Düzce Üniversitesi Bilim ve Teknoloji Dergisi

Araştırma Makalesi

Kötü Amaçlı Yazılım Tespiti için Makine Öğrenmesi Algoritmalarının Kullanımı

Pınar TÜFEKCİ [a,*], Çetin Mutlu ÖNAL [a]

[a] *Bilgisayar Mühendisliği Bölümü, Çorlu Mühendislik Fakültesi, Tekirdağ Namık Kemal Üniversitesi, Tekirdağ, TÜRKİYE*

** Sorumlu yazarın e-posta adresi: ptufekci@nku.edu.tr*

DOI: 10.29130/dubited.1287453

ÖZET

Gelişen teknoloji sayesinde bilgiye kolay erişim sağlansa da, bu durum kötü amaçlı eylemlerin artışına da sebep olmuştur. Android işletim sistemlerinde sıklıkla rastlanan kötü amaçlı yazılımlar (malware), kullanıcıların cihazındaki verilere erişerek büyük bir tehdit oluşturmaktadır. Bu çalışma, kötü amaçlı yazılımları tespit etmek amacıyla yüksek doğruluklu ve güvenilir bir model geliştirmeyi hedeflemektedir. Modelleme çalışmalarında popüler bir veri seti olan DREBIN-215 Android Malware Dataset kullanılmıştır. Makine Öğrenmesi algoritmaları arasından Support Vector Machines (SVM), Gradient Boosting (GB), Multi Layer Perceptron (MLP), Naïve Bayes (MNB), K-En Yakın Komşu (KNN) ve Random Forest (RF) algoritmaları uygulanmıştır. Algoritmaların performansları, varsayılan parametreler ve GridSearch yöntemiyle elde edilen en iyi hiperparametre değerlerinin kullanılmasıyla değerlendirilmiştir. En başarılı model, SVM algoritmasıyla en iyi hiperparametrelerin uygulanması sonucu %99.07 doğruluk oranıyla elde edilmiştir.

Anahtar Kelimeler: *Kötü amaçlı yazılım, Makine öğrenmesi, Support Vector Machines, Gradient Boosting, Multi Layer Perceptron.*

Using Machine Learning Algorithms for Malware Detection

ABSTRACT

Although advanced technology has facilitated easy access to information, it has also led to an increase in malicious activities. Malware, frequently encountered in Android operating systems, poses a significant threat by accessing users' data on their devices. This study aims to develop a highly accurate and reliable model for detecting malware. The modeling work used the popular and imbalanced dataset, DREBIN-215 Android Malware Dataset. Machine Learning algorithms such as Support Vector Machines (SVM), Gradient Boosting (GB), Multi Layer Perceptron (MLP), Naïve Bayes (MNB), K-Nearest Neighbor (KNN) and Random Forest (RF) were applied. The performance of the algorithms was evaluated using default parameters and the best hyperparameter values obtained through the GridSearch method. The most successful model was achieved with an accuracy rate of 99.07% by applying the best hyperparameters in the SVM algorithm.

Keywords: *Malware, Machine learning, Support Vector Machines, Gradient Boosting, Multi Layer Perceptron.*

Geliş: 25/04/2023, Düzeltme: 11/10/2023, Kabul: 24/10/2023

I. GİRİŞ

Malware, "Zararlı Yazılım" anlamına gelen İngilizce "Malicious Software" kelimelerinin kısaltması olup, bilgisayarlar ve akıllı telefonlar gibi teknolojik cihazlardan hassas bilgilere erişmeyi amaçlayan kötü niyetli programlardır. Android işletim sistemi, mobil cihazlarda yaygın bir şekilde kullanılmakta ve Android cihaz sayısı arttıkça, bu işletim sistemi için hedeflenen kötü amaçlı yazılımların sayısı da artmaktadır. Android cihazlar üzerindeki kötü amaçlı yazılımlar, donanım birimine, yerel dosya erişimine ve işletim sistemindeki okuma/yazma iznine izinsiz erişerek hassas verilere erişebilir [1].

Kötü amaçlı yazılımları tespit etmek için imza tabanlı [2, 3], makine öğrenimi tabanlı [4] ve derin öğrenme tabanlı [5, 6] yöntemler gibi çeşitli yöntemler önerilmiştir. Derin öğrenme algoritmalarının son zamanlarda kötü amaçlı yazılımları tespit etmede umut verici sonuçlar verdiği gözlemlenmiştir, çünkü kötü amaçlı yazılımın evrimine adapte olabilme özelliği vardır [5, 6].

Android uygulamalarından özellik çıkarmak, kötü amaçlı yazılım tespitinin performansını artırmak için önemlidir ve bu amaçla iki ortak analiz yöntemi kullanılmaktadır: Statik Analiz [6, 7, 8] ve Dinamik Analiz [9, 10]. Statik Analiz, uygulamanın çalıştırılmadan önce Android uygulama ve kaynak kodu dosyalarından çıkarılan özelliklere dayanarak uygulamaları sınıflandırır. Örneğin, AndroPyTool [7], AndroidManifest.xml'deki izinlerden, niyetlerden, hizmetlerden ve sağlayıcılardan ve Smali kodundan API çağrılarından özellikler çıkarır. Dinamik Analiz, şüpheli uygulamaları izole bir ortamda çalıştırılarak sisteme veya ağ trafiği verilerine odaklanır ve bu verileri buna göre sınıflandırır. Statik Analizin avantajı, uygulamanın cihaza yüklenmeden önce kötü amaçlı yazılımın tespit edilebilmesidir, Dinamik Analizin avantajı ise uygulama çalıştırma sürecinde ortaya çıkan açıklıkların tespit edilebilmesidir [1].

Son zamanlarda, literatürde birçok çalışma, Statik Analizde grafiklerden semantik özellikler elde etmek için daha fazla odaklanmışlardır [6, 11, 12]. Bu, semantik özelliklerin, kötü amaçlı yazılımın kaçınması için sözdizimsel özelliklere göre daha sağlam olarak kabul edildiği için yapılmaktadır [13]. Bir Fonksiyon Çağrı Grafiği (FCG), bir APK'daki yöntemler arasındaki çağırıcı-çağrılan ilişkileri yakalar, bu da yöntemler arasındaki ilişkiler hakkında bir içgörü sağlayabilir. Grafik yapısı, makine öğrenme algoritmalarına daha derin bir şekilde gömülebilir ve son zamanlarda Grafik Sinir Ağları (GNN), FCG'leri gömme için kullanılmıştır [14]. GNN, topolojik yapı ve düğüm özelliklerini değerlendirerek her düğüm için bilgilendirici gömme oluşturmak için kullanılabilir [15].

Literatürde, Android malware tespiti için makine öğrenmesi algoritmalarını kullanan birçok çalışma bulunmaktadır. Rana ve arkadaşları DREBIN veri seti (123453'te gerçekleşen 11120 Android uygulamasından oluşan) üzerinde ağaç tabanlı makine öğrenimi algoritmaları kullanarak modeller geliştirmişlerdir. Veri seti, 5560 kötü amaçlı yazılım örneği içermektedir ve iki sınıfa eşit olarak bölünmüştür. Bu çalışmada, Karar Ağaçları (DT), RF ve GB algoritmaları ile geliştirilen modeller sırasıyla %96.13, %97.24 ve %93.68 doğruluk değerleri elde etmiştir [16]. Peng ve arkadaşları tarafından yapılan bir çalışmada, Android sistemlerindeki zararlı kodlarla ilgili olarak, veri iletimi ve haberleşme sırasındaki riskleri analiz etme ve ölçme amaçlanmıştır. Bu çalışmada, olasılık tabanlı Naive Bayes algoritması kullanılarak %93.5 doğruluk başarısı elde edilmiştir [17].

Arp ve arkadaşları ise, çalışmalarında SVM algoritmasını kullanarak malware yazılım tespiti yapmaya çalışmışlardır. Bu çalışmada 131611 örnek içeren DREBIN veri seti ham haliyle kullanılmıştır. Veri setindeki malware olarak sınıflandırılan örnek sayısı 5560, dengesiz olarak iki sınıfa ayrılmıştır. SVM algoritması ile geliştirilen model ile %94 doğruluk başarısına ulaşıldığı belirtilmiştir [18]. Li ve arkadaşları tarafından yapılan çalışmada, makine öğrenmesi yöntemleri kullanarak malware tespitinde izinler incelenmiştir. Malware olan ve olmayan uygulamaları ayırmak için, veri setinde bulunan tüm izinler yerine sınıflandırmada en etkili olan izinler belirlenmiştir. Belirlenen izinlere uygulanan modeller sonucunda, SVM ile %91.22 ve Fonksiyonel Ağaç ile %93.62 sınıflandırma başarılarına ulaşılmıştır [19].

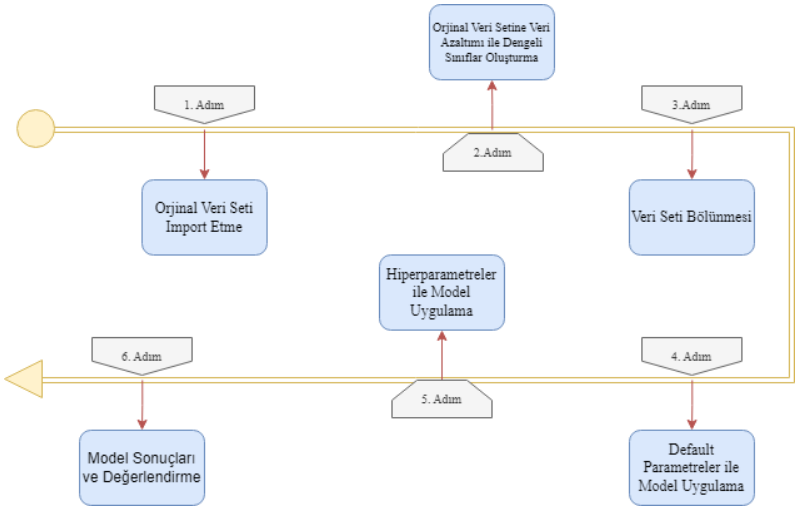
Qiao ve arkadaşlarının çalışmasında, veri setindeki 104 izin ve 654 API'den oluşan öznitelikler ikili ve sayısal tabanlı öznitelik çıkarımına tabi tutularak, RF, Yapay Sinir Ağları (ANN) ve SVM modelleri uygulanmıştır. En başarılı model, RF ile %94.41 olarak elde edilmiştir [20]. Aydın ve arkadaşları, çalışmalarında malware tespitinde izin tabanlı bir yöntem kullanmış ve yöntemin üzerinde performansı ölçülen SVM, NB, RF ve KNN modelleri arasından en yüksek doğruluk başarısına %95.65 ile RF modelinde ulaşılmıştır [21]. Güngör ve arkadaşları tarafından yapılan çalışmada, Malimg veri setinde bulunan APK dosyaları gri ölçekli görüntüye çevirerek öznitelik çıkarımı yapılmış ve sınıflandırma performansları bu şekilde elde edilmiştir. Bu çalışmada uygulanan KNN ve RF algoritmalarından sırasıyla %96.72 ve %97.44 doğruluk başarıları elde edilmiştir [22].Utku ve arkadaşlarının çalışmasında ise, Android cihazlarda malware tespitinde Karar Ağacı (DT) tabanlı tespit sistemi üzerinde, C4.5 ve Hoeffding ağacı algoritmaları ile modeller oluşturulmuştur. C4.5 karar ağacı algoritması ile %95.86 ve Hoeffding ağacı algoritması ile de %93.18 doğruluk başarısına ulaşılmıştır [23].

Liu ve arkadaşları da çalışmalarında, Android kötü amaçlı yazılımlarını tespit etmek için SeGDroid adı verilen yeni bir yöntem önermişlerdir. Bu yöntem, duyarlı işlev çağrısı grafiklerinden (FCG) semantik bilgi öğrenmeye odaklanmaktadır. SeGDroid, güvenlikle ilgili API çağrısı bağlamını koruyan ve gereksiz düğümleri çıkaran grafik budama kullanarak duyarlı bir FCG inşa etmektedir. SeGDroid, kötü amaçlı yazılım tespiti için bir grafik evrişimli sinir ağı algoritması kullanarak grafik gömütlemeleri indüklemekte olup, CICMal2020 veri kümesinde % 98 ve MalRadar veri setinde kötü amaçlı yazılım ailesi sınıflandırması için % 96 F-score değerine ulaşmıştır. Ayrıca, kötü amaçlı davranışları açıklamak için bir düğüm önemi hesaplama yöntemi önerilmiştir [24]. Yang ve arkadaşları, mevcut bilgilerin etkisini azaltmaya ve etiketlerin katılımı olmadan modeli önceden eğitmeye yönelik kontrastif öğrenmeye dayanan bir çalışma sunmuşlardır. Yapılan deneysel çalışmalar sonucunda, bu yöntemin halka açık veri setleri üzerinde zararlı yazılım tanımlama için %96'nın üzerinde ve çoklu sınıf tespiti (zararlı yazılım sınıfı ve ailesi) için %98'in üzerinde doğruluk oranına ulaşabildiği gözlemlenmiştir [25].

Bu çalışmada ise, bir uygulamanın malware olup olmadığını tespit edebilen yüksek doğruluklu ve güvenilir bir model oluşturulmaya çalışılmıştır. Bu amaçla Support Vector Machines (SVM), Gradient Boosting (GP) ve Multi Layer Perceptron (MLP), Naïve Bayes (MNB), K-En Yakın Komşu (KNN) ve Random Forest (RF) gibi makine öğrenmesi algoritmaları kullanılarak modelleme çalışmaları yapılmıştır. Modelleme sürecinde kullanılan veri seti (DREBIN-215 Android Malware Dataset) popüler bir veri seti olup, 2 sınıflı ve her sınıfta farklı sayıda örnek bulunan dengesiz bir veri setidir. Bu veri setine rastgele örnekleme uygulanarak veri azaltımı yapılmış ve her sınıfta eşit sayıda örnek içeren dengeli 3 yeni veri seti oluşturulmuştur. Yeni oluşturulan veri setleri, 70-30 oranında eğitim ve test veri seti olarak bölünerek algoritmalara uygulanmıştır. Algoritmaların varsayılan parametreleri ve GridSearch sonucu elde edilen en iyi hiper parametreleri ile model performansları karşılaştırılmış ve en iyi performansa sahip olan model bulunmaya çalışılmıştır. Sonuç olarak, en yüksek başarı, SVM modelinden, hiper parametre optimizasyonu ile %99.07 doğruluk başarısı ile elde edilmiştir.

II. MATERYAL VE METOT

Bu çalışmada Android'de kötü amaçlı yazılım tespiti için gerçekleştirilen modelleme süreci Şekil 1'de gösterilmiştir ve süreçte bulunan adımlar bu bölümde açıklanmıştır.



Şekil 1 *Modelleme süreci.*

Bu çalışma, Google Colab ortamında Python pandas, numpy, seaborn ve sklearn kütüphaneleri kullanılarak gerçekleştirilmiştir. Python'da her işlem için kullanılan kütüphaneler Tablo 1'de belirtilmiştir.

Tablo 1. *Kullanılan Kütüphaneler*

Kullanılan Kütüphane	Amaç	Kullanılan Aşama
Drive	Google Colab Drive Erişimi	Veri Seti Okunması ve İşlenmesi
Numpy	Matris İşlemleri	
Pandas	Veri Yapısı	
Seaborn	Veri Görselleştirme	
Random	Veri İşlemleri	
Sklearn	Sınıflandırma Algoritmaları, Veri Seti Bölümleme, Veri İşlemleri	Modelleme, Veri Bölümleme, Veri Önişleme
Matplotlib.pyplot	Veri Görselleştirme	Model Sonuçları

A. 1. Veri Seti Tanımı

Bu çalışma halka açık olarak paylaşılan DREBIN veri seti kullanılarak yapılmıştır [18]. 2010-2012 yılları arasında 131611 gerçek Android uygulamasının verileriyle oluşturulan bu veri setinin güncel olarak erişilebilen ve bu çalışmada kullanılan versiyonunda 2 sınıfa ait toplamda 15036 tane örnek bulunmaktadır. Bu örnekler Android uygulamaların manifest.xml dosyası ve gömülü kaynak kodlarından elde edilen 216 tane özniteliğe sahiptir. Bu iki kaynaktan elde edilen öznitelikler, kaynaklara ait 4'er ana grup olmak üzere 8 grupta sınıflandırılmaktadır.

A.1.1. Manifest.xml Dosyasından Elde Edilen Öznitelik Grupları

Android işletim sisteminde bulunan uygulamalar, Manifest.xml adında bir manifest dosyası içerir [26]. Ve bu dosyada, uygulamayla ilgili temel bilgiler ve veriler bulunur. Manifest.xml dosyasından elde edilen öznitelikler şunlardır:

- **Donanım Bileşenleri:** Kamera, GPS gibi erişim istenen donanım bileşenlerini belirtir.
- **Erişim İstenen İzinler:** Yazılım üzerinde yürütülen bilgilere ve eylemlere erişim iznini belirtir.
- **Uygulama Bileşenleri:** Servisler, İçerik sağlayıcıları, yayın alıcıları gibi uygulama bileşenlerini belirtir.
- **Filtrelenmiş Amaçlar:** Android iç süreçleri tarafından yürütülen amaçları belirtir.

A.1.2. Gömülü Kaynak Kodlarından (.DEX) Elde Edilen Öznitelik Grupları

Android işletim sistemi üzerinde bulunan uygulamaların kaynak kodlarında bulunan erişim isteklerinden ve çağrılardan elde edilen öznitelikler gömülü kaynak kodlarından elde edilen özniteliklerdir. Bu özniteliklerin sınıflandırılması şu şekildedir:

- **Kısıtlanmış API Çağrıları:** Manifest.xml dosyasında belirtilmeyen izinler kullanılarak API isteği gönderilmesini belirtir.
- **Kullanılan İzinler:** Bu özellik sayesinde API çağrısı için istenen iznin kötü amaçlı olup olmadığı anlaşılabilir.
- **Şüpheli API Çağrıları:** Cihaz bilgileri gibi hassas verileri isteyen API çağrılarını belirtir.
- **Ağ Adresleri:** Kötü amaçlı yazılımlar tarafından harici komut çalıştırma veya veri aktarma için kullanılan ağ adreslerini veya URL'leri belirtir [16, 27].

A.2.Veri Önişleme

Model geliştirme sürecinde kullanılacak olan veri setine karar verme aşamasında, Drebin (Orijinal) veri setinin dengesiz bir veri seti olduğu görülmüştür. Bu veri seti, 2 sınıflı (“Malware” ve “Malware Değil”) olup, “Malware” sınıfı 5560 örnek, “Malware Değil” sınıfı ise 123453 örnek içermektedir. Bu çalışmada kullanılan ve orijinal veri setinin güncel olarak bulunabilen versiyonu (D0) ise toplam 15036 örnek, 5560 tanesinin Malware ve kalan 9476 tanesinin ise Malware değil olarak yine dengesiz bir dağılıma sahiptir. Oluşturulacak modelin güvenilir ve başarı değerlerinin tutarlı olması istendiğinden bu veri seti üzerinde veri dengeleme yapılması yoluna gidilmiştir. Bunun için, D0 veri setinden rastgele örnekleme yapılarak veri azaltımı uygulanmıştır ve sınıf başına düşen örnek sayıları eşitlenmiştir. Modelin çeşitli verilerle eğitilerek test edilmesi ve sonuçların birlikte kıyaslanması için veri azaltımı sonucunda rastgele 5560'ar tane örnek içeren 2 sınıfa sahip veri seti elde edilmiştir. Daha sonra bu işlem D0 veri seti üzerinde 2 defa daha uygulanarak toplamda 3 tane eşit sınıf dağılımına sahip veri seti (D1, D2 ve D3) elde edilmiştir. Tablo 2’de bu çalışmada kullanılan, D0, D1, D2 ve D3 veri setlerine ait detaylar paylaşılmıştır. Bu veri setlerine %70-%30 oranında eğitim ve test veri seti ayrımı uygulanarak modelleme çalışmalarına başlanmıştır.

Tablo2. *Veri Setlerinin Kıyaslanması.*

Veri Seti	Sınıf Adı	Her Bir Sınıftaki Örnek Sayısı	Toplam Örnek Sayısı
D0	Malware	5560	15036
	Malware Değil	9476	
D1	Malware	5560	11120
	Malware Değil	5560	
D2	Malware	5560	11120
	Malware Değil	5560	
D3	Malware	5560	11120
	Malware Değil	5560	

A. 3. Modellemede Kullanılan Makine Öğrenmesi Algoritmaları

Modeller, veri setlerine uygulanan SVM, GB, MLP, MNB, KNN ve RF algoritmaları kullanılarak geliştirilmiştir.

A.3.1. Support Vector Machines (SVM)

SVM algoritması, verileri iki veya daha fazla sınıfa ayırmak için en uygun ayrım hiper düzlemi (ayırıcı) bulmayı amaçlayan bir sınıflandırma yöntemidir. SVM, sınıflar arasında en geniş marjı sağlayacak şekilde bu ayrım hiper düzlemi oluşturarak sınıflandırma başarısını artırır. SVM, doğrusal ve doğrusal olmayan problemlerde kullanılabilen ve özellikle küçük veri setlerinde başarılı sonuçlar elde eden bir yöntemdir [28].

A.3.2. Gradient Boosting (GB)

GB algoritması, zayıf öğrenen modelleri, ardışık olarak eğiterek daha güçlü ve doğru bir sınıflandırıcı oluşturan bir yöntemidir. Her aşamada, önceki aşamalardaki hataları en iyi düzelten model seçilir ve bu model, toplam modelin bir parçası haline gelir. GB, model performansını artırmak için hataların azaltılması sürecini optimize eder ve daha hızlı öğrenme sağlar [29].

A.3.3. Multi Layer Perceptron (MLP)

MLP algoritması, yapay sinir ağlarının bir türüdür ve özellikle sınıflandırma ve regresyon problemlerinde kullanılır. MLP, giriş, çıktı ve ara katmanlardan oluşan bir yapıya sahiptir ve her katmanda birden fazla nöron bulunur. Model eğitimi sırasında, ağırlıklar, geri yayılım algoritması kullanılarak güncellenir ve hedef değerlere yaklaşan bir yapı elde edilir [30].

A.3.4. Naive Bayes (MNB)

Naive Bayes algoritması, istatistiksel olarak Bayes teoremine dayanan bir sınıflandırma algoritmasıdır. Bu teorem, koşullu olasılıkların ilişkilerini hesaplamaya dayanır. Farklı türlerde Naive Bayes sınıflandırıcıları vardır, örneğin Gauss, Bernoulli ve MultiNominal Naive Bayes [31]. Bu çalışmada, Multi Nominal Naive Bayes (MNB) sınıflandırıcısı kullanılmıştır.

A.3.5. K-En Yakın Komşu (KNN)

K-En Yakın Komşu, bir veri noktasının sınıf üyeliğini, komşu veri noktalarına olan yakınlığına dayalı olarak ölçen tembel, parametrik olmayan bir sınıflandırma algoritmasıdır. Sorgu yapılmadığı sürece eğitim setine dayalı ayrım modeli oluşturmaz. Algoritma, temel veri dağılımı hakkında herhangi bir varsayım yapmaz ve verileri komşularına olan yakınlığına göre sınıflandırır [32].

A.3.6. Random Forest (RF)

RF algoritması, makine öğrenme teorisi temelinde çalışan bir denetimli öğrenme algoritmasıdır ve sınıflandırma ve regresyon için ensemble yöntemleri ailesine aittir. Adından da anlaşılacağı gibi, RF, bir topluluk olarak çalışan birçok bireysel karar ağacından oluşur. RF'deki her karar ağacı, bir sınıf tahmini verir ve en yüksek skorlu sınıf modelimizin tahmini olur [33].

A. 4. Performans Ölçütü

Bu çalışmada, karmaşıklık matrisi sonuçlarından doğruluk, özgüllük, duyarlılık ve kesinlik metriklerini elde etmek için öncelikle doğru pozitif (DP), doğru negatif (DN), yanlış pozitif (YP), yanlış negatif (YN) değerleri elde edilmiştir. Elde edilen değerler aşağıdaki eşitlikler [34] kullanılarak sonuç hesaplamasında kullanılmıştır.

$$Doğruluk\ (Accuracy) = \frac{DP + DN}{DP + DN + YP + YN} \quad (1)$$

$$Duyarlılık\ (\ Recall) = \frac{DP}{DP + YN} \quad (2)$$

$$Özgüllük\ (Specifity) = \frac{DN}{DN + YP} \quad (3)$$

$$Kesinlik\ (Precision) = \frac{DP}{DP + YP} \quad (4)$$

A. 5. Modelleme ve Performans Analizi

Modelleme sürecinde, SVM, GB, MLP, MNB, KNN ve RF algoritmaları varsayılan hiperparametrelerle dört veri setine (D0, D1, D2 ve D3) uygulanmıştır. Tablo 3, bu modellerin doğruluk, duyarlılık, özgüllük ve kesinlik sonuçları ile modellerin çalışma sürelerini göstermektedir.

Tablo 3. *Varsayılan Hiperparametrelerle Modelleme Sonuçları.*

Veri Seti	Model	Doğruluk (%)		Duyarlılık (%)		Özgüllük (%)		Kesinlik (%)		Süre
		Eğitim	Test	Eğitim	Test	Eğitim	Test	Eğitim	Test	(dk:sn)
D0	SVM	98.75	98.27	97.32	96.72	99.26	98.60	99.58	99.19	0:08
	GB	97.75	97.23	95.61	94.88	98.22	97.61	98.99	98.62	0:07
	MLP	99.94	98.63	99.85	**97.92**	98.38	98.38	100.00	99.05	0:013
	MNB	92.93	93.08	94.50	95.18	87.34	87.37	92.00	91.84	**0:01**
	KNN	98.68	98.14	97.78	97.20	98.62	97.78	99.20	98.69	0:07
	RF	99.94	**98.65**	99.85	97.38	100.00	**98.97**	100.00	**99.40**	0:012
D1	SVM	98.84	97.87	98.35	97.42	99.32	98.31	99.32	98.31	0:04
	GB	97.54	96.61	97.17	95.86	97.90	97.32	97.90	97.32	0:03
	MLP	99.92	**98.77**	99.85	**98.68**	100.00	98.86	100.00	98.86	0:08
	MNB	92.57	92.83	95.52	96.22	90.19	90.12	89.62	89.44	**0:01**
	KNN	98.37	97.42	98.17	97.48	98.55	97.37	98.56	97.36	0:03
	RF	99.92	98.41	99.85	97.90	100.00	**98.91**	100.00	**98.92**	0:06
D2	SVM	98.64	97.78	97.97	96.88	99.30	**98.66**	99.31	**98.68**	0:03
	GB	97.42	96.58	96.86	95.62	97.94	97.49	97.97	97.54	0:04
	MLP	99.92	**98.50**	99.85	**98.56**	100.00	98.44	100.00	98.44	0:16
	MNB	92.43	92.71	95.50	96.10	89.97	90.01	89.37	89.32	**0:01**
	KNN	98.32	97.36	98.17	97.42	98.45	97.31	98.46	97.30	0:03
	RF	99.92	98.26	99.85	98.02	100.00	98.49	100.00	98.50	**0:01**
D3	SVM	98.86	98.11	98.46	97.48	99.25	98.72	99.26	98.74	0.03
	GB	97.53	96.34	96.73	95.32	98.30	97.31	98.33	97.36	0.04
	MLP	99.91	**98.71**	99.82	**98.38**	100.00	99.03	100.00	99.04	0.08
	MNB	92.37	93.07	95.57	95.98	89.80	90.71	89.16	90.16	**0:01**
	KNN	98.29	97.60	98.25	97.48	98.33	97.72	98.33	97.72	0:06
	RF	99.92	98.59	99.85	98.02	100.00	**99.15**	100.00	**99.16**	**0.01**

Tablo 3'deki sonuçlar, MLP algoritmasının varsayılan hiperparametrelerle D0 haricindeki tüm veri setlerinde en yüksek doğruluk ve duyarlılık değerlerini ürettiğini ortaya koymaktadır. Ayrıca, RF algoritması D0 ve D1 veri kümelerinde özgüllük ve kesinlik açısından en iyi performansı gösterirken, SVM algoritması D2 veri kümesinde, RF algoritması da D3 algoritmasında özgüllük ve kesinlik açısından en yüksek değerler elde edilmiştir.

Modellemenin ikinci aşamasında, performansı artırmak amacıyla, GridSearch yöntemi kullanılarak en iyi hiperparametreler araştırılmıştır. Her algoritma için bulunan en iyi hiperparametrelerin uygulanmasıyla elde edilen doğruluk, duyarlılık, özgüllük, kesinlik değerleri ve çalışma süreleri ise Tablo 4'te sunulmuştur.

Tablo 4'deki sonuçlara göre, D0 veri seti için, SVM algoritmasından %99.07 ile en yüksek doğruluk başarısı elde edilmiştir. Aynı algoritma ile en yüksek duyarlılık, özgüllük ve kesinlik değerleri de, sırasıyla %98.21, %99.28 ve %99.58 olarak elde edilmiştir. MLP algoritması ise D0 veri seti için, %98.82 doğruluk başarısı ile ikinci en iyi performansı gösteren algoritma olmuştur. Ayrıca, duyarlılık,

özgüllük ve kesinlik skorları da sırasıyla %98.15, %98.68 ve %99.22 olarak bulunmuştur. GB ve RF algoritmalarının da %98.58 ve %98.51 doğruluk değerleri ile yüksek performans sergilediği görülmüştür. MNB algoritmasının başarı performansı, diğer sınıflandırıcıların performansları kadar yüksek olmasa da, zaman performansı açısından, en kısa sürede modelleme yapabilen algoritma olduğu görülmüştür. KNN algoritması da, %98.43 doğruluk başarısı ile diğer sınıflandırıcılara kıyasla nispeten düşük performans göstermiştir, zaman performansı açısından da en yavaş çalışan model olduğu gözlemlenmiştir.

D1 veri kümesi için, MLP algoritmasının, en yüksek doğruluk (%98.83), duyarlılık (%98.68), özgüllük (%98.98) ve kesinlik değerleri (%98.98) ile, en iyi performansı sergilediği görülmüştür. SVM algoritması ise, %98.59 doğruluk başarısı ile ikinci en yüksek performansı gösteren algoritma olmuştur. GB ve RF algoritmaları da %98.62 ve %98.41 doğruluk başarı değerleri ile iyi performans göstermişlerdir.

Tablo 4. *En İyi Hiperparametrelerle Modelleme Sonuçları.*

Veri Seti	Model	Doğruluk (%)		Duyarlılık (%)		Özgüllük (%)		Kesinlik (%)		Süre (dk:sn)
		Eğitim	Test	Eğitim	Test	Eğitim	Test	Eğitim	Test	
D0	SVM	99.91	**99.07**	99.77	**98.21**	100.00	**99.28**	100.00	**99.58**	1:10
	GB	99.92	98.58	99.79	97.80	100.00	98.38	100.00	99.05	3:23
	MLP	99.94	98.82	99.85	98.15	100.00	98.68	100.00	99.22	0:59
	MNB	92.97	93.08	94.48	95.18	87.44	87.37	92.08	91.84	0:09
	KNN	99.94	98.43	99.85	97.74	100.00	98.03	100.00	98.83	8:00
	RF	99.90	98.51	99.74	97.20	100.00	98.79	100.00	99.29	1:06
D1	SVM	99.90	98.59	99.79	98.50	100.00	98.68	100.00	98.68	0:27
	GB	99.92	98.62	99.85	98.50	100.00	98.74	100.00	98.74	2:09
	MLP	99.92	**98.83**	99.85	**98.68**	100.00	**98.98**	100.00	**98.98**	0:42
	MNB	92.70	92.83	95.57	96.22	90.37	90.18	89.83	89.44	0:03
	KNN	99.92	98.11	99.85	98.38	100.00	97.85	100.00	97.84	3:00
	RF	99.88	98.41	99.77	97.90	100.00	98.91	100.00	98.92	0:37
D2	SVM	99.90	**98.71**	99.79	98.44	100.00	**98.98**	100.00	**98:98**	0:34
	GB	99.92	98.50	99.85	98.50	100.00	98.50	100.00	98.50	2:04
	MLP	99.92	98.62	99.85	**98.56**	100.00	98.68	100.00	98.68	0:52
	MNB	92.46	92.77	95.50	96.10	90.00	90.10	89.42	89.44	0:06
	KNN	99.92	97.66	99.85	98.20	100.00	97.15	100.00	97.12	4:00
	RF	99.88	98.32	99.77	98.02	100.00	98.61	100.00	98.62	0:42
D3	SVM	98.91	98.74	99.82	98.26	100.00	99.21	100.00	**99.22**	0:23
	GB	99.92	98.68	99.85	**98.44**	100.00	98.92	100.00	98.92	2:00
	MLP	99.92	**98.80**	99.85	98.14	100.00	**99.45**	100.00	96.46	0:43
	MNB	92.38	93.04	95.57	95.92	89.82	90.70	89.19	90.16	0:03
	KNN	99.92	98.47	99.85	98.38	100.00	98.56	100.00	98.56	3:00
	RF	99.88	98.56	99.77	97.90	100.00	99.21	100.00	99.22	0:37

D2 veri kümesi için de SVM algoritması, %98.71 doğruluk başarı değeriyle en yüksek performansı gösteren algoritma olmuştur. MLP algoritması da, SVM algoritmasıyla benzer performans sonuçları sergilemiş, %98.62 doğruluk başarı değeri elde edilmiştir. GB ve RF algoritmalarının ise, sırasıyla %98.50 ve %98.32 doğruluk başarı değerleri ile saygın sonuçlar sergiledikleri görülmüştür.

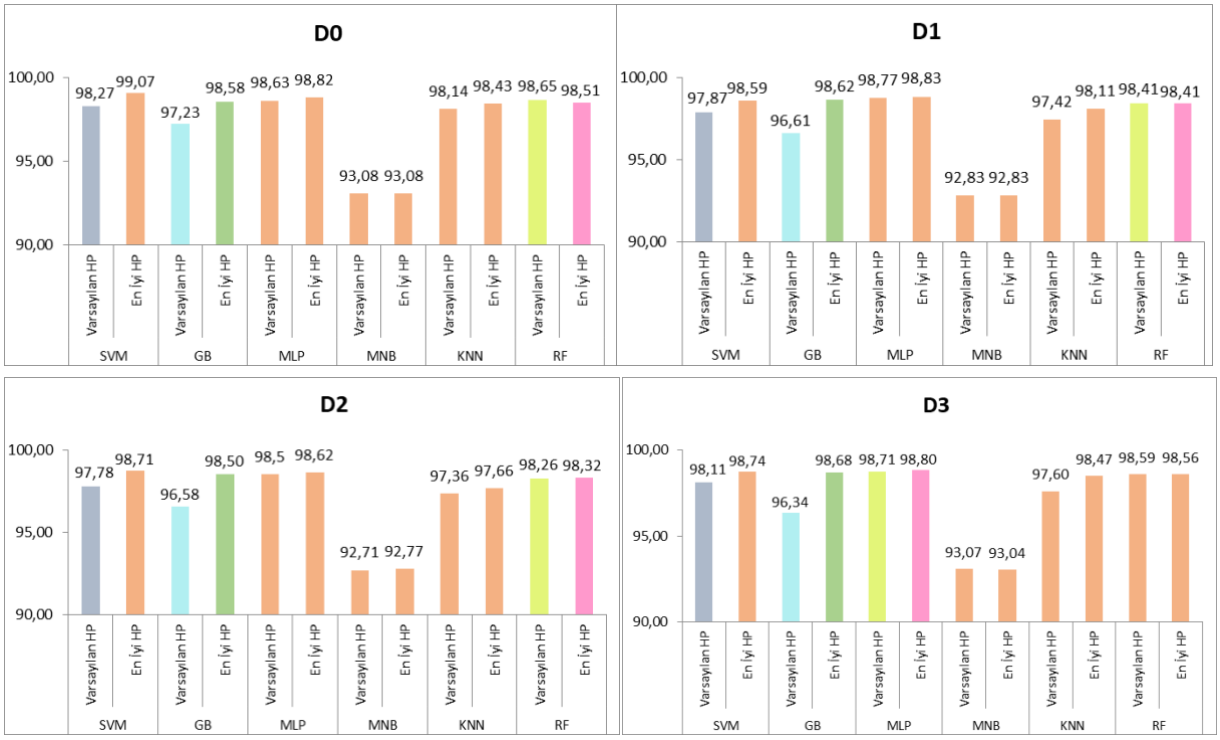
Son olarak, D3 veri kümesi için de, MLP algoritması ile en yüksek doğruluk başarısı, %98.80 olarak elde edilmiştir. SVM algoritması ise, %98.74 doğruluk başarısı ile ikinci sırada yer almıştır. GB ve RF algoritmalarının da %98.68 ve %98.56 doğruluk başarı değerleriyle, MLP ve SVM algoritmalarına yakın performanslar sergilediği görülmüştür.

Tablo 4'teki değerlere göre, GridSearch yöntemiyle bulunan en iyi hiperparametrelerle, D0 veri seti için en yüksek performans metrikleri SVM algoritması kullanılarak elde edilmiştir. D1 veri seti için ise, tüm performans metriklerine göre en başarılı model, MLP algoritması kullanılarak elde edilmiştir. D2 veri

seti için en yüksek doğruluk, özgüllük ve kesinlik başarıları SVM algoritması kullanılarak alınmıştır. D3 veri seti için ise en yüksek doğruluk ve özgüllük başarıları, MLP algoritması kullanılarak elde edilmiştir.

III. BULGULAR VE TARTIŞMA

Tüm veri setlerinin performansları, Tablo 3 ve 4'teki değerlere göre karşılaştırıldığında, genel olarak, GridSearch ile bulunan en iyi hiperparametreler kullanılarak elde edilen doğruluk başarısı sonuçlarının, varsayılan hiperparametrelerin kullanılmasıyla elde edilen sonuçlara kıyasla daha yüksek olduğu görülmüştür. Bu durum, her bir veri seti ve algoritma için Şekil 2'de de gözlemlenebilir.

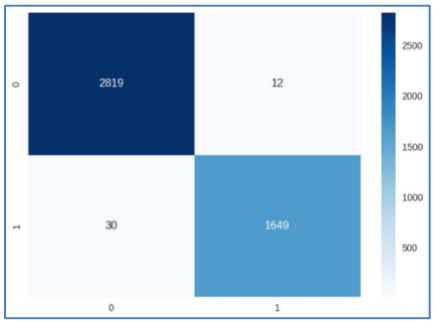


Şekil 2. *Her bir veri seti için varsayılan ve en iyi hiperparametre doğruluk başarı sonuçlarının karşılaştırılması.*

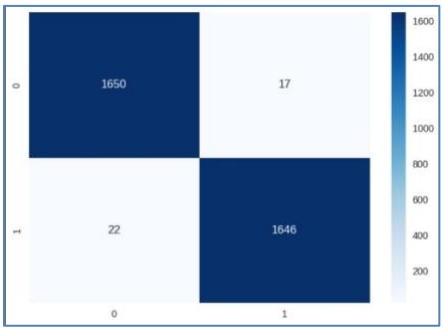
Şekil 2'de görüldüğü üzere, D0 veri seti için en iyi hiperparametre değerleriyle oluşturulan SVM modeli, en başarılı model olarak belirlenmiştir. GridSearch çalışmaları sonucunda SVM modeli için elde edilen en iyi hiperparametreler şunlardır: C=10, gamma=0.1 ve kernel=rbf. Bu hiperparametre değerleri kullanıldığında, model D0 veri seti için en yüksek doğruluk (%99.07), özgüllük (%99.28) ve kesinlik (%99.58) değerlerine ulaşmıştır.

İlk aşamada amaç, varsayılan hiperparametrelerle elde edilen başarı sonuçlarını artırmak için hiperparametre optimizasyonu yapmaktır. Aynı modellerin varsayılan hiperparametre ve en iyi hiperparametrelerle elde edilen sonuçları karşılaştırıldığında, en fazla başarı artışının GB modelinde olduğu görülürken, D0 ve D2 veri setleri için en başarılı sonuçlar SVM algoritması ile; D1 ve D3 veri setleri için ise en başarılı modeller MLP algoritması ile elde edilmiştir.

Tüm modeller kıyaslandığında, dengesiz veri seti (D0) için en iyi hiperparametrelerin kullanıldığı SVM modeli, %99.07 doğruluk başarısıyla en iyi model olarak belirlenmiştir. Bu modelin karmaşıklık matrisi Şekil 3'te gösterilmektedir. Dengeli veri setleri (D1, D2 ve D3) için uygulanan modeller arasında ise, D1 veri seti için en iyi hiperparametrelerin kullanıldığı MLP modeli %98.83 doğruluk başarısıyla ikinci en iyi model olarak bulunmuştur ve bu modelin karmaşıklık matrisi de Şekil 4'te gösterilmiştir.



Şekil 3. *D0 veri seti için en iyi hiperparametrelerin kullanıldığı SVM modeline ait karmaşıklık matrisi*



Şekil 4. *D1 veri seti için en iyi hiperparametrelerin kullanıldığı MLP modeline ait karmaşıklık matrisi*

IV. SONUÇ

Bu çalışmada, Android kötü amaçlı yazılım tespiti için SVM, GB, MLP, MNB, KNN ve RF makine öğrenmesi algoritmaları kullanılarak modeller oluşturulmuş ve bu modellerin performansları karşılaştırılmıştır. DREBIN veri seti üzerinde gerçekleştirilen modelleme çalışmaları sonucunda, bu çalışma Android kötü amaçlı yazılım tespiti için ilk olarak, %99.07 doğruluk başarısına sahip olan SVM modelini ve ikinci olarak da %98.83 doğruluk başarısı ile MLP modelini önermektedir. Hiperparametre optimizasyonu ile geliştirilen bu modeller, diğer algoritmalara kıyasla daha yüksek doğruluk başarısı sunmaktadır. Bu nedenle, Android kötü amaçlı yazılım tespiti ve ilgili çalışmalarda kullanılacak model olarak SVM ve MLP modelleri önerilmektedir.

Önerilen modellerin başarısı yanında, bu çalışmada elde edilen sonuçlar hiperparametre optimizasyonunun önemini de vurgulamaktadır. Hiperparametre optimizasyonu, modelin performansını önemli ölçüde iyileştirerek, Android kötü amaçlı yazılım tespiti gibi kritik alanlarda daha güvenilir ve etkili sonuçlar elde etmeye yardımcı olduğu görülmüştür.

Bu çalışmada elde edilen sonuçlar, makine öğrenmesi algoritmalarının Android cihazlarda kötü amaçlı yazılım tespiti için etkili ve güvenilir çözümler sunabileceğini göstermektedir. Bu çalışmaların devamı olarak, daha geniş veri setleri ve farklı algoritmalar kullanarak daha başarılı modeller geliştirilebilir. Ayrıca, gerçek zamanlı tespit sistemleri ve güvenlik uygulamaları ile bu modellerin kullanılması sayesinde kullanıcıların güvenliği daha da artırılabilir.

Gelecek çalışmalarda, farklı makine öğrenmesi ve derin öğrenme yöntemleri, öznitelik seçimi ve öznitelik mühendisliği teknikleri, sınıf dengesizliği sorununun çözümüne yönelik yöntemler ve farklı performans metriklerinin kullanılması gibi öneriler dikkate alınarak, Android kötü amaçlı yazılım tespiti alanında daha başarılı ve güvenilir modeller geliştirilebilir. Bu sayede, Android işletim sistemine sahip cihazlarda güvenlik ve gizlilik risklerinin azaltılması ve kullanıcıların korunması sağlanabilir.

Sonuç olarak, bu çalışma ve önerilen SVM ve MLP modelleri, Android kötü amaçlı yazılım tespiti alanında önemli bir katkı sunmaktadır. Hiperparametre optimizasyonu ve model seçimi, bu alanda daha iyi sonuçlar elde etmek için kritik faktörlerdir. Gelecekteki çalışmalar, bu alandaki bilgi birikimini ve teknolojinin etkinliğini daha da artırabilir ve böylece kullanıcıların cihazlarını ve verilerini korumaya yönelik daha güvenilir ve etkili çözümler sunabilir.

V. KAYNAKLAR

[1] A. T. Kabakuş, İ. A. Doğru and A. Çetin, "Android kötücül yazılım tespit ve koruma sistemleri", Erciyes Üniversitesi Fen Bilimleri Enstitüsü Fen Bilimleri Dergisi, vol. 31, no. 1, pp. 9-16, Feb. 2015.

[2] M. Grace, Y. Zhou, Q. Zhang, S. Zou and X. Jiang, "RiskRanker: scalable and accurate zero-day android malware detection", MobiSys '12: Proceedings of the 10th international conference on Mobile systems, applications, and services, June 2012, Pages 281–294, https://doi.org/10.1145/2307636.2307663

[3] N. Zhang, Y. Tan, C. Yang and Y. Li, "Deep learning feature exploration for Android malware detection", Applied Soft Computing, Volume 102, April 2021, https://doi.org/10.1016/j.asoc.2020.107069

[4] A. Razgallah, R. Khoury, S. Halle and K. Khanmohammadi, "A survey of malware detection in Android apps: Recommendations and perspectives for future research", Computer Science Review**,** Volume 39, February 2021, https://doi.org/10.1016/j.cosrev.2020.100358

[5] A. Guerra-Manzanares, M. Luckner and H. Bahsi, "Concept drift and cross-device behavior: Challenges and implications for effective android malware detection", Computers & Security**,** Volume 120, September 2022, https://doi.org/10.1016/j.cose.2022.102757

[6] F. Ou and J. Xu, "S^3Feature: A static sensitive subgraph-based feature for android malware detection", Computers & Security**,** Volume 112, January 2022, https://doi.org/10.1016/j.cose.2021.102513

[7] A. Martin, R. Lara-Cabrera and D. Camacho, "Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset", Information Fusion, Volume 52, December 2019, Pages 128-142, https://doi.org/10.1016/j.inffus.2018.12.006

[8] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture", Computer Networks**,** Volume 171, 22 April 2020, https://doi.org/10.1016/j.comnet.2020.107138

[9] A. Ananya, P. Vinod and M. Shojafar, "SysDroid: A Dynamic ML-based Android Malware Analyzer using System Call Traces", Cluster Computing, December 2020, DOI:10.1007/s10586-019-03045-6

[10] K. Lin, X. Xu and F. Xiao, "MFFusion: A Multi-level Features Fusion Model for Malicious Traffic Detection based on Deep Learning", Computer Networks**,** Volume 202, 15 January 2022, https://doi.org/10.1016/j.comnet.2021.108658

[11] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher and M. Portmann, "Graph neural network-based android malware classification with jumping knowledge", 2022 IEEE Conference on Dependable and Secure Computing (DSC) , 1–9, 2022.

[12] L. Onwuzurike, E. Mariconti, P. Andriotis, E. D. Cristofaro, G. J. Ross and G. Stringhini, "Mamadroid: Detecting android malware by building markov chains of behavioral models", ACM Trans. Priv. Secur. 22, 14:1–14:3, 2019.

[13] Y. Wu, X. Li, D. Zou, W. Yang, X. Zhang and H. Jin, "Malscan: Fast market-wide mobile malware scanning by social-network centrality analysis", in: 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, IEEE. pp. 139–150, 2019.

[14] P. Xu, C. Eckert and A. Zarras, "Detecting and categorizing android malware with graph neural networks", in: SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, pp. 409–412, 2021.

[15] H. Gao, S. Cheng and W. Zhang, "Gdroid: Android malware detection and classification with graph convolutional network", Comput. Secur. 106, 2021.

[16] M.S. Rana, S. S. M. M. Rahman, and A. H. Sung, "Evaluation of tree based machine learning classifiers for android malware detection", Computational Collective Intelligence: 10th International Conference, ICCCI 2018, Bristol, UK, September 5-7, 2018, Proceedings, Part II 10. Springer International Publishing, 2018.

[17] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, and I. Molloy, I. "Using probabilistic generative models for ranking risks of android apps", In Proceedings of the 2012 ACM conference on Computer and communications security (pp. 241-252), 2012.

[18] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. E. R. T. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket", In Ndss, Vol. 14, pp. 23-26, 2014.

[19] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection", in IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3216-3225, July 2018, doi: 10.1109/TII.2017.2789219.

[20] M. Qiao, A. H. Sung and Q. Liu, "Merging Permission and API Features for Android Malware Detection", 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Kumamoto, Japan, 2016, pp. 566-571, doi: 10.1109/IIAI-AAI.2016.237.

[21] A. Aydın , İ. A. Doğru and M. Dörterler , "Makine Öğrenmesi Algoritmalarıyla Android Kötücül Yazılım Uygulamalarının Tespiti", *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, vol. 22, no. 2, pp. 1087-1094, Aug. 2018, doi:10.19113/sdufbed.20066

[22] A. Güngör , İ. Dogru , N. Barışçı and S. Toklu , "Görüntü tabanlı özelliklerden ve makine öğrenmesi yöntemlerinden faydalanılarak kötücül yazılım tespiti", Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, vol. 38, no. 3, pp. 1781-1792, Jan. 2023, doi:10.17341/gazimmfd.994289

[23] A. Utku, İ. A. Doğru and M. A. Akcayol, "Decision tree based android malware detection system", 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2018, pp. 1-4, doi: 10.1109/SIU.2018.8404151.

[24] Z. Liu, R. Wang, N. Japkowicz, H. M. Gomes, B. Peng and W. Zhang, "SeGDroid: An Android malware detection method based on sensitive function call graph learning", Expert Systems with Applications, 2023, https://doi.org/10.1016/j.eswa.2023.121125

[25] S. Yang, Y. Wang, H. Xu, F. Xu and M. Chen, "An Android Malware Detection and Classification Approach Based on Contrastive Lerning", Computers & Security Volume 123, 2022, https://doi.org/10.1016/j.cose.2022.102915

[26] J. Sahs and L. Khan, "A Machine Learning Approach to Android Malware Detection," 2012 European Intelligence and Security Informatics Conference, Odense, Denmark, 2012, pp. 141-147, 2012

doi: 10.1109/EISIC.2012.34.

[27] Ö. Kiraz and İ. A. Doğru, "Android Kötücül Yazılım Tespit Sistemleri İncelemesi", Düzce Üniversitesi Bilim ve Teknoloji Dergisi, vol. 5, no. 1, pp. 281-298, Jan. 2017.

[28] S. Haykin, "Neural Networks: A Comprehensive Foundation", Prentice- Hall, Ontario, 837s, 1999.

[29] J. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine", The Annals of Statistics, 29(5), 11-28, 2000.

[30] V. N. Vapnik "The Nature of Static Learing Theory", Springer, 314s, 2000.

[31] J. VanderPlas, "Python Data Science Handbook Essential Tools for Working with Data", O'Reilly Media, 2016.

[32] G. O. Campos, A. Zimek, J. Sander, R.J.G.B. Campello, B. Micenková, E. Schubert, I. Assent and M.E. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study", Data Mining and Knowledge Discovery, vol. 30, no. 4, pp. 891–927, 2016.

[33] L. Breiman, "Random Forests", Statistics Department University of California Berkeley, 1- 33, 2001.

[34] B. J. Erickson and F. Kitamura, “Magician's Corner: 9. Performance Metrics for Machine Learning Models”, Radiology. Artificial intelligence vol. 3, 3e, 2021, doi:10.1148/ryai.2021200126.