İzin Tabanlı Android Kötücül Amaçlı Yazılım Tespitinde İzin Ağırlıklandırma Yaklaşımları

Permission Weighting Approaches in Permission Based Android Malware Detection

Oğuz Emre Kural
Bilgisayar Mühendisliği Bölümü
Ondokuz Mayıs Üniversitesi
Samsun, Türkiye
oguz.kural@ bil.omu.edu.tr

Durmuş Özkan Şahin
Bilgisayar Mühendisliği Bölümü
Ondokuz Mayıs Üniversitesi
Samsun, Türkiye
durmus.sahin @ bil.omu.edu.tr

Sedat Akleylek
Bilgisayar Mühendisliği Bölümü
Ondokuz Mayıs Üniversitesi
Samsun, Türkiye
sedat.akleylek @ bil.omu.edu.tr

Erdal Kılıç
Bilgisayar Mühendisliği Bölümü
Ondokuz Mayıs Üniversitesi
Samsun, Türkiye
erdal.kilic @ bil.omu.edu.tr

Öz*—Mobil cihazların günlük hayattaki kullanımlarının artması ile mobil cihazlar üzerinde çalışan kötücül yazılımların sayısında da bir artış görülmektedir. Artan kötücül yazılımlar kullanıcıların kişisel bilgilerinin ele geçirilmesi veya kişisel verilerin bozulması gibi maddi ve manevi zararları meydana getirebilmektedir. Bu nedenle kötücül yazılımları yüksek doğrulukta tespit eden sistemlere duyulan ihtiyaç her geçen gün artmaktadır. Bu çalışmada Android işletim sistemleri için makine öğrenmesi tabanlı statik analiz tekniği kullanılarak kötücül yazılımların tespiti amaçlanmıştır. Kötücül yazılımların tespitinde yüksek başarım oranları elde etmek için metin sınıflandırmada sıkça kullanılan 14 farklı terim ağırlıklandırma tekniği kötücül yazılım tespitine uyarlanarak geniş kapsamlı bir çalışma yapılmıştır. Uyarlanan yöntemler 2 farklı veri kümesi üzerinde denenerek, 3 farklı sınıflandırma algoritması ile kıyaslanmıştır. AMD veri kümesi üzerinde en başarılı sınıflandırma sonucu ikili terim ağırlıklandırma tekniği ve destek vektör makinesi sınıflandırma algoritmasından elde edilmiştir. M0DROID veri kümesi üzerinde en başarılı sınıflandırma sonucu ayrım yapan uygunluk özellik çıkarma tekniği ve destek vektör makinesi sınıflandırma algoritmasından elde edilmiştir.

Anahtar Sözcükler — Android kötücül yazılım, statik analiz, mobil güvenlik, özellik çıkarımı, mobil kötücül yazılım.

Abstract— With the increasing use of mobile devices in daily life, the number of malware running on mobile devices is increasing. Increased malware may cause material and non-pecuniary damage, such as the seizure of personal information of users or the deterioration of personal data. Therefore, the need for systems that detect malware with high accuracy is increasing day by day. In this study, it is aimed to determine malware using the machine learning based static analysis technique for Android operating systems. In order to obtain high performance rates in malware detection, 14 different terms weighting techniques frequently used in text classification have been extensively adapted to this. Adapted methods were tested on 2 different datasets and compared with 3 different classification algorithms. The most successful classification result on the AMD data set was obtained from binary term weighting technique and support vector machine classification algorithm. The most successful classification result on the M0DROID data set was obtained from discriminative weighting technique and support vector machine classification algorithm.

Keywords — Android malware, static analysis, mobile security, feature extraction, mobile malware.

I. GİRİŞ

Mobil cihaz teknolojisinin gelişmesi, insanların günlük hayatlarını kolaylaştıran birçok fonksiyonu beraberinde getirmektedir. Bu gelişim sayesinde mobil cihazlar insanlar için vazgeçilemez olmakta ve olmaya da devam edecektir. 2018 yılı itibariyle dünya üzerinde 4,57 milyar cep telefonu kullanıcısı olduğu saptanmış, 2020 itibariyle de bu rakamın 4,78 milyara ulaşacağı tahmin edilmektedir [1]. Günümüzde kullanılan mobil cihazların büyük çoğunluğunu akıllı mobil cihazlar oluşturmaktadır. Bu rakam 2018 yılı itibariyle 3 milyar olurken 2021 yılı itibariyle 3,8 milyara ulaşacağı ön görülmektedir [2].

Akıllı mobil cihazlar sadece haberleşme için değil aynı zamanda bankacılık işlemleri, alışveriş, sosyal medya kullanımı gibi birçok alanda insanlar tarafından yaygınca kullanılmaktadır. Akıllı mobil cihaz kullanımı insanların işlerini kolaylaştırdığı gibi birçok tehlikeyi de peşinde getirmektedir. Çünkü akıllı mobil cihaz sayısına paralel bir artış da mobil cihazlar üzerinde çalışan kötücül yazılım sayısında görülmektedir.

Mobil cihazlarda gerçekleştirilen bankacılık işlemleri, alışveriş gibi maddi işlemlerin yanında insanların kişisel verilerinin yer aldığı sosyal medya hesaplarının, fotoğraf ve videoların bulunması bu cihazların güvenliğini ön planda tutmaktadır. Mobil cihazlardan konum bilgisi, ses kaydı, kamera kaydı gibi veriler elde edilebilmektedir. Bu bilgilerin kullanıcının haberi olmadan elde edilmesi de ayrı bir güvenlik zafiyeti olup kötücül yazılım geliştiricilerinin ilgisini çekmektedir.

Android, mobil cihazlar için geliştirilmiş açık kaynak kodlu Linux tabanlı mobil işletim sistemidir. Android, dünyada en çok kullanılan mobil işletim sistemidir. 2018 yılı içerisinde mobil cihaz satışları incelendiğinde satılan ürünlerin %88'ini Android işletim sistemli cihazlar oluşturmaktadır [3]. Android işletim sistemli cihazların yaygın kullanımı kötü niyetli yazılım geliştiricilerinin odak noktası haline gelmektedir. Bu nedenle piyasaya sürülen

978-1-7281-3964-7/19/$31.00 ©2019 IEEE

kötücül yazılım sayısı her geçen gün artmaktadır. 2014 yılına kadar Android işletim sistemi üzerinde çalışabilen 3 milyondan fazla kötücül yazılım olduğu tespit edilirken, her ay 279 bin yeni kötücül yazılımında piyasaya sürüldüğü vurgulanmaktadır [4]. Bu nedenle araştırmacılar ve şirketler kötücül yazılımların yüksek doğrulukta tespit edilmesi için çalışmalarını sürdürmektedir.

Kötücül yazılım tespitinde çoğunlukla iki yola başvurulur. Bunlar imza tabanlı teknikler ve anomali tabanlı tekniklerdir [5]. İmza ve anomali tabanlı yaklaşımlarda statik, dinamik ve iki yöntemin birleşimi olan karma analiz olmak üzere üç farklı şekilde yapılmaktadır. Statik analizde uygulama çalıştırılmadan kaynak kod incelemesi veya API (Application Programming Interface) çağrıları elde edilerek uygulama hakkında bilgi elde edilmektedir. Dinamik analizde ise uygulama çalıştırılarak uygulamanın davranışları incelenir ardından uygulamanın iyicil mi kötücül mü olduğuna karar verilmektedir. Her iki yönteminde birbirlerine göre artı ve eksi yönleri mevcuttur [6]. Statik ve dinamik analiz yöntemlerinin bir birlerine göre artı ve eksi yönlerinden yola çıkılarak karma yöntemler geliştirilmektedir.

Kötücül yazılım tespiti yapılırken imza tabanlı yaklaşım sıkça tercih edilmektedir [7]. İmza tabanlı yaklaşımda bir imza veritabanı bulunmaktadır. Bu veritabanı aracılığıyla uygulamaların test edilmesi sağlanmaktadır. Öncelikle test edilecek uygulamanın bir imza şablonu oluşturulmaktadır. Daha sonra oluşturulan bu imza şablonu veritabanında yer alan kötücül yazılım şablonuna benziyorsa kötücül yazılım olarak etiketlenmektedir. Son yıllarda kötücül yazılım geliştiricileri uygulama kaynak kodlarına fazladan kod ekleme, kod şifreleme ve kod çıkarma gibi teknikler uygulayarak imza tabanlı kötücül yazılım sistemlerini yanıltmayı başarmaktadırlar [8]. Bu nedenle imza tabanlı yaklaşımlar yerine makine öğrenmesi tekniklerini kullanan kötücül yazılım tespit sistemleri ön plana çıkmaktadır. Uygulamanın çalışmasına ihtiyaç duyulmadan hızlı ve kolay sonuçlar alındığından, makine öğrenmesi tabanlı analiz sistemlerinde statik analiz sıkça kullanılmaktadır.

Makine öğrenmesi algoritmalarını kullanarak kötücül yazılım sistemi tasarlamak için öncelikle algoritmaya anlamlı girdi (özellik vektörü) verilmelidir. Bu sayede algoritma aldığı girdiler ile eğitim aşamasında çeşitli bilgileri öğrenecek, test aşamasında da yeni gelen girdilerin hangi türe ait olacağına karar verecektir. Algoritmanın performansı ve sistemin başarısı girdi ile doğrudan ilişkili olduğundan, karşılaşılan temel problem algoritmaya verilecek girdinin ne olacağıdır. Kötücül yazılım tespitinde, makine öğrenmesi algoritmalarına girdiler üretebilmek için birçok yöntem uygulanmaktadır. Bu yöntemlerden birisi de metin madenciliğidir.

Liao ve arkadaşları saldırı tespit sisteminin modellenmesinde metin madenciliği tekniklerini kullanmışlardır [9]. Önerilen yöntemde her bir sistem çağrısı metin sınıflandırmada kelimeye karşılık gelirken, sistem çağrıları dizisi de dokümana karşılık gelmektedir. Ayrıca her sistem çağrısı terim frekansı-ters doküman frekansı (TF-IDF) ile ağırlıklandırılarak KNN sınıflandırma algoritmasına verilmiştir.

İmran ve arkadaşları metin sınıflandırmada sıkça kullanılan dört farklı özellik çıkarma yöntemi ve bu yöntemlerin varyasyonlarını Windows işletim sistemleri üzerinde çalışan kötücül yazılımların tespiti için kullanmışlardır [10]. Çalışmada kullanılan yöntemler ikili, frekans tabanlı, TF-IDF ve gizli markov modeline dayalı özellik çıkarma yöntemleri ve türevleridir. Bu yöntemler ile uygulamaların sistem çağrı günlüklerinden (system call log) istatistiksel bilgiler toplanarak özellik vektörü oluşturulmuş ve rastgele orman algoritmasına girdi olarak verilmiştir. Çalışmada en iyi sonucu gizli markov modeline dayalı özellik çıkarma yöntemi vermiştir. Sınıflandırma sonucu F-ölçeğine göre 0,87 tespit edilmiştir.

Lin ve arkadaşları kötücül yazılımların davranışları hakkında bilgi elde edebilmek için sanal makineler üzerinde uygulamaları kurmuşlardır [11]. Metin sınıflandırmada sıkça kullanılan n-gram, TF-IDF ve kelime çantası (bag of words) yaklaşımları kötücül yazılımların tespitine uyarlanmıştır. Çalışmada, n-gram kullanılmasının sınıflandırma başarımına olumlu yönde etki ettiği görülmüştür. Buna karşın özellik sayısının artması nedeniyle sınıflandırma algoritmasının karar verme süresinin uzadığı vurgulanmıştır. Sınıflandırma algoritmasının daha etkin olabilmesi için iki aşamalı bir yapı önerilmiştir. İlk aşama TF-IDF ile ağırlıklandırma olurken ikinci aşama da temel bileşen analizi kullanılarak boyut indirgemesi olmuştur.

Şahin ve arkadaşları Android işletim sistemi için izin tabanlı kötücül yazılım tespiti yapmışlardır [12]. Literatürde yapılan çoğu izin tabanlı kötücül yazılım tespiti çalışmalarında izinlerin uygulama tarafından kullanılıp kullanılmamasına göre ikili ağırlıklandırma tercih edilmektedir [13, 14]. Şahin ve arkadaşları ikili ağırlıklandırma yerine metin sınıflandırmada iyi sonuçlar veren terim ağırlıklandırma yöntemini kötücül yazılım tespitinde kullanmışlardır. Uyarlanan terim ağırlıklandırma yönteminin ikili ağırlıklandırma yöntemine göre başarıyı arttırdığı vurgulanmıştır [12].

Suarez-Tangil ve arkadaşları DENDROID adını verdikleri sistemde, uygulamaların kaynak kodlarını metin madenciliği ve bilgi getirme teknikleriyle analiz ederek Android kötücül yazılımları türlerine ayrıştırmışlardır [15]. Bu çalışmanın kaynak kod analizinin metin madenciliği ile yapılarak mobil kötücül yazılım tespitinde kullanılması bakımından ilk çalışma olduğu vurgulanmıştır. İncelenen her uygulama bir dokuman olarak temsil edilirken, uygulamaya ait kodlarda kelime gibi temsil edilmektedir. Her uygulama ve kod bir vektöre dönüştürüldükten sonra analiz aşamasına geçilmiştir. Bu aşamada kötü amaçlı yazılım ailelerinin filogenetik ağaçlar olarak anlaşılabilmesi hiyerarşik kümeleme tekniği olan dendogram kullanılmıştır. Dendogram kullanılmasıyla kötücül yazılım aileleri arasındaki ortak özelliklerin çıkarılması hedeflenmiştir. Bu ortak özelliklere belirli kod blok veya satırların benzerlikleri örnek olarak verilebilir.

Düşün ve arkadaşları Android kötücül yazılım tespitinde tersine mühendislik tekniğini uygulayarak öncelikle uygulama kaynak kodlarından makine kodlarının elde edilmesini sağlamışlardır [8]. Makine kodlarının elde edilmesinden sonra belirlenen komut setlerine n-gram tekniği uygulanmıştır. Çalışmada 2-gram ve 3-gram dil modelleri tercih edilmiştir. n-gram tekniğinin ardından oluşturulan her öznitelik TF-IDF'den geçirilmiştir. N-gram ve TF-IDF modellerinden elde edilen vektör derin öğrenme ağı ve bazı klasik makine öğrenme algoritmalarına verilerek sınıflandırma başarımları kıyaslanmıştır. En başarılı sınıflandırma sonucu 0,9204 doğruluk ile derin öğrenme

ağından elde edilmiştir. Derin öğrenmenin ardından derin öğrenmeye yakın sonuçlar veren rastgele orman algoritması olmuştur.

Kötücül yazılım tespitinde makine öğrenmesi algoritmalarının sınıflandırma başarımını artırmak ve çalışma sürelerini azaltmak için çok sayıda çalışmanın yapıldığı görülmektedir. Bu çalışmada ise Android işletim sistemleri için makine öğrenmesi tabanlı statik analiz tekniği kullanılarak kötücül yazılımların tespiti yapılmıştır.

A. Motivasyon

Android kötücül yazılım tespitinde uygulamaların kaynak kodlarından elde edilen izinler kullanılmaktadır [13, 14]. İzin temelli çalışmaların çoğunda uygulamada izinlerin olup olmadığına göre bir başka ifadeyle uygulama içerisinde ilgili izin varsa 1, yoksa 0 şeklinde öznitelik vektörü oluşturulmaktadır [13, 14]. Bu ikili yapı metin sınıflandırma çalışmalarında da kullanılmaktadır [16]. Bazı metin sınıflandırma çalışmalarında ikili vektör yerine her kelime önem derecesine göre ağırlıklandırılarak kategorilerin daha kolay ayrıştırılması hedeflenmektedir [17]. Her bir izin farklı bir önem derecesine sahip olup sınıflandırma başarımında doğrudan etkili olduğundan, benzer bir yaklaşım izin tabanlı kötücül yazılım üzerinde de uygulanabilir. Literatürde bu alanda yapılan çalışmaların sınırlı olduğu görülmektedir. Bu çalışmada, Android kötücül yazılımların tespitinde yüksek başarım oranları elde etmek için metin sınıflandırmada sıkça kullanılan 14 farklı terim ağırlıklandırma tekniği kötücül yazılım tespitine uyarlanarak geniş kapsamlı bir çalışma yapılmıştır.

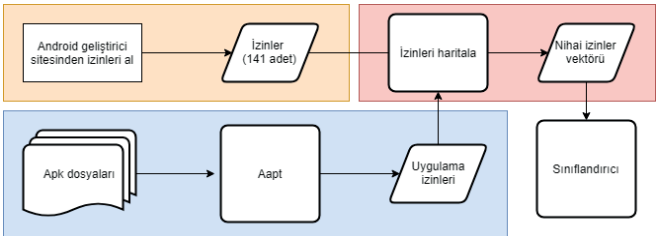
B. Katkı

Bölüm III'de anlatılacak olan ağırlıklandırma yöntemlerinden ikili ağırlıklandırma, TFIDF ve RF izin tabanlı kötücül yazılım tespitinde kullanılmışken [9, 12, 13, 14] GSS, DESCR, DISCR gibi bazı yöntemler izin tabanlı Android kötücül yazılım tespit sistemlerinde hiç denenmemiştir. Metin sınıflandırmada kullanılan bu terim ağırlıklandırma yöntemlerinin izin tabanlı kötücül yazılım tespitinde kullanılması çalışmanın özgünlüğünü oluşturmakta ve diğer izin tabanlı çalışmalara göre farklılaşma sağlamaktadır. Ayrıca metin sınıflandırmada sıkça kullanılan ağırlıklandırma metriklerinin kötücül yazılım tespitine uyarlanmasıyla sınıflandırma başarımı en yakın komşu algoritmasıyla birlikte arttırılmaktadır.

Çalışma şu şekilde organize edilmiştir: Bölüm II'de Android uygulama dosyalarının işlenmesi ve bilgilerin çıkarılmasına değinilmiştir. Bölüm III'de terim ağırlıklandırma yöntemlerinin matematiksel gösterimleri verilmiş, Bölüm IV'de çalışmada kullanılan veri kümeleri, sınıflandırma algoritmaları ve sınıflandırma sonuçlarının değerlendirilmesini sağlayan metrikler açıklanmıştır. Bölüm V'de elde edilen sonuçlar verilerek yorumlanmış, son olarak Bölüm VI'da ise gelecek çalışmalar hakkında bilgi verilmiştir.

II. VERİ ÖN İŞLEME

Bu bölümde Android uygulama dosyalarının nasıl işleneceğine ve bu işlem adımlarına değinilmektedir. Veri kümesinde yer alan APK dosyalarına sınıflandırma işlemlerinin uygulanabilmesi için öncelikle uygulama izinlerinin çıkartılarak işlenebilir bir veri dosyası haline getirilmesi gerekmektedir. Çalışma kapsamında belirlenen doğrultuda uygulamalar, mevcut durumda Android geliştirici sitesinde [24] yer alan 141 izin üzerinden değerlendirilmektedir. Öncelikle iyicil ve kötücül uygulamaların her biri için Android aapt (Android Asset Packaging Tool) aracı kullanılarak izinler elde edilmektedir. Elde edilen izinler geliştirici sitesinden elde edilen 141 izine haritalanarak her bir uygulama için nihai özellik vektörü oluşturulmaktadır. İşlemlere ait genel akış Şekil 1'de verilmektedir.



Şekil 1. Özellik vektörü oluşturma adımları

III. TERİM AĞIRLIKLANDIRMA YÖNTEMLERİ

Bu bölümde metin sınıflandırma çalışmalarında sıkça kullanılan terim ağırlıklandırma yöntemleri verilmektedir. Ayrıca bu yöntemlerin izinlerin ağırlıklandırılmasında nasıl kullanılacağına değinilmektedir.

İzinler ile uygulama dosyaları arasındaki ilişki bir matris şeklindedir. Bu matriste her sütun bir izne karşılık gelmekte, her bir satır ise uygulamadaki izinleri temsil etmektedir.

Tablo I'de metin sınıflandırmada kullanılan terim ağırlıklandırma yöntemleri verilmektedir.

TABLO I. AĞIRLIKLANDIRMA YÖNTEMLERİ ve GÖSTERİMLERİ

Yöntem	Formül
İkili ağırlıklandırma (binary)	İlgili izin uygulama tarafından kullanılıyorsa 1, kullanılmıyorsa 0 şeklinde ilgili izine skor ataması yapılmaktadır.
IDF [16]	$log\frac{N}{a+c}$
CHI [17, 19]	$N\frac{(a*d-b*c)^2}{(a+c)(b+d)(a+b)(c+d)}$
OR [17, 19]	$log(\frac{a*d}{b*c})$
RF [19]	$log(2+\frac{a}{c})$
MI [17, 19]	$log(N\frac{a}{(a+b)(a+c)})$
TGF [21]	$a+c$
IDFEC [20]	$log(\frac{c+d}{c})$
GSS [18]	$log(2+\frac{a+c+d}{c})$
IG [17, 19]	$\left(\frac{a}{N}\right)log\left(\frac{a}{a+c}\right)-\left(\frac{a+b}{N}\right)log\left(\frac{a+b}{N}\right)+\left(\frac{b}{N}\right)log\left(\frac{b}{b+d}\right)$
GR [17, 19]	$\frac{IG}{-\left(\frac{a+b}{N}\right)log\left(\frac{a+b}{N}\right)-\left(\frac{c+d}{N}\right)log\left(\frac{c+d}{N}\right)}$
DESCR [21]	$\frac{a}{a+b}$
DISCR [21]	$\frac{a}{a+c}$
FDD [21]	$\frac{(1+beta^2)*DISCR*DESCR}{beta^2*DISCR+DESCR}$ $beta=0{,}477$

Tablo I'de $a\ p_j$ izininin kaç tane kötücül uygulama tarafından istendiğinin sayısını, $b\ p_j$ izininin kaç tane kötücül uygulama tarafından istenmediğinin sayısını, $c\ p_j$ izininin kaç tane iyicil uygulama tarafından istendiğinin sayısını, $d\ p_j$ izininin kaç tane iyicil uygulama tarafından istenmediğinin sayısını göstermekte, N ise $N = a + b + c + d$ şeklinde hesaplanmaktadır.

Tablo I'de verilen formüllerde paydanın 0 olması veya logaritma fonksiyonunun tanımsız ($log(0)$) olması durumları ile zaman zaman karşılaşılmaktadır. Bunun için uygulama kısmında bazı değişiklikler yapılmalıdır. Örneğin RF yöntemi ele alınırsa c değerinin 0 olması durumunda bu değerin yerine 1 değeri kullanılmaktadır. Bu diğer formüller içinde geçerlidir.

Metin sınıflandırmada, Tablo I'de verilen ağırlıklandırma yöntemleri ile her kelimeye genel bir ağırlıklandırma değeri atanmaktadır. Ardından her kelimenin ağırlık değeri dokümanda geçen kelimenin frekansı (terim frekansı) ile çarpılarak elde edilen değer özellik vektörüne eklenmektedir. Bu durum kötücül yazılım tespitinde farklı bir yapıdadır. Çünkü bir uygulama bir izini ya kullanır ya da kullanmaz. Bu nedenle terim frekansı, ilgili izin uygulama tarafından isteniyorsa 1, istenmiyorsa 0 olmaktadır. Tablo I'deki formüllerden elde edilen genel izin ağırlıkları ile terim frekansı çarpılarak özellik vektörüne eklenmektedir. Bu adımlar eğitim aşamasındaki tüm uygulamalara ve izinlere uygulanıp özellik vektörü oluşturulmaktadır. Elde edilen özellik vektörü sınıflandırma algoritmasına girdi olarak verilerek sınıflandırıcı eğitilmektedir. Test edilecek uygulamalara ait izinlerin ağırlık değerlerine eğitim aşamasında elde edilen genel ağırlık değerleri atanmaktadır. Bunun sebebi test edilecek uygulamanın sınıfı bilinmediği için $a, b, c\ ve\ d$ değerlerinin elde edilemeyecek olmasından kaynaklanmaktadır.

IV. DENEYSEL AYARLAMALAR

Bu bölümde, çalışmada kullanılan veri kümesi, makine öğrenme algoritmaları ve performans ölçüm metrikleri verilmektedir.

A. Kullanılan Veri Kümeleri

Bu çalışmada iki farklı veri kümesi kullanılarak yöntemler kıyaslanmaktadır. İlk veri kümesinde (M0DROID) 200 kötücül ve 200 iyicil Android uygulama yer almaktadır [22]. Kullanılan ikinci veri kümesi AMD'dir (Android Malware Dataset). Bu veri kümesinde 24 binden fazla Android kötücül yazılım vardır [23]. Bu veri kümesinde rastgele alınan 2000 uygulama ile Play Store'dan indirilen 1000 iyicil uygulama kullanılmaktadır. İyicil uygulamalar yüksek puanlı olup, resmi Android uygulama merkezi olan Google Play Store'den sağlanmaktadır.

B. Kullanılan Sınıflandırma Algoritmaları

Çalışma prensipleri birbirlerinden farklı olan 3 sınıflandırma algoritması bu çalışmada kullanıldı. Bunlar doğrusal ayırma ilkesine göre çalışan destek vektör makinesi (SVM), olasılık ilkesine göre çalışan naive bayes (NB) ve örnek tabanlı en yakın komşu algoritmalarıdır. Çalışmada kullanılan SVM doğrusal çekirdek fonksiyonu, NB algoritması çok terimli, KNN algoritmasında k değeri 5 seçilmiştir.

C. Performans Ölçeği

Bu çalışmada sınıflandırma algoritmalarının başarımı ölçmek için iki farklı başarım metriği kullanılmaktadır. Bunlardan birincisi doğruluk (Accuracy) olurken, ikincisi F-ölçütüdür (F-measure). Eşitlik 1'de doğruluk hesabı verilmektedir.

$$Doğruluk = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

Gerçekte kötücül olan bir uygulama kötücül olarak sınıflandırılmışsa TP (True Positive - Doğru Pozitif), gerçekte kötücül olan bir uygulama iyicil olarak sınıflandırılmışsa FN (False Negative - Yanlış Negatif) gerçekte iyicil olan bir uygulama iyicil olarak sınıflandırılmışsa TN (True Negative - Doğru Negatif) ve gerçekte iyicil olan bir uygulama kötücül olarak sınıflandırılmışsa FP (False Positive - Yanlış Pozitif) olarak adlandırılmaktadır.

F-ölçütü iki bileşenden meydana gelmektedir. Bunlar kesinlik (precision) ve duyarlılık (recall) değerleridir. Eşitlik 2 ve 3'de sırasıyla bu değerlerin gösterimi verilmektedir.

$$Kesinlik = \frac{TP}{TP+FP} \quad (2)$$

$$Duyarlılık = \frac{TP}{TP+FP} \quad (3)$$

F-ölçütü de kesinlik ve duyarlılık değerlerinin harmonik ortalamasıdır ve Eşitlik 4'de gösterilmektedir.

$$F-ölçüt = \frac{2*Kesinlik*Duyarlılık}{Kesinlik*Duyarlılık} \quad (4)$$

V. SONUÇ VE TARTIŞMA

Bu bölümde, iki farklı veri kümesinden elde edilen sonuçlar verilecek ve yorumlanacaktır. Çalışmada danışmanlı öğrenme algoritmaları kullanıldığından algoritmaların eğitim aşamasına ihtiyaç duyulmaktadır. Tüm veri kümesinin %70'i eğitim ve %30'u test için kullanılmaktadır. M0DROID ve AMD veri kümeleri içerisinde yer alan uygulamalar Tablo I'de verilen 14 farklı ağırlıklandırma tekniğinden geçirilerek farklı özellik vektörleri oluşturulmuştur. Oluşturulan her özellik vektörü KNN, NB ve SVM sınıflandırma algoritmalarına girdi olarak verilmekte ve bu algoritmalardan test sonuçları elde edilmektedir. Her ağırlıklandırma yöntemi ve sınıflandırma algoritmasının ortalama doğruluk ve F-ölçütü sonuçları tablolar halinde verilmektedir.

Tablo II'de M0DROID veri kümesi kullanılarak elde edilen sonuçlar verilmektedir. Tablo II incelendiğinde en başarılı sınıflandırıcı SVM algoritmasıdır. SVM'nin en başarılı olduğu ağırlıklandırma yöntemi DISCR metriğidir. En yüksek başarımın elde edildiği değerler doğruluk için 0,8739 olurken F-ölçüt için 0,8823 olmaktadır. SVM algoritmasından elde edilen en başarılı diğer ağırlıklandırma teknikleri ikili ağırlıklandırma ve FDD olmaktadır. SVM üzerinde en kötü ağırlıklandırma teknikleri IG ve GR olmakla birlikte diğer metriklerin oldukça gerisinde kalmaktadırlar. Bunlar haricinde diğer ağırlıklandırma metriklerinin birbirlerine yakın sonuçlar verdiği gözlemlenmektedir.

KNN algoritmasıyla elde edilen en başarılı sonuç IDF ağırlıklandırma yöntemiyle elde edilmektedir. Bu sonuçlar doğruluk için 0,8546 olurken F-ölçüt için 0,85 olmaktadır. IDF'nin ardından IDF'nin başka bir türü olan IDFEC yöntemi en başarılı yöntem olmaktadır. KNN algoritmasıyla birlikte IDF veya IDFEC gibi bir ağırlıklandırma kullanılması durumunda ikili ağırlıklandırma yöntemine göre %5'den fazla bir iyileştirme elde edilmektedir.

NB algoritmasıyla elde edilen en başarılı sonuç TGF yöntemiyle elde edilmektedir. Bu sonuçlar doğruluk için 0,7910 olurken F-ölçüt için 0,8041 olmaktadır. NB algoritmasıyla elde edilen sonuçların SVM ve KNN algoritmalarına göre oldukça kötü sonuçlar verdiği görülmektedir.

Tablo III'de AMD veri kümesi kullanılarak elde edilen sonuçlar verilmektedir. Tablo III incelendiğinde en başarılı sınıflandırma SVM algoritmasıdır. SVM'nin en başarılı olduğu ağırlıklandırma yöntemi ikili ağırlıklandırma olmaktadır. En yüksek başarımın elde edildiği değerler doğruluk için 0,9632 olurken F-ölçüt için 0,9484 olmaktadır. SVM algoritmasından elde edilen en başarılı diğer Ağırlıklandırma teknikleri MI ve RF olmaktadır. SVM üzerinde en kötü ağırlıklandırma tekniğinin TGF olduğu gözlemlenmektedir. TGF haricindeki tüm ağırlıklandırma yöntemlerinden doğruluk metriğine göre %90'ın üzerinde başarı elde edilmekteyken TGF'de bu oran %83 olarak tespit edilmektedir. Bunlar haricinde diğer ağırlıklandırma yöntemlerinin birbirlerine yakın sonuçlar verdiği gözlemlenmektedir.

KNN algoritmasıyla elde edilen en başarılı sonuç OR ağırlıklandırma yöntemiyle elde edilmektedir. Bu sonuçlar doğruluk için 0,9591 olurken F-ölçüt için 0,9399 olmaktadır. OR'nin ardından GR ve IG en başarılı yöntem olmaktadır.

NB algoritmasıyla elde edilen en başarılı sonuç FDD yöntemiyle elde edilmektedir. Bu sonuçlar doğruluk için 0,9472 olurken F-ölçüt için 0,9230 olmaktadır. AMD veri kümesi üzerinde NB algoritmasıyla yapılan deneylerde bütün ağırlıklandırma yöntemlerinin birbirlerine yakın ve iyi sonuçlar verdiği görülmektedir. Çok terimli NB algoritması dağılımlar üzerinden çalışmaktadır. AMD veri kümesine 14 farklı terim ağırlıklandırma modeli eklendiğinde her bir yönteminin dağılımları benzer sonuçlar verdiği için yöntemlerin başarım sonuçları da benzer çıkmaktadır.

TABLO II. M0DROID VERİ KÜMESİ İLE ELDE EDİLEN SONUÇLAR

	KNN		NB		SVM	
Yöntem	**Doğruluk**	**F-ölçüt**	**Doğruluk**	**F-ölçüt**	**Doğruluk**	**F-ölçüt**
BINARY	0,8	0,7808	0,7814	0,7915	0,8680	0,872
IDF	**0,8546**	**0,85**	0,7596	0,7673	0,8310	0,8234
CHI	0,7932	0,7830	0,7807	0,7909	0,8235	0,8367
OR	0,8226	0,8158	0,7709	0,7882	0,8151	0,8325
RF	0,8319	0,8268	0,7857	0,7960	0,8437	0,8455
MI	0,8445	0,8463	0,7746	0,7874	0,8201	0,8429
TGF	0,8084	0,8079	**0,7910**	**0,8041**	0,8294	0,8291
IDFEC	0,8512	0,8459	0,7864	0,7949	0,8403	0,8384
GSS	0,8445	0,8432	0,7779	0,7882	0,8386	0,8416
IG	0,7798	0,7708	0,7892	0,8069	0,7336	0,7835
GR	0,7857	0,7792	0,7854	0,8068	0,7403	0,7872
DESCR	0,7815	0,7866	0,7508	0,7813	0,8193	0,8233
DISCR	0,8395	0,8348	0,7794	0,7943	**0,8739**	**0,8823**
FDD	0,8008	0,7996	0,7839	0,7950	0,8563	0,8612

TABLO III. AMD VERİ ERİ KÜMESİ İLE ELDE EDİLEN SONUÇLAR

	KNN		NB		SVM	
Yöntem	**Doğruluk**	**F-ölçüt**	**Doğruluk**	**F-ölçüt**	**Doğruluk**	**F-ölçüt**
BINARY	0,9315	0,8925	0,9421	0,9155	**0,9632**	**0,9484**
IDF	0,9300	0,897	0,9469	0,9219	0,9585	0,9416
CHI	0,9488	0,9239	0,9459	0,9213	0,8889	0,8491
OR	**0,9591**	**0,9399**	0,9452	0,9201	0,9600	0,9436
RF	0,9327	0,8977	0,9446	0,9198	0,9611	0,9455
MI	0,9412	0,9149	0,9451	0,9198	0,9628	0,9475
TGF	0,9429	0,9175	0,9468	0,9222	0,8335	0,7880
IDFEC	0,9222	0,8810	0,9453	0,9204	0,9576	0,9399
GSS	0,9336	0,8984	0,9423	0,9158	0,9572	0,9392
IG	0,9488	0,9237	0,9456	0,9206	0,9199	0,8894
GR	0,9511	0,9271	0,9450	0,9201	0,9172	0,8855
DESCR	0,9365	0,9084	0,9474	0,9249	0,9235	0,8889
DISCR	0,9214	0,8808	0,9453	0,9202	0,9547	0,9360
FDD	0,9326	0,9027	**0,9472**	**0,9230**	0,9545	0,9362

VI. BULGULAR VE GELECEK ÇALIŞMALAR

Bu çalışmada metin sınıflandırma çalışmalarında kullanılan terim ağırlıklandırma teknikleri izin tabanlı Android kötücül yazılım tespitine uygulanmıştır. Bu sayede her izine ait önem skoru oluşturulmuştur. Uyarlanan yöntemlerin veri kümelerine göre başarımları farklılık gösterse de genel olarak sınıflandırma başarımını olumlu yönde etkilediği görülmüştür. Ayrıca metin sınıflandırmada kelimelere ağırlıklar verildiği gibi izin tabanlı yaklaşımlarda, izinlere ağırlık atamasının nasıl yapılacağının modellenmesi detaylıca anlatılmıştır. İzin tabanlı statik analiz yaklaşımında, ikili ağırlıklandırma yönteminin yerine kullanılabilecek olan çoklu ağırlıklandırma tekniğinin uygulanabilirliği gösterilmiştir. Gelecek çalışmalarda veri kümesi genişletilerek güncel uygulamalar eklenecek ve yeni ağırlıklandırma tekniklerinin oluşturulması ile daha kapsamlı analizler yapılacaktır.

KAYNAKLAR

[1] https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/ Son erişim tarihi: 08.04.2019

[2] https://venturebeat.com/2018/09/11/newzoo-smartphone-users-will-top-3-billion-in-2018-hit-3-8-billion-by-2021/ Son erişim tarihi: 08.04.2019

[3] https://www.gartner.com/en/newsroom/press-releases/2018-08-28-gartner-says-huawei-secured-no-2-worldwide-smartphone-vendor-spot-surpassing-apple-in-second-quarter Son erişim tarihi: 09.04.2019

[4] https://www.statista.com/statistics/680705/global-android-malware-volume/ Son erişim tarihi: 09.04.2019

[5] N. Idika, A. P. Mathur, "A Survey of Malware Detection Techniques," Purdue University, 2007.

[6] P. Faruki et al, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," in IEEE Communications Surveys & Tutorials, 17(2):998-1022, 2015.

[7] M. Zheng, M. Sun and J. C. S. Lui, "Droid Analytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware," 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, VIC, 2013, pp. 163-171.

[8] B. Düşün, I. Bulut, R. C. Aygün, A. G. Yavuz, "Combat mobile malware via N-gram based deep learning," 2018 IEEE 26th Signal Processing and Communications Applications Conference (SIU), Izmir, 2018, pp. 1-4.

[9] Y. Liao, V. R. Vemuri, "Using text categorization techniques for intrusion detection," 2002 USENIX Security Symposium, 12, 51-59, 2002.

[10] M. Imran, M. T Afzal, M. A. Qadir, "A comparison of feature extraction techniques for malware analysis," Turkish Journal of Electrical Engineering & Computer Sciences, 25(2):1173-1183, 2017.

[11] Lin, C. T., Wang, N. J., Xiao, H., & Eckert, C., "Feature Selection and Extraction for Malware Classification," Journal Of Informatıon Scıence And Engıneerıng, 31(3):965-992, 2015.

[12] D. Ö. Şahın, O. E. Kural, S. Akleylek, E. Kiliç, "New results on permission based static analysis for Android malware," 2018 IEEE 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, 2018, pp. 1-4.

[13] X. Liu, J. Liu, "A Two-Layered Permission-Based Android Malware Detection Scheme," 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Oxford, 2014, pp. 142-148.

[14] A. Aydın, İ. A. Doğru, M. Dörterler, "Makine Öğrenmesi Algoritmalarıyla Android Kötücül Yazılım Uygulamalarının Tespiti," Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 22(2):1087-1094, 2018.

[15] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, J. Blasco, "Dendroid: A text mining approach to analyzing and classifying code structures in android malware families," Expert Systems with Applications, 41(4):1104-1117, 2014.

[16] G. Salton, C. Buckley, "Term-weighting approaches in automatic text retrieval," Information processing & management, 24(5):513-523, 1988.

[17] F. Debole, F. Sebastiani, "Supervised Term Weighting for Automated Text Categorization," Text Mining and its Applications. Studies in Fuzziness and Soft Computing, 138:81-97, 2004.

[18] L. Galavotti, F. Sebastiani, M. Simi, "Experiments on the use of feature selection and negative evidence in automated text categorization," In International Conference on Theory and Practice of Digital Libraries, 2000, pp. 59-68.

[19] M. Lan, C. L. Tan, J. Su and Y. Lu, "Supervised and Traditional Term Weighting Methods for Automatic Text Categorization," IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(4):721-735, 2009.

[20] G. Domeniconi, G. Moro, R. Pasolini, C. Sartori, "A Study on Term Weighting for Text Categorization: A Novel Supervised Variant of tf. İdf," 4th International Conference on Data Management Technologies and Applications, 2015, pp. 26-37.

[21] M. Maisonnave, F. Delbianco, F.A. Tohmé, A. G. Maguitman, "A Flexible Supervised Term-Weighting Technique and its Application to Variable Extraction and Information Retrieval," Inteligencia Artificial, 22(63):61-80, 2019.

[22] M. Damshenas, A. Dehghantanha, K. K. R. Choo, R. Mahmud, "M0droid: An android behavioral-based malware detection model," Journal of Information Privacy and Security, 11(3):141-157, 2015.

[23] F. Wei, Y. Li, S. Roy, X. Ou, W. Zhou, "Deep ground truth analysis of current android malware," International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 2017, pp. 252-276.

[24] https://developer.android.com/reference/android/Manifest.permission.html Son erişim tarihi: 11.04.2019