YÖNETİM BİLİŞİM SİSTEMLERİ DERGİSİ
http://dergipark.gov.tr/ybs

Yayın Geliş Tarihi: 29.10.2019
Yayına Kabul Tarihi: 23.12.2019
Online Yayın Tarihi: 31.12.2019

Cilt:5, Sayı:2, Yıl:2019, Sayfa:21-28
e-ISSN: 2630-550X

DERİN OTOMATİK KODLAYICI TABANLI ÖZELLİK ÇIKARIMI İLE ANDROİD KÖTÜCÜL YAZILIM UYGULAMALARININ TESPİTİ

Murat UÇAR[1], Emine UÇAR[2]

[1,2] Yönetim Bilişim Sistemleri, İşletme ve Yönetim Bilimleri Fakültesi, İskenderun Teknik Üniversitesi, Hatay Türkiye

ÖZET

Günümüzde akıllı telefonlar insan hayatının vazgeçilmez bir parçası haline gelmiştir. Android işletim sistemi bu cihazlar arasında en yüksek kullanım oranına sahiptir. Gelişmiş özellikleri sayesinde kullanıcıların fotoğrafları, sağlık verileri, kimlik bilgileri ve banka bilgileri gibi kişisel bilgilerini saklamalarını sağlar. Yaygın kullanımı ve gelişmiş özellikleri nedeniyle kötü amaçlı yazılım geliştiricileri tarafından en çok hedeflenen işletim sistemidir. Bu çalışmada Android kötücül yazılım uygulamalarının tespitinde başarıyı artırmak için öncelikle derin oto kodlayıcı mimarisi kullanılarak özellik çıkarımı yapılmıştır. Bir sonraki aşamada ise makine öğrenmesi yöntemlerinden Rasgele Orman (RO), K-En Yakın Komşu (K-EYK) ve Karar Ağacı (KA) algoritmaları kullanılarak sınıflandırma yapılmıştır. Deneysel sonuçlar derin oto kodlayıcı ve temel bileşen analizi kullanarak özellik çıkarımının başarıyı artırdığını göstermiştir. Yapılan analizlere göre, Rastgele Orman algoritmasının % 94,40 ile en iyi doğruluğa sahip olduğu görülmüştür.

Anahtar Kelimeler: Güvenlik, Kötü Amaçlı Yazılım Algılama, Makine Öğrenmesi, Oto Kodlayıcı

THE DETECTION OF ANDROID MALWARE APPLICATIONS WITH DEEP AUTOENCODER BASED FEATURE EXTRACTION

ABSTRACT

Nowadays, smart phones have become an indispensable part of human life. The Android operating system has the highest utilization rate among these devices. Its advanced features allow users to store personal information such as photos, health data, identity information, and bank information. It is the most targeted operating system by malware developers because of its widespread usage and advanced features. In this study, in order to increase the success in the detection of Android malware applications, feature extraction was performed by using deep autoencoders. In the next step, data are classified by Random Forest (RF), K-Nearest Neighbor (K-NN) algorithm and Decision Tree (DT) algorithms. Experimental results showed that feature extraction using deep autoencoders and principal component analysis increased the accuracy of model. According to the analyses made, it has been observed that Random Forest algorithm had the best accuracy with 94.40%.

Keywords: Security, Malware Detection, Machine Learning, Autoencoder

GİRİŞ

Son yıllarda mobil cihazların kullanımı konuşma ve mesajlaşma gibi haberleşme işlevlerinin yanı sıra kişisel bilgisayarların yapabildiği hemen hemen her türlü uygulamayı yapabilir hale gelmesi ile birlikte önemli oranda artmıştır. International Data Corporation Media Center'in yapmış olduğu araştırma 2018 yılında Android destekli cihazların dağılımının 1,4 milyar olduğunu ve uzun vadeli tahminlerde 2023 yılında bu sayının yaklaşık 1,6 milyar adede ulaşmasının beklendiğini öngörmektedir. Aynı araştırma Android işletim sisteminin %85 oranıyla dünya çapında en çok kullanılan mobil işletim sistemi olduğunu belirtmektedir. (International Data Corporation Media Center, 2019). Android işletim sisteminin geniş kullanımı ve bu cihazlar için geliştirilen uygulamaların yaygınlaşması, Android platformlarını kullanan mobil cihazları saldırganların hedefi konumuna getirmektedir. Bunun yanı sıra Android işletim sisteminin, iOS işletim sisteminin aksine kullanıcılarını sadece belirli bir uygulama marketi üzerinden yükleme yapmaları konusunda sınırlamaması ve uygulamaların cihazlara Google Play Store haricinde farklı ortamlardan yüklenebilmesi de ciddi oranda güvenlik açıklarına sebep olabilmektedir.

Yukarıda sayılan tüm bu nedenler araştırmacı ve akademisyenleri Android kötücül yazılım uygulamalarının tespit edilebilmesi amacıyla çeşitli algoritmalar geliştirmeye yöneltmiştir. Android kötücül yazılımların tespit edilebilmesi için statik ve dinamik olmak üzere iki temel analiz yöntemi bulunmaktadır. Statik analizde uygulamaların kötücül olup olmadığını belirlemek için uygulamalar cihaza yüklenmeden önce analiz edilirken dinamik analizde uygulamaların kötücül olup olmadığının belirlenebilmesi için uygulama önce cihaza yüklenir daha sonra uygulamanın davranışlarının analizine göre karar verilir. Literatürde bu yöntemleri kullanarak önerilmiş çeşitli yaklaşımlar bulunmaktadır.

Li ve arkadaşları, derin inanç ağlarına dayalı bir kötü amaçlı yazılım tespit etme sistemi tasarlamışlardır. Kötü niyetli yazılımların tespiti için tehlikeli API çağrıları ve riskli izinler olmak üzere iki tür özellik kullanmışlardır. Önermiş oldukları yaklaşım %90 ın üzerinde bir doğruluk oranı göstermiştir (Li vd., 2018). Xu ve arkadaşları, farklı gizli katmanlara sahip derin sinir ağları kullanarak DeepRefiner adını verdikleri kötü amaçlı yazılım tanıma sistemini geliştirmişlerdir. Geliştirmiş oldukları sistem, kötü amaçlı yazılımları %97.74 doğruluk oranı ile tespit etmeyi başarmıştır (Xu vd., 2018). Wang ve arkadaşları, Android kötü amaçlı yazılım tespit sisteminin doğruluğunu artırmak için derin otomatik kodlayıcı ve konvolüsyonel sinir ağlarına dayalı hibrit bir model önermişlerdir. Deneysel sonuçları seri konvolüsyonel sinir ağı yapısının en iyi doğruluğu sağladığını göstermiştir (Wang, Zhao ve Wang 2018). Zhu ve arkadaşları, kötü amaçlı Android yazılımlarını belirlemek için derin öğrenmeye dayalı DeepFlow'u önermişlerdir. Sonuçlar, DeepFlow'un geleneksel bilgisayar öğrenmesine dayalı yaklaşımlardan daha iyi bir performans göstererek % 95.05 gibi yüksek bir F1 skoru elde ettiğini göstermiştir (Zhu vd., 2017).

Yuan ve arkadaşları, kötü amaçlı yazılımları tespit edebilmek için statik analiz özelliklerini, Android uygulamalarının dinamik analiz özellikleri ile ilişkilendirerek derin öğrenmeye dayalı bir model önermişlerdir. Bir uygulamanın kötü amaçlı olup olmadığını anlamak için çevrimiçi bir derin öğrenme tabanlı Android kötü amaçlı yazılım algılama motoru(DroidDetector) uygulamışlardır. Deneysel sonuçlar, DroidDetector'ın % 96.76 doğruluk oranı ile geleneksel makine öğrenme tekniklerinden daha iyi bir performans sağladığını göstermiştir (Yuan, Lu ve Xue, 2016). Yuxin ve Siyi, kötü amaçlı yazılımları tespit etmek için derin inanç ağları yöntemini uygulamışlardır. Önerdikleri modelin performansını destek vektör makineleri, karar ağaçları ve k en yakın komşu algoritmasını sınıflandırıcı olarak kullanan üç temel kötü amaçlı yazılım algılama modeliyle karşılaştırmışlardır. Deneysel sonuçlar derin inanç ağlarının daha iyi bir performans sağladığını göstermiştir (Yuxin ve Siyi, 2017). He ve arkadaşları, derin oto kodlayıcıyı temel alan bir Android kötü amaçlı yazılım algılama yöntemi önermiştir. Tasarlanan oto kodlayıcı yapısı APK'dan çıkarılan ve dönüştürülen özellik vektörlerinin boyutunu azaltmak için kullanılmıştır ve sınıflandırma için lojistik regresyon modeli uygulanmıştır. Deney sonuçları önerilen yöntemin iyi bir performans sağladığını göstermiştir (He vd., 2018). Yousefi-Azar ve arkadaşları, kötü amaçlı yazılımları sınıflandırabilmek için oto kodlayıcı tabanlı yeni bir özellik öğrenimi yöntemi önermişlerdir. Çalışmalarında otomatik özellik öğrenme modelleri olan oto kodlayıcıların, diğer yaklaşımların aksine daha ayırt edici özellikler sağlayabildiğini göstermişlerdir (Yousefi-Azar vd., 2017). Naway ve Li, Android kötü amaçlı yazılım tespiti için derin öğrenme tabanlı bir yaklaşım önermişlerdir. Önerilen yaklaşım beş farklı özellik setini araştırmakta ve

otomatik kodlayıcıyı kötü amaçlı yazılımları sınıflandırmak için kullanmaktadır. Deneysel sonuçlar, önerilen yaklaşımın kötü amaçlı yazılımları %96,81 gibi yüksek bir doğrulukla tanımlayabildiğini göstermiştir (Naway ve Li, 2019). John ve arkadaşları, yapmış oldukları çalışmada Android kötücül yazılım ailelerinin tespiti için derin oto kodlayıcı tabanlı özellik çıkarımı yöntemini uygulamışlardır. Sonuçlar oto kodlayıcı tabanlı özellik çıkarımı yöntemini kullanarak yapılan sınıflandırmanın daha başarılı olduğunu göstermiştir (John, Thomas ve Uddin, 2017).

Zhou ve arkadaşları, bu çalışmalarında geleneksel makine öğrenmesi yöntemlerinde Monte Carlo algoritmasını kullanarak ağırlıkları rasgele ayarlayan yeni bir metot önermişlerdir. Yapmış oldukları karşılaştırmalı deneysel sonuçlar önerilen yöntemin kötü amaçlı yazılımları tespit etmede en iyi performansı elde ettiğini göstermiştir (Zhou vd., 2019).

Vinayakumar ve arkadaşları, çalışmalarında Android kötü amaçlı yazılım tespiti için statik ve dinamik özelliklere dayalı, tekrarlayan bir sinir ağı türü olan Uzun Kısa Süreli Belleğin (LSTM) kullanılmasını önermiştir. Sonuçlar LSTM tarafından gerçekleştirilen sınıflandırmanın iyi bir performans elde ettiğini göstermiştir (Vinayakumar vd., 2018). Alshahrani ve arkadaşları, DDefender ismini verdikleri bir kötü amaçlı yazılım tanımlama sistemi geliştirmişlerdir. Bu sistem; kullanıcının telefonunda dinamik analiz için önceden çalışan ve kullanıcılara analiz raporu sağlayan bir uygulama ile sunucu tarafında statik analiz ve tanımlama prosedürünü hazırlayarak sonuçları müşteri tarafına geri gönderen bir sistem olmak üzere iki ana parçadan oluşmaktadır. Yapılan deneyler, önerilen sistemin %95' e varan bir tespit doğruluğu elde ettiğini göstermiştir (Alshahrani vd., 2018).

Bu çalışmada Android kötücül yazılımlarının tespiti için Lashkari ve arkadaşları, tarafından üretilen CICAAGM veri seti kullanılmıştır (Lashkari vd., 2017). Lashkari ve arkadaşları sınıflandırma yapmadan önce özellik çıkarımı için, Java'da geliştirilmiş olan ve başka veri setlerine de uyarlanabilen CICFlowMeter uygulamasını kullanmışlardır. Bu çalışma kapsamında kullanılan Android uygulamalarının sınıflandırılması için öncelikle derin oto kodlayıcılar kullanılarak öznitelik indirgeme işlemi gerçekleştirilmiş daha sonra geleneksel makine öğrenmesi yöntemlerinden k en yakın komşu, rastgele orman ve karar ağaçları kullanılmıştır.

Çalışmanın ikinci bölümünde kullanılan veri seti ve yöntemler sunulmuştur. Üçüncü bölümünde deneysel çalışma sonucu elde edilen bulgular paylaşılmıştır. Dördüncü bölümünde ise çalışmadan elde edilen sonuçlara değinilmiştir.

YÖNTEM

Çalışmanın bu bölümünde kullanılan veri seti ve kötü amaçlı yazılım tespiti için önerilen yöntemlere ait tanımlamalar bulunmaktadır.

Veri Seti

Bu çalışma kapsamında kullanılan veriler University of New Brunswick, Canadian Institute for Cybersecurity (https://www.unb.ca/cic/datasets/android-adware.html) internet sitesinden alınmıştır. Lashkari ve arkadaşları, tarafından üretilen CICAAGM veri seti aşağıdaki üç kategoriye sahip 1900 uygulamadan elde edilen verileri içermektedir. (Lashkari vd., 2017).

Adware kötücül uygulamalar: Airpush, Dowgin, Kemoge, Mobidash ve Shuanet ailelerini içeren 250 uygulamadan oluşmaktadır.

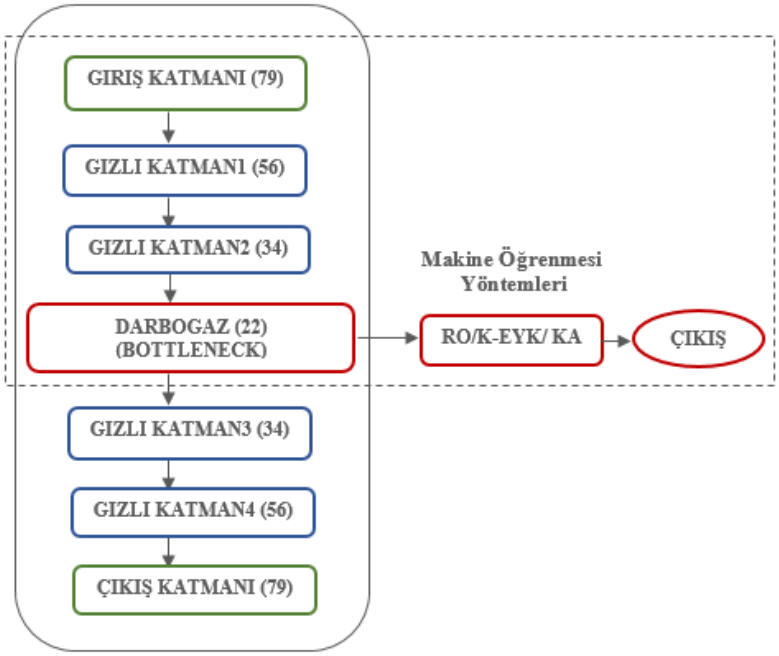
General Malware kötücül uygulamalar: AVpass, FakeAV, FakeFlash/FakePlayer, GGtracker ve Penetho ailelerini içeren 150 uygulamadan oluşmaktadır.

Normal Uygulamalar: 2015-2016 yıllarında Googleplay marketten elde edilen 1500 iyicil uygulamadan oluşmaktadır.

Kullanılan Yöntemler

Bu çalışmada öncelikle başlangıçta 79 olan öznitelik sayısı derin oto kodlayıcı mimarisi kullanılarak 22 özniteliğe indirgenmiştir. Daha sonra makine öğrenmesi teknikleri ile kötücül yazılım tespiti gerçekleştirilmiştir. Bu çalışmada önerilen kötücül yazılım tespit mimarisi Şekil 1'de gösterildiği gibidir.

Şekil 1. Kötücül yazılım tespiti için önerilen model mimarisi



Derin oto kodlayıcı (Deep autoencoder)

Oto kodlayıcılar, giriş ve çıkış katmanları aynı özelliklere sahip olan çok katmanlı algılayıcıdan oluşan bir makine öğrenme metodudur. Oto kodlayıcı algoritması, ilk olarak 1980'li yıllarda Hinton ve PDP grup tarafından ortaya konulmuştur. 2006 yılında derin öğrenme(deep learning) mimarisinin geliştirilmesi ile makine öğrenmesi alanında ana konulardan biri olmuştur. Derin Oto kodlayıcı modeli uygulamada daha az öznitelik kullanarak daha fazla başarı oranı elde etmeyi amaçlayan bir derin öğrenme yöntemidir. Veri setindeki öznitelik sayısını düşürerek, hem doğruluk oranını hem de hızı artırmayı hedeflemektedir (Baldi, 2011). Bir oto kodlayıcının şifreleme ve şifre çözme olmak üzere iki aşaması vardır. Şifreleme aşamasında, verilen girişi alır ve gizli katmanlarını kullanarak girişi, giriş birimlerinden daha küçük birimlerle ifade etmeye çalışır. Şifre çözme aşamasında ise, şifrelenmiş bilgiyi gizli katmanlarını kullanarak verilen girişi yeniden oluşturmaya çalışır (Bengio, 2009). Derin oto kodlayıcılar özellik çıkarma ve veri sıkıştırma işlemlerinde sıklıkla kullanılmaktadır.

Rasgele orman

Rastgele orman (RO), 2001 yılında Leo Breiman tarafından ortaya atılan ve birden çok karar ağacının birleşiminden oluşan bir modeldir. (Breiman 2001). RO, her bir düğümden rastgele olarak seçilmiş değişkenlerden en iyisini kullanarak düğümleri dallara ayırır. Orijinal veri setinden oluşturulan her bir veri seti için yer değiştirmeli olarak üretilen bir yöntem kullanır. Ardından rastgele özellik seçimi kullanılarak ağaçlar geliştirir ve geliştirilen ağaçları budamaz (Archer 2008; Breiman 2001).

Çalışmada RO sınıflandırıcısı uygulanırken, en uygun parametreleri belirlemek için tekrarlı denemeler yapılmıştır. Denemeler sonucunda seçilen parametrelerin hem işlem süreleri hem de test doğrulukları incelenerek RO sınıflandırıcısı için en uygun ağaç sayısının 80 olmasına karar verilmiştir. Dallanma kriteri olarak Gini indeksi kullanılmıştır ve ağacın maksimum derinliği 26 seçilmiştir.

K en yakın komşu

K-En Yakın Komşu (K-EYK) algoritması en çok kullanılan örüntü tanıma yöntemlerinden biri olup nesneleri öznitelik uzayındaki en yakın eğitim örneklerine göre sınıflandırır (Tatlıdil, 1996). K-EYK algoritması sınıflandırma işlemi için verilen K değeri kadar en yakın komşunun sınıfına göre

değerlendirme yapmaktadır. Bu algoritmada bir vektörün sınıflandırılması, sınıfı bilinen vektörler kullanılarak yapılır. Test edilmek istenen bir örnek seçildiğinde bu örneğin sınıfını belirlemek için eğitim kümesindeki o örneğe en yakın olan k adet örnek seçilir ve seçilen bu örneklerden oluşmuş kümede hangi sınıfa ait en çok örnek varsa test edilen örnek bu sınıfa aittir denir. Örnekler arasındaki uzaklıklar eşitlik 1'deki Öklid uzaklığı ile hesaplanır.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} \left(dizi_r(x_i) - dizi_r(x_j)\right)^2} \quad (1)$$

Uygulamada en uygun k değerini belirlemek için denemeler yapılmıştır. Yapılan denemeler sonucunda k değeri 6 seçildiğinde en iyi sonucun elde edildiği görülmüştür.

Karar ağacı

Karar ağacı, hem regresyon hem de sınıflandırma problemlerinde kullanılabilen denetimli bir öğrenme algoritması türüdür. Karar ağacı tekniği herhangi bir karar problemi için kullanılabilir fakat birden fazla kararın ardışık olarak verilmesi gereken durumlarda karar problemlerinin gösteriminde çok daha kullanışlıdır (Albright, Winston ve Zappe, 2006). Karar ağacının ilk düğümüne kök düğüm, ara düğümlerine yaprak düğümleri ve ağacın son düğümlerine uç düğüm denir. Karar ağacının her bir düğümündeki değişkenler, bir ayırma algoritması kullanılarak eğitim veri setine karşı test edilir. Hangi değişkenin ağaç kökünde test edilmesi gerektiği sorusundan yola çıkarak karar ağacı algoritması ağacı temelden oluşturur. Her düğüm için oluşturulacak dal sayısı, kullanılan algoritmaya ve seçilen değişkenin değer sayısına bağlıdır.

Bu çalışma kapsamında, karar ağacı algoritmaları uygulanırken dallanma kriteri olarak Gini indeksi kullanılmıştır ve ağacın maksimum derinliği 26 seçilmiştir.

BULGULAR

Çalışmada 1500 iyicil, 400 kötücül olmak üzere 1900 uygulamadan elde edilen veriler kullanılmıştır. Android kötücül uygulamaların tespiti için oluşturulan yazılım, Google Colaboratory platformu üzerinde Python programlama dili kullanılarak geliştirilmiştir. Öncelikle veri setinin sınıflandırmaya hazır hale getirilmesi amacıyla bazı veri ön işleme adımları kullanılmıştır. Verilerin normalleştirilmesi için min-max normalizasyon yöntemi kullanılmıştır. Daha sonra derin oto kodlayıcı ile boyut düşürme uygulanarak sınıflandırma için gereken yeni veri seti oluşturulmuştur. Oluşturulan veri setinin %80'lik kısmı eğitim, %20'lik kısmı ise test için kullanılmıştır. Çalışmada ikili sınıflandırma yapılmıştır. Sınıflandırmada kullanılan makine öğrenmesi yöntemlerinin başarısı Doğruluk (2), TPR (True Positive Rate) (3) ve TNR (True Negative Rate) (4) testlerine göre değerlendirilmiştir.

$$Doğruluk = \frac{TP+TN}{TP+FP+TN+FN} \quad (2)$$

$$TPR = \frac{TP}{TP+FN} \quad (3)$$

$$TNR = \frac{TN}{TN+FP} \quad (4)$$

*True Positive (TP): G*erçekte kötücül olan uygulamaların kötücül olarak tespit edilme sayısını;

False Negative (FN): Gerçekte kötücül olan uygulamaların normal olarak tespit edilme sayısını;

False Positive (FP): Gerçekte normal olan uygulamaların kötücül olarak tespit edilme sayısını;

True Negative (TN): Gerçekte normal olan uygulamaların normal olarak tespit edilme sayısını göstermektedir.

Doğruluk (accuracy), doğru sınıflandırılmış örnek sayısının toplam örnek sayısına olan oranıdır. TPR değeri, gerçekte kötücül olan uygulamaları kötücül olarak etiketleme olasılığını, TNR değeri ise gerçekte normal olan uygulamaları normal olarak etiketleme olasılığını göstermektedir.

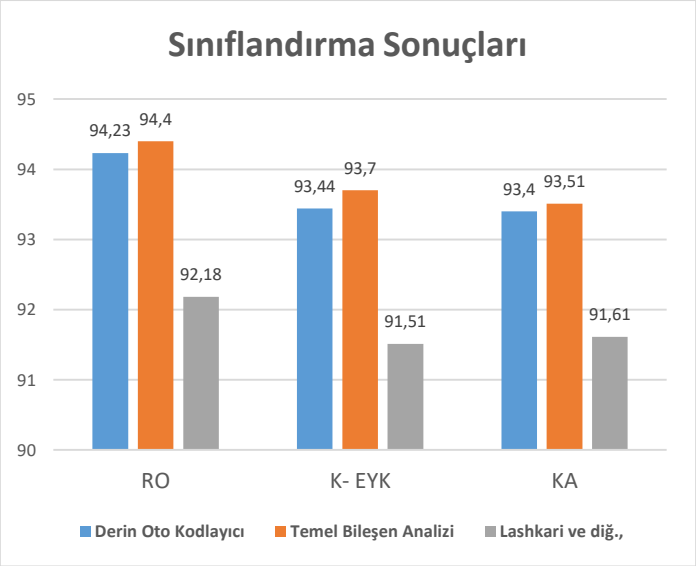
Çalışmada derin oto kodlayıcı ile yapılan boyut indirgeme işlemi, bu alanda sıkça kullanılan Temel Bileşen analizi ile de gerçekleştirilmiş ve bu analizlerden elde edilen değerlerle yapılan sınıflandırma başarısı sonuçları Tablo 1'de gösterilmiştir. Her iki metot ile gerçekleştirilen sınıflandırma sonuçlarının birbirine çok yakın olduğu görülmüştür.

Tablo 1. Sınıflandırma başarısı sonuçları

Kullanılan Sınıflandırma Yöntemi	Derin oto kodlayıcı (Deep autoencoder)			Temel Bileşen Analizi (Principal Component Analysis)		
	Doğruluk(%)	TPR (%)	TNR(%)	Doğruluk(%)	TPR (%)	TNR(%)
RO	94.23	85.47	97.22	94.40	86.00	97.27
K- EYK	93.44	82.00	97.35	93.70	83.18	97.30
KA	93.40	84.26	96.53	93.51	84.12	96.73

Uygulamada elde edilen sonuçlar Lashkari ve arkadaşları tarafından yapılan çalışma sonuçları ile karşılaştırıldığında derin oto kodlayıcı veya temel bileşen analizi ile boyut düşürme uygulandıktan sonra yapılan sınıflandırma başarılarının daha yüksek olduğu görülmüştür. Doğruluk parametresine göre karşılaştırmalı sonuçlar Şekil 2'de gösterilmiştir.

Şekil 2. Model doğruluklarının karşılaştırmalı analizi



TARTIŞMA VE SONUÇLAR

Android işletim sistemi, yaygın kullanımı ve gelişmiş özellikleri nedeniyle kötü amaçlı yazılım geliştiricileri tarafından en çok hedeflenen işletim sistemidir. Bu nedenle kötü amaçlı yazılımları tespit etmek siber güvenlik alanında ele alınması gereken önemli bir problemdir. Bu çalışmada kötücül yazılım uygulamalarının tespiti için derin oto kodlayıcı tabanlı bir yaklaşım önerilmiştir. Makine öğrenmesi yöntemi kullanılarak sınıflandırma yapılırken karşılaşılan en büyük problemlerden birisi öz nitelik seçimidir. Çünkü öz nitelik sayısı artıkça, işlem yapma zamanının artması ya da öğrenme aşamasında

gereksiz özniteliklerin uygulamaya dâhil edilmesi sebebiyle başarı oranının azalması gibi sorunlar meydana gelebilmektedir. Deneysel sonuçlar incelendiğinde, oto kodlayıcılı derin öğrenme yaklaşımı veya temel bileşen analizi ile boyut düşürme işlemi yapıldıktan sonra sınıflandırma yapıldığında başarı oranlarının arttığı görülmüştür.

KAYNAKÇA

Albright, S. C., Winston, W. L., & Zappe, C. (2006). Data Analysis & Decision Making, Üçüncü Baskı, Australia: Thomson South-Western.

Alshahrani, H., Mansourt, H., Thorn, S., Alshehri, A., Alzahrani, A., & Fu, H. (2018). DDefender: Android application threat detection using static and dynamic analysis. *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 1-6.

Archer, K.J., & Kimes, R.V. (2008). Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, 52, 2249-2260.

Baldi, P. (2011). Autoencoders, Unsupervised Learning and Deep Architectures. *In Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop* - Volume 27 (pp. 37–50). JMLR.org. Retrieved from http://dl.acm.org/citation.cfm?id=3045796.3045801

Bengio, Y. (2009). Learning Deep Architectures for AI. Found. *Trends Mach. Learn.*, 2(1), 1–127. https://doi.org/10.1561/2200000006

Breiman L., (2001). Random forests, machine learning, *2001 Kluwer Academic Publishers*, 45(1), 5-32.

CICAAGM Veri Seti, (2017). https://www.unb.ca/cic/datasets/android-adware.html. [Erişim Tarihi: 06.05.2019].

He, N., Wang, T., Chen, P., Yan, H., & Jin, Z. (2018). An Android Malware Detection Method Based on Deep AutoEncoder. AICCC.

International Data Corporation Media Center Raporu, https://www.idc.com/promo/smartphone-market-share/os [Erişim Tarihi: 11.06.2019].

John, T.S., Thomas, T., & Uddin, M.M. (2017). A Multifamily Android Malware Detection Using Deep Autoencoder Based Feature Extraction. *In 2017 Ninth International Conference on Advanced Computing (ICoAC).*

Lashkari, A.H., Kadir, A.F., González, H.G., Mbah, K.F., & Ghorbani, A.A. (2017). Towards a Network-Based Framework for Android Malware Detection and Characterization. *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, 233-23309.

Li, W., Wang, Z., Cai, J., & Cheng, S. (2018). An Android Malware Detection Approach Using Weight-Adjusted Deep Learning. *2018 International Conference on Computing, Networking and Communications (ICNC)*, 437-441.

Naway, A., & Li, Y. (2019). Android Malware Detection Using Autoencoder. ArXiv, abs/1901.07315.

Tatlıdil, H. (1996). Uygulamalı Çok Değişkenli İstatiksel Analiz Ankara: Cem Web Ofset Ltd.

Vinayakumar, R., Soman, K.P., Poornachandran, P., & Kumar, S.S. (2018). Detecting Android malware using Long Short-term Memory (LSTM). *Journal of Intelligent and Fuzzy Systems*, 34, 1277-1288.

Wang, W., Zhao, M., &Wang, J. (2018). Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-018-0803-6

Xu, K., Li, Y., Deng, R.H., & Chen, K. (2018). DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks. *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 473-487.

Yousefi-Azar, M., Varadharajan, V., Hamey, L., & Tupakula, U.K. (2017). Autoencoder-based feature learning for cyber security applications. *2017 International Joint Conference on Neural Networks (IJCNN)*, 3854-3861.

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

Yuxin, D., & Siyi, Z. (2017). Malware detection based on deep learning algorithm. *Neural Computing and Applications*, 31, 461-472.

Zhou, Q., Feng, F., Shen, Z., Zhou, R., Hsieh, M., & Li, K. (2018). A novel approach for mobile malware classification and detection in Android systems. *Multimedia Tools and Applications*, 78, 3529-3552.

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