	Bu özellik, bir uygulamanın çalışma zamanında CPU kullanımını ifade eder.
Bellek kullanımı	Bu özellik, belirli bir uygulamanın bellek kullanımını kötü niyetli desenleri bulmak için yakalar.

2.8.3 Hibrit analiz

Hibrit analiz, kötü amaçlı yazılım tespiti için Android uygulamalarından çeşitli özellikler çıkarmak amacıyla statik ve dinamik analiz tekniklerinin bir kombinasyonunu kullanır. Statik analiz, yeni ve tekrarlanan kötü amaçlı yazılımları tespit etmede hafif ve etkilidir; ancak, kod karıştırma ve dinamik kod yükleme

durumlarında başarısız olabilir. Öte yandan, dinamik analiz bir uygulamanın çalışma zamanındaki davranışını yakalar [27]. Kendini gizleyen kötü amaçlı yazılımları tespit etmede daha etkilidir, ancak kontrol edilen bir ortam gerektirir ve zaman alıcı bir görevdir. Hibrit analiz bu yüzden statik ve dinamik analiz tekniklerinin eksikliklerini gidermek için kullanılır.

3. VERİ SETİ

Çalışmamızda kullanılan, zararlı yazılım içeren APK dosyaları virusshare.com sitesi üzerindeki veri tabanından elde edilmiştir. "VirusShare" web sitesi, zararlı yazılım örneklerini paylaşan ve analiz eden bir platformdur. Bu site, güvenlik uzmanları ve araştırmacılar için zararlı yazılım tehditlerini incelemek ve anlamak için bir kaynak olarak hizmet vermektedir. Sitede, farklı türlerdeki zararlı yazılım örnekleri, zararlı URL'ler, exploitlar ve diğer güvenlik tehditleri bulunabilir. VirusShare web sitesi, zararlı yazılım analizi ve siber güvenlik araştırmaları yapan kişiler için önemli bir kaynaktır. Araştırmacılar, bu platform üzerinden zararlı yazılım örneklerini indirip analiz ederek, yeni tehditleri belirleme, zararlı yazılım ailelerini sınıflandırma ve güvenlik çözümlerini geliştirme gibi çalışmalar yapabilirler. Bu şekilde, siber saldırıların tespiti, önlenmesi ve mücadelesi için değerli bilgiler elde edilebilir.

Zararlı yazılım içeren APK dosyaları 2022 yılının 3.çeyreğinde yayımlanmış olup, toplamda 20206 adet örnek içeren veri dosyası torrent üzerinden deneme ortamına indirilmiştir. Bu örneklerden yayınlanma güncelliğine ve dosya boyutuna göre arasından 3098 ‘i seçilerek çalışmada kullanılmıştır.

Çalışmamızda kullanılan zararsız APK dosyaları ise bilimsel çalışma amaçlı kullanılan AndroZoo veri tabanı üzerinden elde edilmiştir. AndroZoo web sitesinde, SHA256 hash değerleriyle tanımlanan mevcut APK' ların düzenli olarak güncellenen bir listesini mevcuttur. Her uygulama için derleme tarihi, kötü amaçlı olarak tanımlayan antivirüs motorlarının sayısı (virustotal tarafından taranmaktadır), APK 'nın boyutu, dex kodunun boyutu, ana paket adı, sürüm ve uygulamanın indirildiği pazar gibi metaverileri sağlanmaktadır. Bu bilgilerle, araştırmacılar, istenen bir APK 'nın SHA256 hash değerini belirterek AndroZoo 'dan almak istenilen alt küme seçilebilir. Biz çalışmamızda veri tabanı yöneticilerinden API anahtarı talep ederek, toplamda 5 günde 11000 zararsız APK dosyası toplanmıştır. 11000 dosya seçilirken derlenme tarihi 2020 -2023 yılları arasında seçilmiştir, virustotal tespiti “0” olan yani herhangi bir tehdit içermeyen, market olarak yalnızca Google Uygulama Mağazası üzerinde yer alan APK dosyaları seçilmiştir.

Veri setimiz 3098 zararlı APK, 11000 zararlı olmayan APK dosyası olmak üzere toplamda 14098 adet APK dosyasından meydana gelmiştir. Python programlama

dilinde “androguard ” kütüphanesi kullanılarak verilen 14098 APK dosyasının izin özellikleri çıkarılmıştır. Çıkarılan izin özelliklerinden ayırt edici olanlar (uygulamaya özel izinler değerlendirme dışına alınmıştır.) seçilerek özellik matrisi oluşturulmuştur.

Zararlı yazılım içeren APK’ lar “1” ile etiketlenmiştir, temiz APK dosyaları “0” ile etiketlenmiştir. Bu işlem sonrası son durumda toplamda 14098 APK dosyasına ait 528 adet ayırt edici özellik içeren veri matrisi elde edilmiştir.

APK File	label	android.permission.FOREGROUND_SERVICE	android.permission.READ_LOGS	android.permission.GET_ACCOUNTS
c9fe81e72ff210bb649d9d72384bba4a64	1	0	0	1
c9f3b0fce06d6c550d7c4e0120bd57f478b	0	1	0	0
c9f3254c06f23d0a3f0c43d419951beaf60	1	1	0	0
c9f079c537fba3ebb098b26b69faac39206	1	0	0	1
c9ee50df178f3c433491367b9afb5219d7c	0	0	0	0
c9e6edc814fc8d0a13ff2037293f4517fbf1	1	0	0	1
c9e627aec9847183e799de32971f29ae65	1	0	0	1
c9e1a2c9870e88975fcfa58fd1a828f67e3	1	0	0	1
c9d773269ffd2f7983d04dc30077a2a93a8	0	1	0	0
c9c852054534dc7057ea3f484613d012231	0	0	0	0
c9c70eb4a85f2555ecbfd98cf8d0642a63b	1	0	0	1
c9c69ce4363f62b3255bffbd6872de9856c	1	1	0	0
c9c36d147943e989ac54100ede68c4d418	1	0	0	0
c9c24b8873dede1aed7b9e411db6b4608	0	1	0	1
c9c0e47d41093fb1d21aa01962afaa97c7f	1	0	0	1
c9b2a8d299f4d9a539730cdcd9878d0a4d	1	0	0	0

Şekil 3.1 İzin özellikleri ile oluşturulan veri matrisi

4. MAKİNE ÖĞRENMESİ MODELİ

Makine öğrenimi alanı, bilgisayarların deneyimlerden otomatik olarak "öğrenmesine" olanak tanıyan algoritmalar ve tekniklerin incelenmesiyle ilgilenir. Makine öğrenimi istatistik, bilgi teorisi, yapay zekâ, biyoloji, felsefe ve bilişsel bilim gibi birçok alanın kavramları ve tekniklerinden yararlanır. Genel olarak, iki tür öğrenme vardır: indüktif ve dedüktif. İndüktif makine öğrenimi algoritmaları, bilinmeyen veri örneklerinden (kurallar ve desenler gibi) genelleme yapar veya bilgi çıkarır. Öte yandan, dedüktif öğrenme mevcut bilgi üzerinde çalışır ve eskiden elde edilen bilgilerden yeni bilgiler çıkarır. Makine öğrenme modelleri üç çeşittir.

- Gözetimli Makine Öğrenimi: Gözetimli öğrenmede, eğitim verilerinin sınıf etiketleri zaten bilinir. Eğitim örnekleri, bir giriş nesnesi ve istenen çıktısı (örneğin, sınıf etiketi) çiftleri olarak temsil edilir. Gözetimli bir öğrenicinin görevi, eğitim verileri ile sınıflar arasındaki eşlemeyi yaklaştırmak için bir fonksiyon bulmaktır, böylece yeni verilerin sınıflarını tahmin edebilir. Gözetimli öğrenme için birçok yaklaşım ve algoritma önerilmiştir, bunlar arasında yapay sinir ağları, naif bayes sınıflandırıcıları, karar ağaçları, K-en yakın komşu, destek vektör makineleri (DVM) ve rastgele orman bulunur.
- Gözetimsiz Makine Öğrenimi: Gözetimsiz öğrenme, eğitim verilerinin sınıf etiketlerinin mevcut olmadığı özellikleriyle denetimli öğrenmeden ayrılır. Gözetimsiz öğrenme yöntemleri, hangi nesnelerin bir sınıf olarak bir araya getirilmesi gerektiğine karar verir. Öz örgütleyen haritalar (ÖÖH) ve veri kümeleme algoritmaları genellikle gözetimsiz öğrenme görevleri için kullanılır.
- Pekiştirmeli Makine Öğrenimi: Pekiştirmeli öğrenme ödül ve ceza üzerinden çalışır. Yani, algoritma kararlarını verirken herhangi bir geri bildirim verilmez. Yalnızca sonunda doğru cevaba ulaştığında olumlu bir geri bildirim alır. Bu süreç, doğru cevaba götüren önceki kararları pekiştirir ve sağlamlaştırır. Doğru cevaplar bulunana kadar farklı cevapları keşfetmekle ilgilidir.

4.1 Öznitelik Seçimi

Özellik seçimi (ayrıca alt küme seçimi veya değişken seçimi olarak da bilinir), makine

öğreniminde yaygın olarak kullanılan bir süreçtir ve yüksek boyutluluk sorununu çözmek için kullanılır [28]. Önemli özelliklerin bir alt kümesini seçer ve gereksiz, tekrarlayan ve gürültülü özellikleri çıkararak daha basit ve özlü veri temsili elde eder. Özellik seçiminin birçok faydası vardır. İlk olarak, özellik seçimi, ilgisiz ve tekrarlayan özellikleri çıkararak bir öğrenme sürecinin çalışma zamanını büyük ölçüde azaltır. İkinci olarak, ilgisiz, tekrarlayan ve gürültülü özelliklerin müdahalesi olmadan, öğrenme algoritmaları verinin en önemli yönlerine odaklanabilir ve daha basit ama daha doğru veri modelleri oluşturabilir [29]. Bu nedenle, sınıflandırma performansı artar. Üçüncü olarak, öznitelik seçimi, daha basit ve daha genel bir model oluşturmamıza yardımcı olur ve görevin temel kavramını daha iyi anlamamızı sağlar

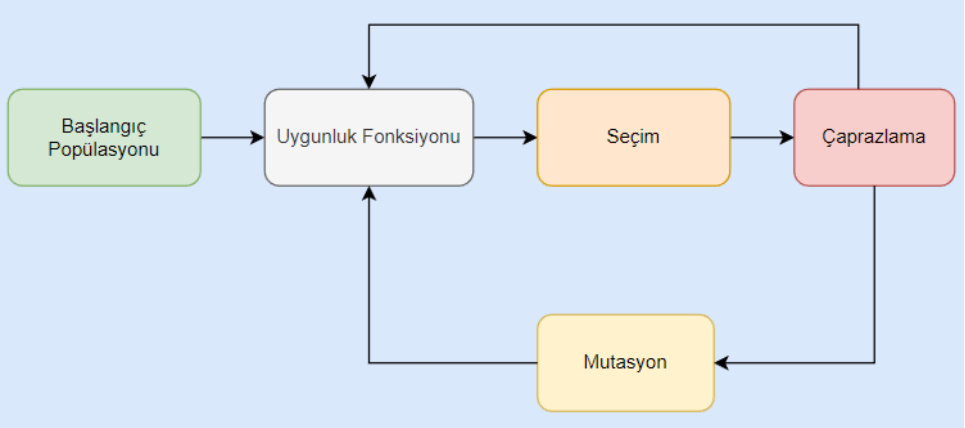
4.1.1 Genetik algoritmalar

Genetik algoritmalar (GA), evrimsel biyolojiden esinlenerek geliştirilen bir öğrenme yöntemi sağlar. Genetik algoritmalar üreme, mutasyon, çaprazlama (rekombinasyon olarak da adlandırılır), doğal seçilim ve en uygunun hayatta kalması gibi mekanizmaları kullanarak biyolojik evrimi simüle etmek için kullanılan en popüler evrimsel algoritma sınıfıdır [30].

Genetik algoritmalar, çeşitli bilimsel ve mühendislik optimizasyon veya arama problemlerine başarıyla uygulanmıştır. Karmaşık etkileşen parçalar içeren hipotezlerin arandığı alanlarda kullanılabilirler, burada her bir parçanın genel bir hipotez üzerindeki etkisi zor modellemesi olan bir durumdur [31]. Genetik algoritmaların gürültüye karşı gösterdiği göreceli hassasiyet ve hiçbir alan bilgisi gerektirmemesi, özellikle alan bilgisinin maliyetli veya mevcut olmadığı durumlarda sınıflandırma sürecini optimize etmek için güçlü bir araç yapar [32]. Çeşitli araştırmalarda öznitelik seçimi için Genetik algoritmaların avantajlarını göstermektedir [33].

Genetik algoritmalar, çözüm arayışını geleneksel olarak rastgele üretilen bir başlangıç hipotezleri popülasyonunda başlatır. Her hipotez, bir birey veya bir kromozom olarak adlandırılır ve sorunun potansiyel bir çözümünü temsil eder. Bireyler, uygulamaya bağlı olarak yorumlanan bit dizileri olarak kodlanır. Genellikle, bireyler sıfırların ve birlerin dizileri olarak ikili olarak temsil edilir. Başlangıç popülasyonu nesiller boyunca evrim geçirir. Her nesilde, mevcut popülasyonun her bireyi, soruna özgü önceden tanımlanmış bir sayısal ölçüt olan uygunluk fonksiyonu F'ye göre

değerlendirilir. Yeni bir popülasyon, mevcut en uygun bireylerin stokastik olarak seçilmesiyle oluşturulur. Seçilen bireylerin bazıları, onların bazı bölümlerini mutasyona uğratarak ve rekombinasyonla yeni nesil bireyler üretmek için değiştirilir. Seçilen bireylerin bazıları aynı şekilde bir sonraki nesile aktarılır. Yeni popülasyon daha sonra algoritmanın bir sonraki iterasyonunda kullanılır.



Şekil 4.1 Genetik algoritmanın akış şeması

Genetik operatörler (mutasyon ve çaprazlama) tarafından desteklenen rastgele arama stratejileri, birçok algoritmanın (örneğin, açgözlü algoritma, tepe tırmanması tekniği vb.) takılabileceği yerel optimumlardan popülasyonu uzaklaştırmak için tasarlanmıştır.

4.1.2 Karınca kolonisi

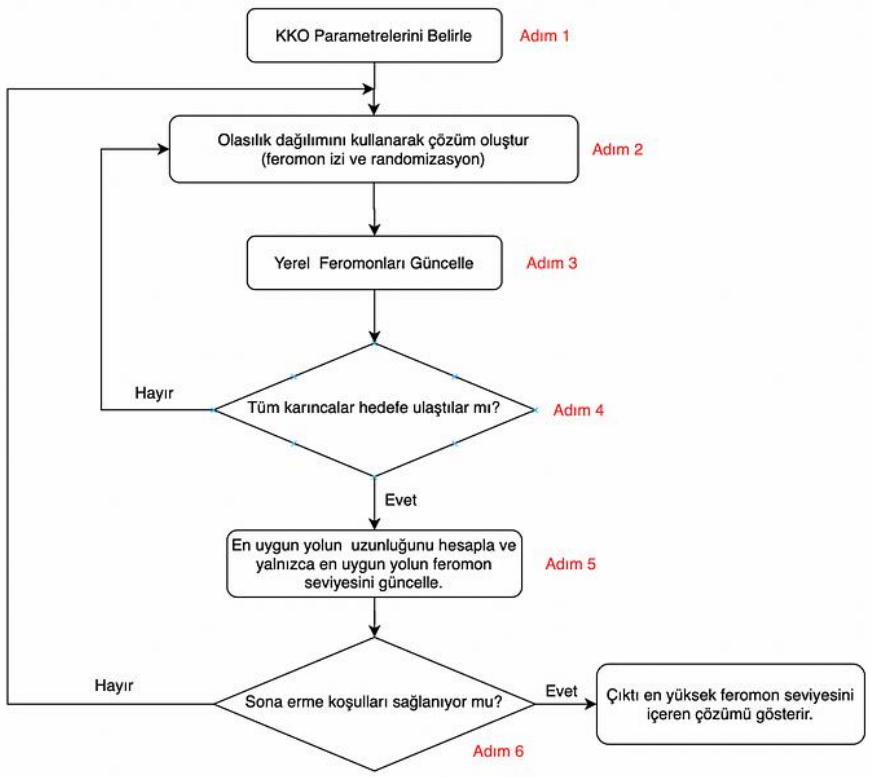
Karınca kolonisi optimizasyon algoritması, gerçek karınca kolonilerinin davranışından esinlenerek geliştirilen bir meta-sezgisel optimizasyon algoritmasıdır. Bu algoritma, 1992 yılında Marco Dorigo tarafından tanıtılan ve karıncaların yiyecek kaynaklarına ulaşmak için kullandığı iletişim ve iş birliği stratejilerini temel alır. Karınca kolonisi optimizasyonu, karmaşık optimizasyon problemlerini çözmek için kullanılır. Özellikle yolların veya rotaların en iyi şekilde belirlenmesi gereken problemlerde etkili bir şekilde kullanılabilir [34].

Algoritmanın çalışma prensibi şu şekildedir:

- İzleme (Pheromone) İzleri: Karıncalar, yiyecek kaynaklarına ulaşmak için iz bırakır. Bu izler, diğer karıncalar tarafından takip edilerek yiyecek kaynaklarına yönlendirilir. Karınca kolonisi optimizasyonunda da benzer bir

mekanizma kullanılır. Her bir çözüm adayı için bir izleme matrisi oluşturulur. Başlangıçta bu izler rasgele değerlerle başlatılır.

- Karınca Yürüyüşü: Her bir karınca, mevcut konumundan başlayarak bir çözüm adayı üretmek için bir yürüyüş gerçekleştirir. Karınca, her adımda belirli bir kurala göre kararlar alarak hareket eder. Örneğin, mevcut konumuna en yakın olmayan ancak daha yüksek izlere sahip bir konuma gitme eğilimi gösterebilir.
- İz Güncelleme: Bir karınca, hedefe ulaştığında veya yürüyüş sırasında belirli bir mesafeden sonra durduğunda, iz güncellemesi gerçekleştirilir. Başarılı bir çözüm bulunması durumunda, bu çözümün geçtiği yollar üzerindeki izler güçlendirilir. Buna karşılık başarısız bir çözüm durumunda ise izler zayıflatılır.
- Feromon Uyarısı: İzlerin güncellenmesi, feromon uyarısının diğer karıncalara iletilmesini sağlar. Diğer karıncalar, izlerin yoğunluğuna ve kendi kararlarını alırken izlerin etkisine göre hareket ederler.
- İterasyon: Belirli bir sayıda yürüyüşten (iterasyon) sonra en iyi çözüm adayları elde edilir. Bu çözümler arasından en iyisi seçilir ve sonuç olarak sunulur.



Şekil 4.2 Karınca kolonisi optimizasyonu akış şeması

Karınca kolonisi optimizasyonu, çeşitli problemlerin optimize edilmesinde etkili bir şekilde kullanılan bir algoritmadır. Karmaşık ve büyük boyutlu problemlerin çözümünde başarılı sonuçlar vermektedir. Ayrıca, paralel hesaplama ile de uygulanabilirliği artırılabilir.

4.1.3 Tepe tırmanma

Tepe tırmanma algoritması, optimizasyon problemlerinde kullanılan bir üst sezgisel algoritmadır. Amacı, bir fonksiyonun maksimum veya minimum değerini bulmaktır. Bu algoritma, problemin çözüm alanındaki bir noktadan başlayarak, komşu noktalara geçerek daha iyi bir çözüm arar [35]. Tepe tırmanma algoritması aşağıdaki adımlardan oluşur:

- Başlangıç noktası belirleme: Problemin çözüm alanında rastgele bir başlangıç noktası seçilir veya önceden belirlenmiş bir başlangıç noktası kullanılır.
- Mevcut çözüm değerini değerlendirme: Seçilen başlangıç noktasındaki çözüm değeri (fonksiyonun değeri) hesaplanır.
- Komşu çözüm arama: Mevcut çözüme komşu olan çözüm alanındaki diğer noktalar incelenir. Bu komşu noktalar, mevcut noktadan bir adım ötede yer alır. Komşu çözümlerin değerleri hesaplanır.
- Yeni çözüm değeri karşılaştırması: Komşu çözümlerden en iyi çözüm değeri seçilir. Eğer bu değer mevcut çözümün değerinden daha iyi ise, yeni çözüm olarak kabul edilir ve algoritma bu yeni çözüm noktasıyla devam eder. Aksi takdirde, algoritma sonlanır ve mevcut çözüm noktası sonuç olarak döndürülür.
- Sonlanma kontrolü: Algoritma sonlanma koşulunu kontrol eder. Örneğin, belirli bir iterasyon sayısına veya çözümün değişiminin belirli bir eşik değerini aşağı veya yukarı geçmesine göre sonlanabilir.
- Yeni çözümle tekrar başla: Algoritma, yeni çözüm noktasıyla başa dönerek işlemi tekrarlar. Yeni çözüm noktası, mevcut çözümün komşu çözümleri arasından en iyi değer olarak seçilen noktadır.

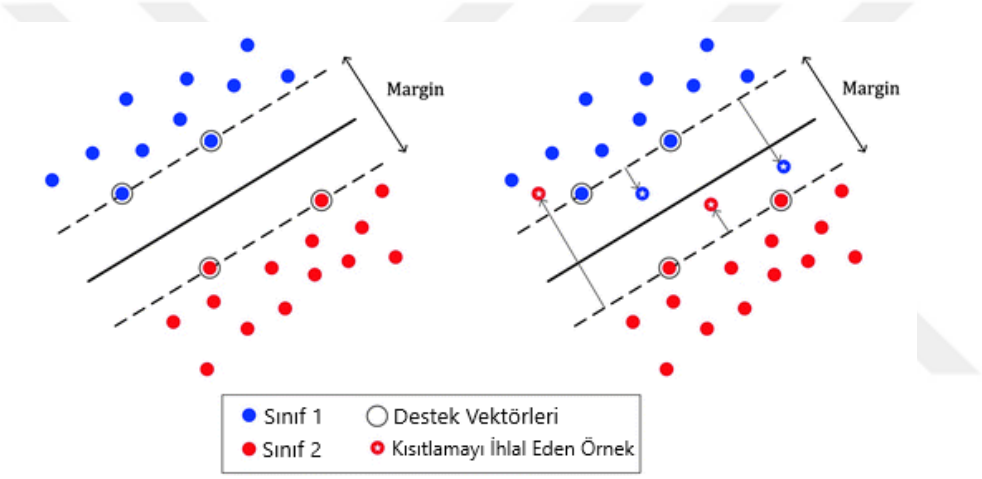
Tepe tırmanma algoritması, bir optimizasyon problemi için yerel bir optimuma ulaşabilir, ancak global optimumu garanti etmez. Bu nedenle, bazen başlangıç noktasının seçimi veya diğer tekniklerle algoritmanın başarısını artırmak için

geliştirmeler yapılabilir. Ayrıca, tepe tırmanma algoritması, bazı problemlerde çözüm alanında sıkışma veya platolar gibi zorluklarla karşılaşabilir. Bu tür durumlarda, daha sofistike üst sezgisel algoritmalar tercih edilebilir.

4.2 Sınıflandırma Algoritmaları

4.2.1 Destek vektör makineleri

Destek Vektör Makineleri (DVM), makine öğrenmesi alanında sınıflandırma ve regresyon problemlerini çözmek için kullanılan güçlü bir öğrenme algoritmasıdır. Temel olarak, DVM, veri noktalarını birbirinden ayıran bir karar sınırı (hiper düzlem) oluşturur [36].



Şekil 4.3 Destek vektör makineleri hiper düzlemi

DVM' nin çalışma prensibi şu adımlardan oluşur:

- Veri Özelliklerinin Tanımlanması: İlk adım, veri özelliklerinin tanımlanmasıdır. Özellikler, her veri noktasını tanımlayan ölçülebilir niteliklerdir.
- Veri Setinin Hazırlanması: DVM' nin eğitim aşamasında kullanılacak veri seti, etiketlenmiş verilerden oluşmalıdır. Her bir veri noktası, özellik vektörüyle birlikte doğru sınıf etiketiyle ilişkilendirilir.
- Sınıflandırma veya Regresyon Problemine Uygun Model Seçimi: DVM, sınıflandırma veya regresyon problemlerini çözmek için kullanılabilir. Sınıflandırma için, DVM' nin doğrusal DVM veya doğrusal olmayan DVM

olarak seçilmesi gerekebilir. Doğrusal DVM, veri noktalarını bir doğru veya düzlemle ayırmayı amaçlar. Doğrusal olmayan DVM ise kernel fonksiyonları kullanarak veri noktalarını karmaşık şekillerde ayırabilir [37].

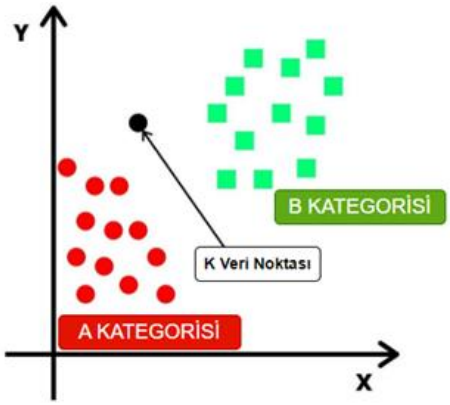
- Optimal Karar Sınırının Bulunması: DVM, doğru bir karar sınırı (hiper düzlem) oluşturmak için veri noktalarını en iyi şekilde ayırmayı hedefler. Bu, sınıflar arasında en geniş marjı (maksimum marj) elde etmek anlamına gelir. Destek vektörler, karar sınırına en yakın veri noktalarıdır ve sınıflandırmada önemli bir rol oynarlar.
- Modelin Eğitimi: Eğitim aşamasında, DVM, veri setindeki örnekler arasında bir karar sınırı oluşturmak için optimize edilir. Bu genellikle bir optimizasyon problemi olarak formüle edilir ve destek vektörlerin belirlenmesini içerir.
- Test ve Tahmin: Eğitilen DVM modeli, yeni verilerin sınıflandırılması veya tahmin edilmesi için kullanılabilir. Veri noktası, karar sınırına göre hangi sınıfa ait olduğuna karar vermek için karar fonksiyonuna (karar sınırının bir fonksiyonu) beslenir.

Destek Vektör Makinelerinin avantajları şunlardır; DVM, verileri en iyi şekilde ayırmak için optimize edildiği için yüksek sınıflandırma doğruluğu sağlar. Kernel fonksiyonları sayesinde, DVM doğrusal olmayan veri kümelerini ayırmak için kullanılabilir. DVM, eğitim sırasında yapılan hatalara karşı dayanıklıdır ve genelleştirme yeteneği yüksektir.

Destek Vektör Makinelerinin dezavantajları şunlardır; DVM, büyük veri kümelerinde ve karmaşık kernel fonksiyonlarıyla çalışırken hesaplama maliyeti yüksek olabilir. DVM, destek vektörleri kullanarak karar sınırını oluşturduğu için modelin boyutu büyük olabilir ve bellek gereksinimlerini artırabilir. DVM' nin performansı, duyarlılık parametrelerinin doğru bir şekilde ayarlanmasına bağlıdır.

4.2.2 K-en yakın komşuluğu

K-En Yakın Komşuluğu (K-EYK), makine öğrenmesi alanında sınıflandırma ve regresyon problemlerini çözmek için kullanılan basit ve yaygın kullanılan bir algoritmadır. K-EYK, bir veri noktasının sınıfını veya değerini, o noktaya en yakın komşularının sınıfı veya değeri temel alarak tahmin eder [38].



Şekil 4.4 K-en yakın komşuluğu gösterimi

K-EYK' nin çalışma prensibi şu adımlardan oluşur:

- Veri Özelliklerinin Tanımlanması: İlk adım, veri özelliklerinin tanımlanmasıdır. Özellikler, her veri noktasını tanımlayan ölçülebilir niteliklerdir.
- Veri Setinin Hazırlanması: K-EYK' nin eğitim aşamasında kullanılacak veri seti, etiketlenmiş verilerden oluşmalıdır. Her bir veri noktası, özellik vektörüyle birlikte doğru sınıf etiketi veya değeriyle ilişkilendirilir.
- K Değerinin Belirlenmesi: K-EYK' nin temel parametresi K değeridir. K, bir veri noktasının tahmin edilmesi için kullanılacak en yakın komşu sayısını belirtir. Genellikle tek sayı olarak seçilir, böylece sınıf veya değerlerde bir çoğunluk kararı alınabilir.
- Uzaklık Metriğinin Belirlenmesi: K-EYK, veri noktaları arasındaki uzaklığı hesaplamak için bir uzaklık metriği kullanır. Öklidyen mesafe en yaygın kullanılan uzaklık metriğidir, ancak Manhattan mesafesi veya diğer metrikler de kullanılabilir.
- Veri Noktasının Tahmin Edilmesi: Bir veri noktasının tahmin edilmesi için, önce K en yakın komşular belirlenir. Bu komşular, verilen uzaklık metriği kullanılarak seçilir. Ardından, bu komşuların sınıf veya değerleri gözlemlenerek, çoğunluk kararı alınır (sınıflandırma için) veya ortalama değer hesaplanır (regresyon için). Bu tahmin sonucu, veri noktasının sınıfı veya

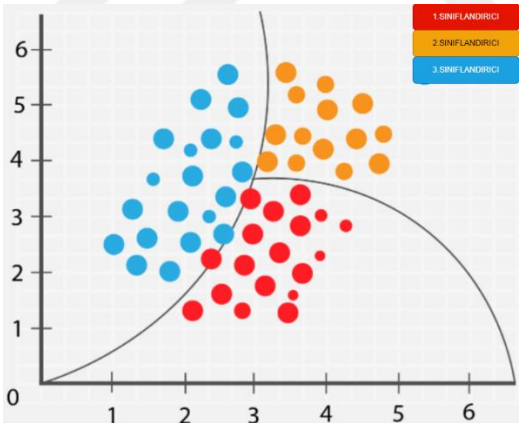
değeri olarak kullanılır.

K-EYK 'nin avantajları şunlardır; K-EYK, basit bir mantığa dayanır ve kolayca uygulanabilir. K-EYK, doğrusal veya doğrusal olmayan veri kümelerini ayırt etmek için kullanılabilir. K-EYK, eğitim sırasında herhangi bir model oluşturmadığından, yeni veri noktaları eklenip hemen kullanılabilir.

Ancak K-EYK 'nin bazı dezavantajları şunlardır; K-EYK, tahmin yapmak için veri noktalarının uzaklıklarını hesaplaması gerektiği için büyük veri kümelerinde yavaş olabilir. Yüksek boyutlu veri kümelerinde K-EYK'nin performansı düşebilir. Bu, "boyutluluk laneti" olarak bilinen bir sorundur. Eğer veri setinde sınıflar arasında dengesizlik varsa, K-EYK' nin çoğunluk sınıfa doğru eğilimli olması olasıdır.

4.2.3 Naif bayes

Naif Bayes (NB), makine öğrenmesi alanında sınıflandırma problemlerini çözmek için kullanılan bir olasılık tabanlı sınıflandırma yöntemidir. NB sınıflandırıcısı, Bayes Teoremi 'ne dayanır ve bağımsızlık varsayımını yaparak basit ve etkili bir şekilde çalışır [39].



Şekil 4.5 Naif bayes sınıflandırıcısı gösterimi

NB'nin ilk adımı, veri özelliklerinin tanımlanmasıdır. Özellikler, her veri noktasını tanımlayan ölçülebilir niteliklerdir. NB'nin eğitim aşamasında kullanılacak veri seti, etiketlenmiş verilerden oluşmalıdır. Her bir veri noktası, özellik vektörüyle birlikte doğru sınıf etiketiyle ilişkilendirilir. NB, sınıflandırma sırasında özelliklerin birbirinden bağımsız olduğunu varsayar. Bu, her özelliğin sınıfı tahmin etmek için ayrı ayrı kullanılabileceği anlamına gelir. Eğitim aşamasında, NB, sınıfların öncelik

olasılıklarını ve her özelliğin sınıf altında olasılık dağılımını hesaplar. Sınıf öncelikleri, eğitim verilerindeki sınıf etiketlerinin dağılımına dayanarak hesaplanır. Özelliklerin sınıf altında olasılık dağılımları, veri setindeki örneklerin dağılımlarına göre hesaplanır. Bir veri noktasının sınıfının tahmin edilmesi için, öncelik ve olasılık dağılımı bilgileri kullanılır. Veri noktasının her bir sınıf altında olasılığı hesaplanır ve en yüksek olasılığa sahip sınıf, tahmin edilen sınıf olarak seçilir.

NB'nin avantajları şunlardır; NB, basit bir varsayıma dayandığı ve hesaplama maliyeti düşük olduğu için hızlıdır ve büyük veri kümelerinde etkili bir şekilde çalışabilir. NB, genellikle düşük boyutlu ve metinsel veri gibi bazı uygulamalarda iyi performans gösterir. NB, eğitim için büyük miktarda veri gerektirmez. Veri seti az olduğunda bile makul bir performans sergileyebilir.

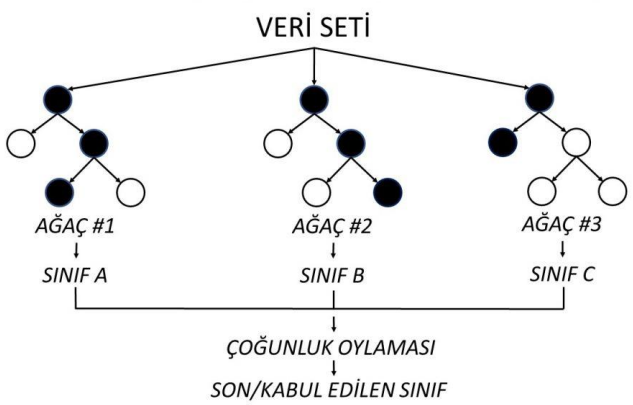
Ancak NB'nin dezavantajları şunlardır; NB, özelliklerin birbirinden bağımsız olduğunu varsayar. Gerçek hayattaki verilerde bu varsayım her zaman geçerli olmayabilir. NB, farklı kelime varyasyonlarını ayrı olarak ele aldığı için eş anlamlı kelimeler arasındaki ilişkiyi göz ardı eder. Ayrıca, eğitim verisinde bazı özelliklerin eksik olduğu durumlarda doğru tahminler yapmak zor olabilir. NB, kolay uygulanabilirliği ve etkili performansı nedeniyle birçok uygulama alanında kullanılan bir sınıflandırma yöntemidir. Veri setinin özelliklerine ve bağımsızlık varsayımının geçerliliğine dikkat edilmeli ve veri seti önceden işlenmelidir.

4.2.4 Rasgele orman

Rasgele Orman Algoritması, 1995 yılında Leo Breiman tarafından önerilmiştir .RO, ensemble öğrenme yöntemlerinden biridir ve birden fazla karar ağacını bir araya getirerek sınıflandırma veya regresyon problemlerini çözmek için kullanılır. Her bir karar ağacı, rastgele seçilen örneklemler ve rastgele seçilen özellikler üzerinde eğitilir. Sonuç olarak, her ağaç kendi sınıf tahminini yapar ve bu tahminler birleştirilerek ensemble modelin final tahmini oluşturulur [40]. Enseble öğrenme, birden fazla öğrenme algoritmasının bir araya getirilerek daha güçlü ve genelleyici bir model oluşturulmasını sağlar. RO, bu amaçla bir araya getirilen karar ağaçlarından oluşan bir modeldir.

RO algoritmasında ilk adımda, veri seti rastgele ve tekrarlı örnekleme yöntemiyle alt veri setlerine bölünür. Her alt veri seti, ana veri setinden aynı boyutta rastgele örnekler

içerir. Bu, veri setinin çeşitliliğini artırır ve her alt veri seti üzerinde ayrı ayrı karar ağaçları oluşturulmasını sağlar. Her alt veri seti üzerinde ayrı ayrı karar ağaçları oluşturulur. Karar ağaçları, veri setinin özelliklerine (bağımsız değişkenler) ve hedef değişkenine (sınıflandırma için etiketler veya regresyon için değerler) dayanarak kararlar verir. Karar ağaçları, veri setindeki ilişkileri öğrenmek ve veriye uyum sağlamak için özniteliklerin ve karar noktalarının en iyi ayrıştırılmasını arar. Oluşturulan tüm karar ağaçları kullanılarak tahmin yapılır. Sınıflandırma problemlerinde, çoğunluk oylaması yöntemiyle sınıf tahminleri birleştirilir ve en yaygın sınıf tahmini sonucu olarak seçilir. Regresyon problemlerinde, ağaçların tahminleri ortalanarak final tahmin elde edilir.



Şekil 4.6 Rasgele orman algoritması

Rasgele Orman algoritmasının çeşitli avantajları şunlardır; Ağaçların bağımsız olarak oluşturulması ve çeşitli veri setlerinin kullanılması nedeniyle aşırı öğrenmeye karşı daha dirençlidir. Bu, modelin genelleştirme yeteneğini artırır ve yeni verilere daha iyi uyarlanmasını sağlar.RO, her bir özniteliğin sınıflandırma veya regresyon performansındaki önemini ölçebilir. Öznitelik önemi, veri setinin hangi özelliklerinin daha etkili olduğunu anlamak için kullanılabilir.RO sınıflandırma ve regresyon problemleri gibi farklı görevlerde etkili bir şekilde kullanılabilir. Ayrıca, kategorik ve sürekli verileri destekler ve eksik değerleri ele alabilir.RO gücünü birden fazla karar ağacının bir araya getirilmesinden ve çeşitli alt veri setleri üzerinde çalışmasından alır. Bu sayede, veri setindeki karmaşık ilişkileri öğrenme yeteneği yüksek bir model oluşturur.

4.2.5 Çok katmanlı algılayıcı

Çok Katmanlı Algılayıcı, Makine Öğrenmesi ve Derin Öğrenme alanında yaygın olarak kullanılan bir yapay sinir ağı modelidir. ÇKA, giriş katmanı, bir veya daha fazla gizli katman ve çıkış katmanından oluşur. Her katman, birbirine tamamen bağlı düğümlerden oluşur. ÇKA, karmaşık ve doğrusal olmayan ilişkileri öğrenme yeteneğine sahip bir modeldir [41]. ÇKA 'ların temel bileşenleri şunlardır:

- Giriş Katmanı: ÇKA'nin ilk katmanıdır. Bu katmanda, veri özellikleri giriş düğümlerine bağlanır. Her bir özellik, ağırlık değeri ile çarpılarak gizli katmana iletilir.
- Gizli Katman(lar): ÇKA 'da bir veya daha fazla gizli katman bulunabilir. Gizli katmanlar, giriş katmanından aldıkları değerleri işleyerek çıkış katmanına iletmek için kullanılır. Her gizli katman, bir veya daha fazla düğümden oluşur ve bu düğümlerde aktivasyon fonksiyonları uygulanır. Aktivasyon fonksiyonları, gizli katmanlardaki düğümlerin çıktılarını belirlemek için kullanılır ve non-lineerlik sağlar. Yaygın kullanılan aktivasyon fonksiyonları arasında Sigmoid, ReLU (Rectified Linear Unit) ve Tanh bulunur.
- Ağırlık Matrisleri: ÇKA'nin her bağlantıda ağırlık değerleri bulunur. Bu ağırlıklar, giriş verileriyle çarpılarak gizli katmanlardaki düğümlere iletilir. Ağırlıklar, eğitim sürecinde optimize edilir ve öğrenme işlemi boyunca güncellenir.
- Bias Terimleri: ÇKA 'nin her katmanında bir bias terimi bulunur. Bias terimleri, ilgili katmandaki düğümlere eklenerek aktivasyon fonksiyonuna giriş olarak verilir. Bias terimleri, modele esneklik katar ve sıfır girişlerde bile düğümlerin aktive olabilmesini sağlar.
- Çıkış Katmanı: ÇKA'nin son katmanıdır. Bu katmanda, gizli katmanlardan gelen çıktılar toplanır ve sonuç olarak modelin tahminini temsil eden bir çıkış üretilir. Çıkış katmanındaki düğümlerde genellikle aktivasyon fonksiyonları kullanılmaz. Örneğin, sınıflandırma problemlerinde çıkış katmanında softmax aktivasyon fonksiyonu kullanılarak sınıf olasılıkları hesaplanabilir.
- İleri Yayılım: ÇKA 'daki veri işleme sürecidir. Giriş verileri, gizli

katmanlardan geçerek çıkış katmanına doğru ilerler. Her katmandaki düğümlerde aktivasyon fonksiyonları uygulanır ve çıktılar hesaplanır.

- Geri Yayılım: ÇKA 'nın eğitim sürecinde kullanılan bir optimizasyon algoritmasıdır. Geri yayılım, hedef çıktılarla elde edilen tahmin çıktıları arasındaki hata miktarını geriye doğru hesaplar ve bu hata miktarına göre ağırlık değerlerini günceller. Geri yayılım algoritması, hata miktarının azaltılmasına ve modelin daha iyi sonuçlar elde etmesine yardımcı olur.

ÇKA, geniş bir uygulama yelpazesine sahiptir ve sınıflandırma, regresyon, desen tanıma, doğal dil işleme gibi birçok problemin çözümünde kullanılabilir. Yüksek esneklik ve güçlü öğrenme yetenekleri sayesinde ÇKA, karmaşık veri yapılarını ve doğrusal olmayan ilişkileri modelleyebilme kabiliyetine sahiptir. Ancak, büyük veri setleri veya çok karmaşık problemlerle karşılaşıldığında eğitim sürecinde zaman ve hesaplama gücü açısından yoğun olabilir.

4.2.6 Gradyan artırma

Gradyan artırma, karmaşık tahmin modelleri oluşturmak için kullanılan bir makine öğrenmesi yöntemidir. Özellikle sınıflandırma ve regresyon problemlerinde etkilidir. Gradyan artırma, zayıf öğrenicilerin (genellikle karar ağaçları) ardışık bir şekilde birleştirilmesiyle güçlü bir öğrenici oluşturmayı hedefler.

Gradyan artırmanın temel amacı, bir hata fonksiyonunu (kayıp fonksiyonu) minimize ederek, ardışık olarak oluşturulan modellerin hatalarını azaltmaktır. Bu, önceki modelin hatalarını takiben yeni bir modelin oluşturulması ve bu sürecin birkaç kez tekrarlanmasıyla gerçekleştirilir. Her bir ardışık model, önceki modele göre kalan hatayı en aza indirecek şekilde ayarlanır. Bu şekilde, bir dizi zayıf öğrenici bir araya getirilerek güçlü bir tahminci elde edilir. Gradyan artırma uygulamasında genellikle regresyon problemleri için bir sabit değer olarak belirlenir. Kayıp fonksiyon tanımlanır. Kayıp fonksiyon gerçek değerlerle tahminler arasındaki farkı ölçen bir fonksiyondur. Kayıp fonksiyonunun, mevcut model tahminlerine göre türevi hesaplanır. Hesaplanan gradyan kullanılarak yeni bir model oluşturulur. Bu model, mevcut hatayı en aza indirecek şekilde ayarlanır. Yeni model, önceki modele eklenir ve tahminler birleştirilir. Oluşturulan modelin performansı kontrol edilir. Eğer belirlenen bir kriter sağlanmazsa, adımlar sonuç alınana kadar 3-5 kez tekrarlanır [42].

Gradyan artırma, birçok avantajı olan güçlü bir makine öğrenimi yöntemidir. Özellikle büyük veri kümeleriyle çalışırken yüksek performans gösterir. Ayrıca, doğrusal ve doğrusal olmayan ilişkileri keşfetme yeteneği ve aykırı değerlere karşı direnciyle de dikkat çeker. Bununla birlikte, hiper parametrelerin doğru şekilde ayarlanması ve eğitim süresinin uzun olabilmesi gibi bazı zorlukları da vardır.

Bu çalışmada popüler gradyan artırma algoritmalarından olan XGBoost ve LightGBM yöntemleri kullanılmıştır. Etkili ve hızlı uygulanabilirlikte olması nedeni ile bu iki algoritma tercih edilmiştir.

4.2.7 LightGBM

LightGBM hızlı, ölçeklenebilir ve yüksek performanslı bir gradyan artırma çerçevesidir. Microsoft tarafından geliştirilen LightGBM, büyük veri kümeleriyle çalışırken etkili bir şekilde kullanılabilir. Geleneksel gradyan artırma yöntemlerine kıyasla daha hızlı eğitim süreleri sağlayarak verimliliği artırır. LightGBM, karar ağaçlarından oluşan bir ensemble modelidir. En önemli özelliklerinden biri, düşey büyüme stratejisini kullanmasıdır. Geleneksel yöntemlerde ağaçlar seviye seviye büyürken, LightGBM ağaçları yaprak düğümlerinden başlayarak yukarı doğru büyütür. Bu şekilde, daha az dallanma adımı gerçekleştirir ve daha az bellek kullanır. Buna ek olarak, LightGBM, veri kümesinin özelliklerine dayalı olarak verimli bir şekilde bölme yapar, bu da hızlı bir eğitim süreci sağlar [43]. LightGBM'in diğer önemli özellikleri şunlardır:

- Ölçeklenebilirlik: LightGBM, büyük veri kümeleriyle çalışırken yüksek performans gösterir. Paralel hesaplama yöntemleri ve veri parçalama stratejileriyle ölçeklenebilirlik sağlar.
- Kategorik Özellik Desteği: LightGBM, kategorik özellikleri doğrudan işleyebilme yeteneğine sahiptir. Kategorik özelliklerin otomatik olarak sayısal formata dönüştürülmesine veya kodlanmasına gerek kalmaz.
- Düşük Bellek Kullanımı: LightGBM, bellek kullanımını optimize eder ve daha az bellek gerektirir. Veri setinin sıkıştırılması ve ağaç büyütme sürecinde belli alanlar için bellek tasarrufu sağlaması gibi teknikler kullanır.
- Performans Ayarları: LightGBM, çeşitli hiper parametrelerle gelişmiş

performans ayarlarına sahiptir. Eğitim sürecini hızlandırmak için özel optimizasyon algoritmaları kullanılır.

- Daha İyi Genelleştirme: LightGBM, düşey büyüme stratejisi sayesinde daha düşük bir eğitim hatası elde eder ve daha iyi genelleştirme yeteneği gösterir.

LightGBM, sınıflandırma, regresyon ve sıralama gibi birçok makine öğrenimi probleminde kullanılabilir. Hem Python hem de R dilleri için kullanımı kolay API'ler sunar. LightGBM, yüksek performansı, hızlı eğitim süreleri ve ölçeklenebilirliği nedeniyle makine öğrenimi uygulamalarında tercih edilen bir seçenektir.

4.2.8 XGBoost

XGBoost, makine öğrenimi alanında kullanılan güçlü bir öğrenme algoritmasıdır. Özellikle yapay zekâ, veri madenciliği ve doğal dil işleme gibi birçok uygulamada etkili sonuçlar verir. XGBoost, özellikle yapılandırılmış veriler üzerinde sınıflandırma ve regresyon problemlerini çözmek için yaygın olarak kullanılan bir tekniktir. XGBoost, gradyan artırma yöntemlerine dayanan bir ağaç tabanlı bir modeldir. Özelleştirilmiş düzenlemeli bir ağaç büyütme yöntemi kullanır. Tüm veri noktalarının yer aldığı yapraklar ve düzenli hedef bölgeleri üzerinde ağaçların büyütülmesi gerçekleştirir. Bu yöntemde, eğitim verileri bellekte tutulur ve büyük veri kümelerinde daha fazla bellek tüketimi gerekebilir. XGBoost, önceki tahminlerin hatalarını minimize etmeye çalışarak her bir ardışık ağaçla birlikte öğrenir. [44] XGBoost, bazı önemli özelliklere sahiptir:

- Ölçeklenebilirlik: Büyük miktarda veriyle etkili bir şekilde çalışabilen paralel hesaplama ve dağıtık sistemlerle uyumludur. Bu, hızlı eğitim süreleri sağlar.
- Hızlı ve verimli tahmin: Optimize edilmiş hesaplama algoritmaları kullanarak hızlı bir şekilde tahminler yapabilir. Ayrıca, düşük bellek kullanımı sayesinde büyük veri kümeleriyle de etkili bir şekilde çalışabilir.
- Düzenleme ve kontrol: Aşırı öğrenmeyle başa çıkabilen düzenleme parametrelerine sahiptir. Ayrıca, ağaç büyüklüğü, derinlik ve düğüm bölme kriterleri gibi çeşitli parametrelerle modelin davranışını kontrol etme imkânı sunar.
- Özellik önemi: Her bir öznitelik için önem puanları sağlayarak veri setinin

özniteliklerinin katkısını değerlendirmenizi sağlar. Bu, veri keşfi ve model anlayışınızı artırmaya yardımcı olabilir.

XGBoost eksik değerleri ve aykırı değerleri doğrudan ele alabilen, düzenli olarak kullanılan bir veri ön işleme sürecine de sahiptir. Çoklu sınıflandırma, regresyon, sıralama ve hatta zaman serisi analizi gibi bir dizi görevi çözebilir. Genel olarak, XGBoost gücünü yüksek tahmin doğruluğundan, hızlı eğitim sürelerinden, ölçeklenebilirlikten ve esneklikten alır. Bu özellikleri sayesinde popüler bir makine öğrenme algoritması haline gelmiştir.

4.3 Model Değerlendirme Metrikleri

Makine öğrenme modellerinin etkinliğini ölçmek için birkaç ölçüt mevcuttur. Her bir algoritmanın farklı bir değerlendirme yaklaşımı vardır. Bu çalışmada F1, doğruluk, hassasiyet, özgüllük, ROC eğrisi altında kalan alan, Matthews Korelasyon Katsayısı değerleri kullanılmıştır.

- F1 Değeri: Bir sınıflandırma modelinin hassasiyet ve özgüllük metriklerinin birleşik bir ölçüsüdür. F1 skoru, bir modelin dengeli bir şekilde hem doğruluğu hem de eksikliği değerlendirmesine yardımcı olur. F1 skoru, sınıflandırma problemlerinde özellikle dengesiz sınıf dağılımlarında kullanılır. F1 skoru, hassasiyet ve özgüllük metriklerinin harmonik ortalamasını temsil eder. Harmonik ortalama, aritmetik ortalamadan farklı olarak, daha düşük değerlere daha fazla ağırlık verir. F1 skoru, aşağıdaki formülle hesaplanır:

$$F1 = 2\,X(Kesinlik\ X\ Duyarlılık)/(Kesinlik + Duyarlılık) \quad (4.1)$$

$$Kesinlik\ =\ DP\ /(DP\ +\ YP) \quad (4.2)$$

$$Duyarlılık\ =\ DP\ /\ (DP\ +\ YN) \quad (4.3)$$

Burada:

DP, gerçek pozitifleri temsil eder. Yani doğru bir şekilde pozitif olarak tahmin edilen örnek sayısıdır. YP, yanlış pozitifleri temsil eder. Yani yanlış bir şekilde pozitif olarak tahmin edilen örnek sayısıdır. YN, yanlış negatifleri temsil eder. Yani yanlış bir şekilde negatif olarak tahmin edilen örnek sayısıdır.

F1 skoru, 0 ile 1 arasında bir değer alır. 1'e yaklaşan bir F1 skoru, daha iyi bir sınıflandırma performansını gösterirken, 0'ın yakınındaki bir skor, daha düşük bir performansı temsil eder.

- Doğruluk: Bir sınıflandırma modelinin doğru tahminlerin toplam verilere oranını temsil eder. Doğruluk, genel olarak bir modelin performansını değerlendirmek için kullanılan temel bir metriktir. Doğruluk skoru, aşağıdaki formülle hesaplanır:

$$Doğruluk = (DP + DN) / (DP + DN + YP + YN) \quad (4.4)$$

Burada:

YP, yanlış pozitifleri temsil eder. Yani yanlış bir şekilde pozitif olarak tahmin edilen örnek sayısıdır. Doğruluk skoru, 0 ile 1 arasında bir değer alır. 1'e yaklaşan bir doğruluk skoru, daha iyi bir performansı temsil ederken, 0'a yakın bir skor, daha düşük bir performansı gösterir. Ancak, doğruluk skoru dengesiz sınıf dağılımlarında yanıltıcı olabilir ve bazen diğer metriklerle birlikte kullanılması daha iyi bir değerlendirme sağlar.

- Kesinlik: Sınıflandırma modelinin pozitif olarak tahmin ettiği örneklerin gerçek pozitif örneklerin oranını temsil eder. Kesinlik, yanlış pozitiflerin minimize edilmesi gereken durumlarda önemli bir metriktir. Kesinlik skoru, aşağıdaki formülle hesaplanır:

$$Kesinlik = DP / (DP + YP) \quad (4.5)$$

Kesinlik skoru, 0 ile 1 arasında bir değer alır. 1'e yaklaşan bir hassasiyet skoru, daha iyi bir performansı temsil ederken, 0'a yakın bir skor, daha düşük bir performansı gösterir. Kesinlik, yanlış pozitiflerin önemli olduğu durumlarda dikkate alınması gereken bir metriktir.

- Duyarlılık: Sınıflandırma modelinin gerçek pozitif örneklerin doğru olarak sınıflandırılan örneklerin oranını temsil eder. Duyarlılık, yanlış negatiflerin minimize edilmesi gereken durumlarda önemli bir metriktir. Duyarlılık skoru, aşağıdaki formülle hesaplanır:

$$Duyarlılık = DP / (DP + YN) \quad (4.6)$$

Duyarlılık skoru, 0 ile 1 arasında bir değer alır. 1'e yaklaşan bir duyarlılık skoru, daha iyi bir performansı temsil ederken, 0'a yakın bir skor, daha düşük bir performansı gösterir. Duyarlılık, yanlış negatiflerin önemli olduğu durumlarda dikkate alınması gereken bir metriktir.

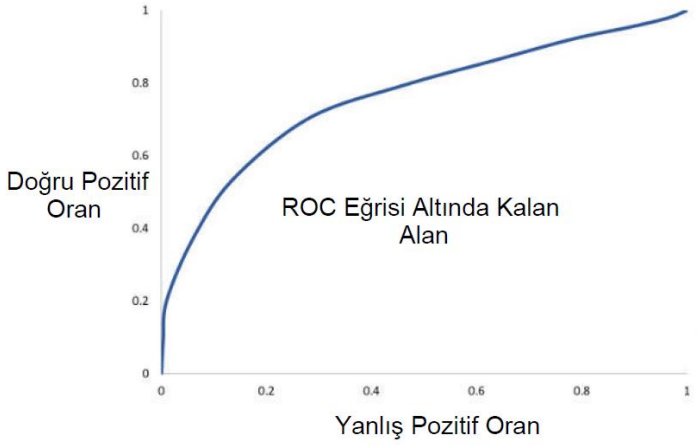
- ROC Eğrisi Altında Kalan Alan: Bir sınıflandırma modelinin performansını değerlendirmek için kullanılan bir metriktir. ROC eğrisi, modelin farklı kesim noktalarında hassasiyet ve yanlış pozitif oranını gösteren bir grafiktir. ROC eğrisi altında kalan alan, ROC AUC skoru olarak adlandırılır ve modelin performansının bir ölçüsüdür. ROC AUC skoru, 0 ile 1 arasında bir değer alır. 1'e yaklaşan bir ROC AUC skoru, daha iyi bir performansı temsil ederken, 0.5'e yakın bir skor, rastgele tahmin etmekle aynı performansı gösterir. ROC AUC skoru hesaplamak için aşağıdaki adımları izlenir:

  Modelin çıktılarına ve gerçek etiketlere (etiketli veri setindeki doğru etiketler) ihtiyaç vardır. Modelin tahmin ettiği olasılıkları kullanarak, sınırlayıcı bir eşik değeri belirleyin (örneğin, 0.5). Her bir örnek için modelin tahmin ettiği olasılık, eşik değerinden büyükse pozitif olarak kabul edilir, aksi takdirde negatif olarak kabul edilir. Gerçek pozitif oranı (DPO) ve yanlış pozitif oranı (YPO) değerlerini hesaplamak için farklı eşik değerlerini deneyin.

$$DPO = DP / (DP + YN) \quad (4.7)$$

$$YPO = YP / (YP + DN) \quad (4.8)$$

DPO ve YPO değerlerini birleştirerek bir ROC eğrisi çizilir.



Şekil 4.7 ROC eğrisi altında kalan alan

ROC eğrisi altında kalan alanı hesaplayarak ROC AUC skorunu elde edilir. Bu genellikle trapez yöntemi veya integral kullanılarak hesaplanır.

ROC AUC skoru, modelin sınıflandırma yeteneğini ölçer ve 1'e yaklaştıkça daha iyi bir performansı temsil eder.

- Matthews Korelasyon Katsayısı: Sınıflandırma modellerinin performansını değerlendirmek için kullanılan bir metriktir. Matthews korelasyon katsayısı, sınıflandırma sonuçlarının doğruluğunu, dengesiz sınıf dağılımlarını ve hata türlerini dikkate alır. Matthews korelasyon katsayısı, -1 ile +1 arasında bir değer alır. +1, mükemmel bir sınıflandırma performansını, 0 ise rastgele sınıflandırma sonuçlarını, -1 ise tamamen ters sınıflandırmayı temsil eder. Matthews korelasyon katsayısı, aşağıdaki formülle hesaplanır:

$$MKK = (DP\ X\ DN\ -\ YP\ X\ YN)\ /\ \sqrt{((DP\ +\ YP)\ (DP\ +\ YN)\ (DN\ +\ YP)\ (DN\ +\ YN))} \quad (4.9)$$

Matthews korelasyon katsayısı, sınıflandırma modelinin performansını değerlendirmek için kullanılır. +1'e yaklaşan bir Matthews katsayısı skoru, daha iyi bir performansı temsil ederken, -1'e yakın bir skor, daha düşük bir performansı gösterir. Matthews katsayısı, özellikle dengesiz sınıf dağılımları ve hata türlerinin önemli olduğu durumlarda tercih edilen bir metriktir.

5. DENEYSEL SONUÇLAR

Tersine mühendislik yöntemleri ile 3098 ‘i zararlı APK, 11000 ‘i zararlı olmayan APK dosyası olmak üzere toplamda 14098 APK dosyasından 2450 farklı izin özelliği elde edildi. Bu izin özelliklerinden uygulamaya özel olan izinler sadeleştirilerek 528 adet ayırt edici izin özelliği elde edilmiştir. Çalışmamızda doğrulama yöntemi olarak k-katlamalı çapraz doğrulama metodu kullanılmıştır. K-Katlamalı Çapraz Doğrulama, Makine öğrenmesi ve istatistiksel modelleme gibi alanlarda yaygın olarak kullanılan bir doğrulama yöntemidir. Bu yöntem, veri kümesinin doğrulama ve eğitim için bölünmesini içerir. K-katlamalı çapraz doğrulama, veri kümesini k adet alt kümeye böler ve her biri sırayla doğrulama seti olarak kullanılırken diğer alt kümeler birleştirilerek eğitim seti oluşturulur. Yani, veri kümesi rastgele şekilde k parçaya bölünür ve her bir parça dönüşümlü olarak doğrulama için kullanılır. Bu işlem, tüm alt kümelerin doğrulama olarak kullanıldığı kadar tekrarlanır [45]. Her bir tekrarda, model eğitim seti üzerinde eğitilir ve doğrulama seti üzerinde test edilir. Doğrulama işlemi sonucunda elde edilen performans ölçümleri (genellikle hata oranı veya doğruluk gibi) kaydedilir. K-katlamalı çapraz doğrulama işlemi tamamlandığında, her bir tekrarın performans ölçümleri ortalaması alınarak genel bir performans değeri elde edilir. K-katlamalı çapraz doğrulamanın avantajları şunlardır; Veri seti hem eğitim hem de doğrulama için kullanılır, bu da veri setinin daha verimli kullanılmasını sağlar. Veri setinin farklı alt kümeleri üzerinde yapılan doğrulama, modelin genelleştirilebilirliğini artırır ve aşırı uydurma sorununu azaltır. K-katlamalı çapraz doğrulama, modelin performansını daha güvenilir bir şekilde değerlendirmek için kullanılır. Tek bir eğitim-doğrulama bölümlemesine göre daha kapsamlı bir sonuç elde etmek mümkündür. Ancak, K-katlamalı çapraz doğrulamanın bazı dezavantajları da vardır; K değeri büyüdükçe, modelin eğitim ve doğrulama süreleri artar. Yüksek k değerleri, daha uzun süren bir doğrulama sürecine neden olabilir. Eğer veri seti dengesiz bir şekilde dağılmışsa (örneğin, sınıflar arasında büyük bir dengesizlik varsa), performansı etkileyebilir. K-katlamalı çapraz doğrulama, modelin performansını değerlendirmek ve hiper parametre ayarlaması yapmak için yaygın olarak kullanılan bir doğrulama yöntemidir. Bu yöntem, modelin genelleştirilebilirliğini iyileştirir ve güvenilir performans ölçümleri sağlar. Biz çalışmamızda k değerini 10 alarak ölçümlerimizi gerçekleştirdik.

Elde edilen özellik veri setinde Destek Vektör Makineleri (DVM), k-En yakın komşu algoritması(K-EYK), Naif Bayes(NB), Rasgele Orman (RO), Çok katmanlı Algılayıcı (ÇKA), LightGBM ve XGBoost sınıflandırıcı algoritmaları test ettik. Testler sonucunda f1 skoru, doğruluk, hassasiyet, özgüllük, ROC eğrisi altında kalan alan, Matthews korelasyon katsayısı başarım metriklerine göre sonuçlar elde ettik.

Tablo 5.1 Veri setinde çoklu sınıflandırma başarımları

Algoritmalar	F1 Değeri	Doğruluk	Kesinlik	Duyarlılık	ROC AUC	MCC
DVM	0,9008	0,9586	0,9520	0,8551	0,9214	0,8769
K-EYK	0,8849	0,9524	0,9437	0,8331	0,9096	0,8577
Naif Bayes	0,6192	0,8693	0,8605	0,4848	0,7312	0,5812
Rasgele Orman	0,9095	0,9615	0,9405	0,8806	0,9324	0,8859
ÇKA	0,9005	0,9577	0,9313	0,8719	0,9269	0,8746
LightGBM	0,9134	0,9631	0,9436	0,8854	0,9352	0,8901
XGBoost	0,9073	0,9608	0,9445	0,8731	0,9293	0,8837

Tablodan 5.1 incelendiğinde Gradyan artırma temelli LightGBM sınıflandırıcısı ile %91 f1 değeri, %96 doğruluk, %89 duyarlılık değerleri yakalanmıştır.

Daha sonra Genetik algoritma kullanarak başarım metrikleri gözlemlendi. Veri setine Genetik Algoritma uygulandığında benzersiz özellik sayısı 275 ‘e düştü. Bu 275 izin özelliğinin aynı sınıflandırma algoritmalarına ait performans çıktısı tablo 5.2 ‘de verilmiştir.

Tablo 5.2 Genetik Algoritma uygulandıktan sonra başarım metrikleri

Algoritmalar	F1 Değeri	Doğruluk	Kesinlik	Duyarlılık	ROC AUC	MCC
DVM	0,8871	0,9535	0,9509	0,8315	0,9097	0,8612
K-EYK	0,8783	0,9492	0,9276	0,8341	0,9079	0,8483
Naif Bayes	0,6509	0,8756	0,8474	0,5300	0,7515	0,6046
Rasgele Orman	0,8997	0,9576	0,9365	0,8661	0,9247	0,8741
ÇKA	0,8954	0,9558	0,9341	0,8602	0,9215	0,8688
LightGBM	0,9021	0,9588	0,9420	0,8657	0,9253	0,8774
XGBoost	0,8977	0,9572	0,9454	0,8548	0,9204	0,8725

Tablo 5.2 incelendiğinde Genetik Algoritma ile genel başarım miktarlarının çok az oranda düştüğü yalnızca Naif Bayes algoritmasında iyileşme olduğu gözlemlenmiştir. Bu durumda LightGBM sınıflandırıcısı ile %90 f1 değeri, %96 doğruluk, %93 ROC AUC değerleri yakalanarak en iyi performansı gösterdiği görülmüştür. Ancak Genetik Algoritma uygulandıktan sonra başarım metriklerinde azalma olduğu gözlemlenmiştir.

Veri setimiz üzerinde Genetik algoritma yerine Karınca Kolonisi Optimizasyonu tekniği kullanarak başarım metrikleri gözlemlendi. Karınca kolonisi optimizasyonu tekniği veri setine uygulandığında benzersiz izin özelliği sayısı 185'e düştü. Bu 185 izin özelliğinin aynı sınıflandırma algoritmaları uygulanarak performans çıktısı tablo 5.3 ‘de verilmiştir.

Tablo 5.3 Karınca Kolonisi tekniği uygulandıktan sonra başarım metrikleri

Algoritmalar	F1 Değeri	Doğruluk	Kesinlik	Duyarlılık	ROC AUC	MCC
DVM	0.9157	0.9615	0.9533	0.8425	0.9144	0.8703
K-EYK	0.8842	0.9511	0.9316	0.8475	0.9147	0.8588
Naif Bayes	0.6334	0.8564	0.8215	0.5154	0.7341	0.5876
Rasgele Orman	0.9245	0.9678	0.9423	0.8835	0.9390	0.8622
ÇKA	0.8954	0.9558	0.9341	0.8602	0.9215	0.8688
LightGBM	0.9376	0.9751	0.9573	0.8993	0.9521	0.9122
XGBoost	0.9143	0.9704	0.9534	0.8822	0.9423	0.9042

Karınca kolonisi optimizasyon tekniği uygulandıktan sonra Naif Bayes sınıflandırıcısı dışındaki sınıflandırıcılarda başarım performansının arttığı gözlemlenmiştir. Tablo 5.3 de görüldüğü gibi en iyi sonucu karınca kolonisi optimizasyonu tekniği ile elde ettiğimiz benzersiz izin özelliklerinin LightGBM sınıflandırıcısı değerlendirilmesinde %98 doğruluk, %96 kesinlik, %90 duyarlılık değerleri yakalanmıştır. Karınca kolonisi optimizasyon tekniği uygulandığında elde edilen başarım metrikleri, Genetik Algoritmanın sonuçları ile herhangi bir özellik seçimi yapılmadan elde edilen sonuçlardan daha iyi olduğu gözlemlenmiştir.

Veri setimiz üzerinde tepe tırmanma tekniği kullanarak başarım metrikleri gözlemlendi. Tepe tırmanma tekniği veri setine uygulandığında benzersiz izin özelliği sayısı 426 ya düştü. Bu 426 izin özelliğinin aynı sınıflandırma algoritmaları uygulanarak (LightGBM ve XGBoost dahil) ait performans çıktısı tablo 5.4 ‘de verilmiştir.

Tablo 5.4 Tepe Tırmanma tekniği uygulandıktan sonra başarım metrikleri

Algoritmalar	F1 Değeri	Doğruluk	Kesinlik	Duyarlılık	ROC AUC	MCC
DVM	0.9016	0.9583	0.9364	0.8696	0.9264	0.8763
K-EYK	0.8876	0.9536	0.9475	0.8351	0.9110	0.8614
Naif Bayes	0.8773	0.9491	0.9329	0.8283	0.9057	0.8479
Rasgele Orman	0.5945	0.8649	0.8690	0.4542	0.7174	0.5639
ÇKA	0.9002	0.9577	0.9354	0.8677	0.9254	0.8744
LightGBM	0.8949	0.9557	0.9339	0.8593	0.9210	0.8682
XGBoost	0.8983	0.9574	0.9448	0.8564	0.9211	0.8732

Tablo 5.4 ‘de görüldüğü gibi tepe tırmanma tekniği uygulandığında genel olarak başarım metriklerinde naif bayes sınıflandırıcısı haricinde düşüş gözlenmiştir. Destek Vektör Makineleri sınıflandırıcısı diğer sınıflandırıcılardan daha iyi performans göstermesine rağmen, genel olarak başarım metrikleri diğer özellik seçim algoritmalarından daha düşük ölçülmüştür.

6. SONUÇ ve ÖNERİLER

Tez çalışmasında, zararlı yazılım tespiti için güncel ve özgün bir veri seti üzerinden tersine mühendislik yöntemleri ile izin özelliklerini elde ettik. Bu elde ettiğimiz izin özelliklerinden uygulamaya özel izinleri çıkararak Android üzerinde ayırt edici izin özelliklerini seçtik. Daha sonra Genetik Algoritma kullanarak veri setindeki öznitelikler seçilmiştir. Bu aşamada 528 benzersiz izin özelliği 275 'e düşmüştür. Daha sonra elde ettiğimiz veri seti destek vektör makineleri, k-en yakın komşu, rasgele orman, naif bayes, çok katmanlı algılayıcı ağ sınıflandırmaları test edilerek rasgele orman algoritmasının diğer sınıflandırıcılara göre daha iyi sonuçlar verdiği gözlemlenmiştir. Bu başarım metriklerini artırmak için gradyan artırma tekniklerinden olan LightGBM ve XGBoost algoritmalarını, 275 izin özelliği üzerine uyguladık. Sonuç olarak LightGBM sınıflandırma algoritması diğer sınıflandırıcılardan daha yüksek başarım metrikleri ile yüksek performans gösterdi. Genetik algoritma yerine karınca kolonisi optimizasyon tekniği kullandığımızda benzersiz izin özelliği sayısı 185'e düştü. Bu özellik seti üzerinde gradyan artırma tekniklerini uyguladık. En iyi sonucu LightGBM sınıflandırma algoritması verdi. Tepe tırmanması tekniğini kullandığımızda ise benzersiz izin özelliği sayısı 426'ya düştü. Ancak başarım metriklerinde düşüş gözlendi.

Karınca kolonisi optimizasyon tekniğinin paralel çalışma ve adaptiftik özellikleri ile LightGBM sınıflandırma algoritmasının yüksek performansı, hızlı eğitim süreleri ve ölçeklenebilirliği nedeniyle deney sonuçlarında iyileşme sağladığı sonucuna varılmıştır.

Gelecek çalışmalarımızda geliştireceğimiz bir web uygulaması ile kullanıcı tarafından verilen bir APK dosyasının zararlı yazılım analizini yapay zekâ temelli, çevrimiçi ortamda gerçekleştirmeyi planlıyoruz. Ayrıca bu tez çalışması kapsamında toplanan veri setini çevrimiçi ortamda akademisyenlerin ve uzmanların kullanımına açmayı planlıyoruz.

KAYNAKÇA

[1] Statcounter, https://gs.statcounter.com/os-market-share/mobile/worldwide , (Erişim Tarihi: 2 Şubat 2023)

[2] A.BOCEREG,Real-Time Behavior-Based Detection on Android Reveals Dozens of Malicious Apps on Google Play Store, https://www.bitdefender.com/blog/labs/real-time-behavior-based-detection-on-android-reveal-dozens-of-malicious-apps-on-google-play-store/ , (Erişim Tarihi: 2 Şubat 2023)

[3] H. M. Ünver and K. Bakour, "Android malware detection based on image-based features and machine learning techniques," *SN Applied Sciences*, vol. 2, no. 7, Jun. 2020, doi: https://doi.org/10.1007/s42452-020-3132-2.

[4] A. Sangal and H. K. Verma, "A Static Feature Selection-based Android Malware Detection Using Machine Learning Techniques," *IEEE Xplore*, Sep. 01, 2020. https://ieeexplore.ieee.org/document/9215355 (Erişim Tarihi: 07, 2021).

[5] O. N. Elayan and A. M. Mustafa, "Android Malware Detection Using Deep Learning," *Procedia Computer Science*, vol. 184, pp. 847–852, 2021, doi: https://doi.org/10.1016/j.procs.2021.03.106.

[6] B. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and S. Riasat, "Malware Detection: A Framework for Reverse Engineered Android Applications through Machine Learning Algorithms," *IEEE Access*, pp. 1–1, 2022, doi: https://doi.org/10.1109/access.2022.3149053.

[7] M. R. Keyvanpour, M. Barani Shirzad, and F. Heydarian, "Android malware detection applying feature selection techniques and machine learning," *Multimedia Tools and Applications*, Sep. 2022, doi: https://doi.org/10.1007/s11042-022-13767-2.

[8] Leonid Batyuk, A.-D. Schmidt, H.-G. Schmidt, Ahmet Camtepe, and Sahin Albayrak, "Developing and Benchmarking Native Linux Applications on Android," pp. 381–392, Jan. 2009, doi: https://doi.org/10.1007/978-3-642-01802-2_28.

[9] S. P. Hall and E. Anderson, "Operating systems for mobile computing." *Journal of Computing Sciences in Colleges* (2009), 25(2), pp.64-71.

[10] S. K. Datta, C. Bonnet, A. Gyrard, R. P. Ferreira da Costa, and K. Boudaoud, "Applying Internet of Things for personalized healthcare in smart homes," *2015 24th Wireless and Optical Communication Conference (WOCC)*, Oct. 2015, doi: https://doi.org/10.1109/wocc.2015.7346198.

[11] A.M.M. Soares and de R.T. Sousa. A Technique for Extraction and Analysis of Application Heap Objects within AndroidRuntime (ART). In *ICISSP* (2017, February) (pp. 147-156).

[12] BläsingT., L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox system for suspicious software detection," *2010 5th International Conference on Malicious and Unwanted Software*, Oct. 2010, doi: https://doi.org/10.1109/malware.2010.5665792.

[13] M. Spreitzenbarth, T. Schreck, F. Echtler, D. Arp, and J. Hoffmann, "Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques," *International Journal of Information Security*, vol. 14, no. 2, pp. 141–153, Jul. 2014, doi: https://doi.org/10.1007/s10207-014-0250-0.

[14] J. Liu and J. Yu, "Research on Development of Android Applications," *2011 4th International Conference on Intelligent Networks and Intelligent Systems*, Nov. 2011, doi: https://doi.org/10.1109/icinis.2011.40.

[15] Y. Pan and Y. Xiao, *Ad Hoc and Sensor Networks*. Nova Publishers, 2006. Accessed:Feb.21,2023.[Online].Available:https://books.google.com.tr/books?hl=en&lr=&id=5uaCnj7_LkC&oi=fnd&pg=PA1&dq=Zheng

[16] L. K. Yan and H. Yin, "{DroidScope}: Seamlessly Reconstructing the {OS} and Dalvik Semantic Views for Dynamic Android Malware Analysis," *www.usenix.org*,2012.(pp569/584)https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/yan (accessed Feb. 21, 2023).

[17] P. M. Carter, C. Mulliner, M. Lindorfer, W. Robertson, and Engin Kirda, "CuriousDroid: Automated User Interface Interaction for Android Application Analysis Sandboxes," pp. 231–249, Jan. 2017, doi: https://doi.org/10.1007/978-3-662-54970-4_13.

[18] A. Bounceur *et al.*, "CupCarbon: A new platform for the design, simulation and 2D/3D visualization of radio propagation and interferences in IoT networks," *IEEE Xplore*, Jan. 01, 2018. https://ieeexplore.ieee.org/abstract/document/8319179 (accessed Jul. 21, 2023).

[19] K. Hamandi, A. Chehab, I. H. Elhajj, and A. Kayssi, "Android SMS Malware: Vulnerability and Mitigation," *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, Mar. 2013, doi: https://doi.org/10.1109/waina.2013.134.

[20] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," *IEEE Security & Privacy Magazine*, vol. 7, no. 1, pp. 50–57, Jan. 2009, doi: https://doi.org/10.1109/msp.2009.26.

[21] BläsingT., L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox system for suspicious software detection," *2010 5th International Conference on Malicious and Unwanted Software*, Oct. 2010, doi: https://doi.org/10.1109/malware.2010.5665792.

[22] A. Hoog, *Android Forensics: Investigation, Analysis and Mobile Security for Google Android*. Elsevier, 2011. Accessed: March. 15, 2023. [Online].

Available:https://books.google.com.tr/books?hl=en&lr=&id=iyWIVd4z7MC&oi=fnd&pg=PP1&dq=android+package+kit+components&ots=L_B_ZHuDLl&sig=x_85aiQ4sB631jf91Oouj3dqQI&redir_esc=y#v=onepage&q=android%20package%20kit%20components&f=false

[23] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. *In Ndss*, volume 14, pages 23‑26, 2014.

[24] S. K. Sasidharan and C. Thomas, "ProDroid — An Android malware detection framework based on profile hidden Markov model," *Pervasive and Mobile Computing*,vol.72,p.101336,Apr.2021,doi:https://doi.org/10.1016/j.pmcj.2021.101336.

[25] F. Faghihi, M. Zulkernine, and S. Ding, "CamoDroid: An Android application analysis environment resilient against sandbox evasion," *Journal of Systems Architecture*, vol. 125, p. 102452, Apr. 2022, doi: https://doi.org/10.1016/j.sysarc.2022.102452.

[26] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided Android malware classification," *Computers & Electrical Engineering*, vol. 61, pp. 266–274, Jul. 2017, doi: https://doi.org/10.1016/j.compeleceng.2017.02.013.

[27] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," *IEEE Access*, vol. 6, pp. 4321–4339, 2018, doi: https://doi.org/10.1109/access.2018.2792941.

[28] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1–2, pp. 273–324, Dec. 1997, doi: https://doi.org/10.1016/s0004-3702(97)00043-x.

[29] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, Jan. 2014, doi: https://doi.org/10.1016/j.compeleceng.2013.11.024.

[30] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, Oct. 2020, doi: https://doi.org/10.1007/s11042-020-10139-6.

[31] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994, doi: https://doi.org/10.1109/2.294849.

[32] S. Ding, C. Su, and J. Yu, "An optimizing BP neural network algorithm based on genetic algorithm," *Artificial Intelligence Review*, vol. 36, no. 2, pp. 153–162, Feb. 2011, doi: https://doi.org/10.1007/s10462-011-9208-z.

[33] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms," *IEEE Transactions on*

Evolutionary Computation, vol. 4, no. 2, pp. 164–171, Jul. 2000, doi: https://doi.org/10.1109/4235.850656.

[34] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: https://doi.org/10.1109/MCI.2006.329691.

[35] M. Mitchell, J. Holland, and S. Forrest, "When will a Genetic Algorithm Outperform Hill Climbing," *Neural Information Processing Systems*, 1993. https://proceedings.neurips.cc/paper/1993/hash/ab88b15733f543179858600245108dd8-Abstract.html (accessed May. 05, 2023).

[36] W. S. Noble, "What is a support vector machine?," *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, Dec. 2006, doi: https://doi.org/10.1038/nbt1206-1565.

[37] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, Jul. 1998, doi: https://doi.org/10.1109/5254.708428.

[38] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN Classification With Different Numbers of Nearest Neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, May 2018, doi: https://doi.org/10.1109/tnnls.2017.2673241.

[39] Liangxiao Jiang, H. Zhang, and Zhihua Cai, "A Novel Bayes Model: Hidden Naive Bayes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1361–1371, Oct. 2009, doi: https://doi.org/10.1109/tkde.2008.234.

[40] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, doi: https://doi.org/10.1023/a:1010933404324.

[41] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14–15, pp. 2627–2636, Aug. 1998, doi: https://doi.org/10.1016/s1352-2310(97)00447-0.

[42] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, Feb. 2002, doi: https://doi.org/10.1016/s0167-9473(01)00065-2.

[43] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Advances in Neural Information Processing Systems*, vol. 30, 2017, Available:https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[44] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 785–794, 2016, doi: https://doi.org/10.1145/2939672.2939785.

[45] J. D. Rodriguez, A. Perez, and J. A. Lozano, “Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, Mar. 2010, doi: https://doi.org/10.1109/tpami.2009.187.

ÖZGEÇMİŞ

İlk öğrenimini Balıkesir'de, orta öğrenimini Ankara'da tamamladı.2004 yılında girdiği Gazi Üniversitesi Elektrik-Elektronik Mühendisliği Bölümünden 2008 yılında mezun oldu.2008 yılından 2016 yılına kadar Türk Telekomünikasyon A.Ş. 'de çeşitli kademelerde haberleşme mühendisi olarak görev yaptı. 2016 yılında istifa ederek Yalova Üniversitesinde öğretim görevlisi olarak göreve başladı. Şu anda Bilgi İşlem Daire Başkanlığında ağ ve sistem sorumlusu olarak çalışmaktadır.