T.C.
KIRŞEHİR AHİ EVRAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İLERİ TEKNOLOJİLER ANABİLİM DALI

BİLGİSAYAR SALDIRILARI İÇİN TAHMİN, TESPİT VE KARŞI KOYMA YÖNTEMLERİ

Ruaa Hussein HYARA

YÜKSEK LİSANS TEZİ

KIRŞEHİR / 2022

T.C.
KIRŞEHİR AHİ EVRAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İLERİ TEKNOLOJİLER ANABİLİM DALI

BİLGİSAYAR SALDIRILARI İÇİN TAHMİN, TESPİT VE KARŞI KOYMA YÖNTEMLERİ

Ruaa Hussein HYARA

YÜKSEK LİSANS TEZİ

DANIŞMAN

Dr. Öğr. Üyesi Şekip Esat HAYBER

KIRŞEHİR / 2022

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Ruaa Hussein HYARA

20.04.2016 tarihli Resmî Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü Teze, Kırşehir Ahi Evran Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Yüksek lisansa başlamamda ve yüksek lisans ders ve tez yazım sürecinde kendisini tanıdığım günden bu yana gösterdiği sakin ve sabırlı hali ile her zaman bana örnek olmasının yanı sıra bir bilim adamının nasıl çalışması gerektiğini kendisinden öğrendiğim değerli danışmanım Dr. Öğr. Üyesi Şekip Esat HAYBER'e teşekkür ederim. Çalışmalarım boyunca maddi manevi destekleriyle beni hiçbir zaman yalnız bırakmayan, beni bugünlere getiren aileme de sonsuz teşekkürler ederim.

Temmuz, 2022 Ruaa Hussein HYARA

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar	Açıklama
YSA	Yapay Sinir Ağları
DVM	Destek vektör makinesi
TP	Gerçek Pozitif
FP	Gerçek Negatif
FP	Yanlış Pozitif
FN	Yanlış Negatif
ACC	Doğruluk
TPR	Duyarlılık
SPC	Özgüllük

ÖZET

YÜKSEK LİSANS TEZİ

BİLGİSAYAR SALDIRILARI İÇİN TAHMİN, TESPİT VE KARŞI KOYMA YÖNTEMLERİ

Ruaa Hussein HYARA

Kırşehir Ahi Evran Üniversitesi
Fen Bilimleri Enstitüsü
İleri Teknolojiler Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Şekip Esat HAYBER

Son teknolojik yenilikler ve dünya çapındaki çok miktarda mevcut veri, ağ sistemlerine karşı siber saldırıların artmasına neden oldu. Bu saldırıların önde gelen çeşitlerinden biri kötü amaçlı yazılım ve aynı zamanda malware diye bilinir. İnternete bağlı bilgisayar sistemleri için kötü amaçlı yazılım analizi ve önleme yöntemleri giderek daha fazla gerekli hale geliyor. Önceki araştırmacılar tarafından birçok verimli yöntem önerilmiştir, ancak bu yöntemler statik ve dinamik analizlere dayalı olduğu için kaçınılmaz olarak karmaşık süreçler gerektirir. Bu çalışmada, makine öğrenmesine dayalı yeni uygulama geliştirildi ve geliştirilen uygulamanın amacı malware saldırılarını tespit etmektir. Sunulan teknik önceki çalışmalarda sunulan ve aynı veri setini kullanan çalışmalar ile karşılaştırıldı ve daha üstün sonuçlar gösterdi. Sunulan yöntem Hassasiyet (presicion), Geri Çağırma (recall), Doğruluk (accuracy) ve tüm bu parametrelerde 100% performans gösterdi. Dahası elde edilen sonuçlar karmaşık matris (confusion matrix) ile sunuldu.

Temmuz 2022, 55 Sayfa

Anahtar Kelimeler: KNN, Makine öğrenme, Kötü amaçlı yazılım.

ABSTRACT

MASTER THESIS

PREDICTION, DETECTION, AND COUNTERMEASURES FOR COMPUTER ATTACKS

Ruaa Hussein HYARA

Kirsehir Ahi Evran University
Science and Engineering Institute
Advanced Technologies Department

Supervisor: Asst. Prof. Dr. Şekip Esat HAYBER

Recent technological innovations and the vast amount of data available worldwide have led to an increase in cyberattacks against network systems. One of the leading variants of these attacks is malware, also known as malware. Malware analysis and prevention methods are becoming increasingly necessary for internet-connected computer systems. Many efficient methods have been proposed by previous researchers, but since these methods are based on static and dynamic analysis, they inevitably require complex processes. In this study, a new application based on machine learning was developed and the purpose of the developed application is to detect malware attacks. The presented technique was compared to studies using the same data set as presented in previous studies and showed superior results. The presented method showed 100% performance in Precision (presicion), Recall (recall), Accuracy (accuracy) and all these parameters. Moreover, the results obtained are presented with a complex matrix (confusion matrix).

July 2022, 55 Pages

Keywords: KNN, Machine learning, Malware.

1. GİRİŞ

İnternet kullanımındaki ve kişisel bilgisayarlardaki son artış, siber suçluların İnternet kullanıcılarını yaygın ve zarar verici siber tehditlere maruz bırakmasını kolaylaştırdı. Kullanıcıların teşhir edilmesi, siber suçluların büyük çapta kâr elde etmesine veya zarar vermesine neden oldu. Kullanıcılar, kötü amaçlı yazılımların alt sınıflarını oluşturan virüsler, solucanlar, casus yazılımlar, reklam yazılımları ve fidye yazılımlarıyla karşı karşıya kaldı. Bu tehditler, e-posta kimlik avı, yazılım güvenlik açıkları, ücretsiz yazılımlar, e-posta ekleri dahil olmak üzere çeşitli saldırı vektörlerini kullanarak kullanıcı sistemlerini tehlikeye atar. Antivirüsler, kötü amaçlı yazılımdan koruma, ana bilgisayar tabanlı saldırı tespit sistemleri (HIDS) ve ağ tabanlı izinsiz giriş tespit sistemleri (NIDS) yaygın olarak kullanılan kötü amaçlı yazılım azaltma teknikleridir. Ancak, kötü amaçlı yazılımın dinamik yapısı ve artan karmaşıklığı nedeniyle, azaltmalar genellikle sürdürülebilir koruma sağlamak için yetersizdir. Kötü amaçlı dosyaların veya yürütülebilir dosyaların son kullanıcıya ulaşmadan algılanması anlamına gelen "proaktif koruma", BT sistemlerindeki kötü amaçlı dosyalardan kaynaklanan bulaşmaları önlemek için bir çözüm olabilir. Günlük ayarlarda, kötü niyetli faaliyetler gerçekleştiren dosyaları ele almak için sıklıkla "kötü amaçlı yazılım" terimini kullanırız. Bu faaliyetler, yetkisiz erişim veya kimlik doğrulama, ayrıcalık yükseltme ve hedef makine, kullanıcıları veya bileşenleri hakkındaki bilgilerin yetkisiz ifşası gibi kullanım güvenlik politikalarının ihlallerini içerir. Kötü amaçlı yazılım algılama kavramı, esas olarak kötü niyetli amaç oluşturmak için yürütülebilir dosyaların analiz edilmesiyle ilgili bir konu olarak bilinmektedir. Kötü amaçlı yazılımdan koruma yazılımının ortaya çıkmasından bu yana, bu yazılımı atlatmak için özel olarak tasarlanmış karmaşık kötü amaçlı yazılımlarda bir artış görülmüştür. Bu da daha gelişmiş algılama tekniklerine yönelik araştırmalara öncülük etmiştir. Kötü amaçlı yazılım analizi veya kötü amaçlı yazılım tespiti iki şekilde gerçekleştirilebilir: statik veya dinamik olarak. Statik Kötü Amaçlı Yazılım Algılama: Statik kötü amaçlı yazılım algılama, bir ikili dosyayı yürütmeden analiz etme sürecidir. Bu, dosyanın tamamen yayılmasını ve her bileşenin incelenmesini, tersine mühendislik yapmak için bir sökücü kullanılmasını veya akışını incelemek için montaj koduna dönüştürülmesini içerebilir [1]. Varsa yazılımın orijinal kaynak koduna da genişletilebilir [2]. Bu, genellikle tüm kötü amaçlı yazılımdan

koruma yazılımları tarafından kullanılan kötü amaçlı yazılımlara karşı ilk savunma hattıdır. Statik analiz genellikle bilinmeyen bir dosyayla uğraşırken ilk olarak gerçekleştirilir. İlk adım, ana bilgisayarda [3] yüklü antivirüs programı ile dosyayı manuel olarak taramaktır. Dosya zaten biliniyorsa, kendi başınıza çözmeye çalışmak için saatler harcamanın bir anlamı yoktur. (Öğrenme deneyimi hariç). Sistem antivirüs programına ek olarak, dosya, 43 farklı antivirüs programı kullanarak dosyayı tarayan VirusTotal gibi bir site üzerinden çalıştırılabilir. Aynı dosyayla başka birinin karşılaşıp karşılaşmadığını görmek için dosyanın karmasını hesaplamak ve çevrimiçi olarak aramak da yararlı olabilir. Dize analizi, dosya hakkında ipuçları almanın basit bir yoludur. Komut satırı seçenekleri, kullanıcı diyaloğu, şifreler, URL'ler e-posta adresleri, kitaplıklar ve işlev çağrıları gibi dosya bilgilerindeki tüm dizeleri listeleyerek bulunabilir [3]. 'Kötü amaçlı yazılım' teriminin genel olarak kötü amaçlı dosyalar olarak kavramsallaştırılmasına rağmen, kötü amaçlı kod parçaları genellikle dosyanın kendisinden ziyade bir dosyanın parçalarıdır. Başka bir deyişle, kötü amaçlı yazılım kod parçaları tipik olarak yürütülebilir dosyalara, yani yüklere sarılır. Sistem düzeyinde bir bakış açısından, bir yazılım kodunun kötü niyetli satırları, makine düzeyinde bir dizi talimatı çalıştıran temel birimlerdir. Başka bir deyişle, kötü amaçlı yazılım, talimatları kötü amaçlı amaçlar için çalıştıran komutları ifade eder [4]. Bu talimatlar, giriş-çıkış işlevleri için sistem çağrıları ve bilgisayar belleği ve dosya sistemlerini çalıştıran bir dizi işlev gerçekleştirebilir. Buna göre, kötü amaçlı yazılım tespitindeki en önemli zorluk, kötü amaçlı yazılımı içeren şüpheli dosyanın kötü amaçlı işlevselliğe sahip olması için bir dosyadaki kod parçalarını tanımlamaktır. Uygulamada, kötü amaçlı yazılım algılama yöntemleri, imza veritabanlarına ve YARA1 kurallarına dayanır [5]. İmza veritabanları, yeni karşılaşılan bir yürütülebilir dosyadan oluşturulan bir imzayla eşleştirmek için kullanılır [6]. Bununla birlikte, kötü amaçlı yazılımın kendi kendini değiştirme yetenekleri, iyi huylu bir dosyayla karıştırılmaması için bu yöntemlerin algılama yeteneklerini sınırlar [7]. Kötü niyetli etkinlikleri kaçırmamak için, şüpheli dosyalarda daha fazla işlevsellik ifade eden kısımlar olan kod satırlarına odaklanmak çok önemlidir. Ancak, yürütülebilir dosyaların kaynak kodları genellikle derlenmiş biçimde bulunmadığından, derleme yönergeleri, şüpheli bir dosyadaki kötü amaçlı işlevleri ortaya çıkarmak için en iyi adaydır. Araştırmacılar ve uygulayıcılar, üç ana dalda sınıflandırılabilecek çeşitli teknikler [8, 9]. Dinamik analiz, dosyayı kötü amaçlı yazılım tespiti için yürütürken, statik analiz, yürütülebilir dosyayı çalıştırmadan tüm dosyayı tarayarak kötü amaçlı yazılımları tespit etmeyi amaçlar. Statik analizin, gizleme ve dinamik kod yükleme gibi kötü niyetli deformasyon tekniklerine direnmede dinamik

analize karşı bazı dezavantajları vardır. Ancak, daha az kaynak tüketir, kötü amaçlı yazılımları verimli bir şekilde tanımlar ve son kullanıcılara veya sunuculara ulaşmadan önce etkisini azaltır. Ayrıca, statik analiz ölçeklenebilir ve toplu bilinmeyen kötü amaçlı yazılım algılamasıyla karşılaşıldığında kullanılabilir ve yürütülebilir dosyanın tüm olası yürütme yollarını geçebilir [10, 11].

1.1. MOTİVASYON VE KATKILAR

Bu çalışmada, birden fazla bilimsel motivasyon vardır. Bunların başında son yıllarda internet üzerinden yapılan işlemlerin (internet üzerinden alışveriş yapmak, banka işlemleri vs.) artması gelir. Üstelik, eski tekniklerin yeni oluşturulan kötü yazılımlara karşı yetersiz kalması. Öte yandan, makine öğrenme tekniklerinin birden fazla alanda çok verimli sonuçlar vermesidir. Bu nedenle, bu çalışmada, makine öğrenmenin denetimli ve denetimli tekniklerinin tüm özelliklerini kullanarak yeni ve verimli bir güvenlik uygulaması sunuldu. Ayrıca, bu araştırmada birden fazla katkı sunulmuştur:

- Yeni ve verimli bir güvenlik uygulaması geliştirildi, bu uygulamada kullanılan teknikler daha önce kullanılmamıştır.
- Son olarak, en yakın komşu (KNN) hiperparametrelerini arama algoritması kullanarak en optimumu bulunmuştur. KNN hiperparametreleri optimum şekilde olduğu zaman en iyi sonucu verir.

2. İLGİLİ ÇALIŞMALAR

Bu bölümde önceden araştırmacılar tarafından yapılan çalışmalar detaylı bir şekilde sunulmuştur.

Sewak ve diğerleri [12], kötü amaçlı yazılım tespiti için bir makine öğrenme yöntemi olan rastgele orman (RF) ile Derin Sinir Ağı (DNN) algoritmalarıyla karşılaştırmıştır. Bu araştırma sonucu, RF 'nin DNN mimarisinden daha iyi performans gösterdiğini göstermiştir. Kullanılan veri seti, DNN sınıflandırmasının sonucunu etkilemiş olabilir ve sonuç olarak derin öğrenme yaklaşımları, makine öğrenmesi yaklaşımlarına kıyasla her zaman daha iyi doğruluk sağlar. Makine öğrenimi ile derin öğrenme arasındaki en önemli farklardan biri, derin öğrenme yönteminde gerekli özellik çıkarımının olmamasıdır. Başka çalışmada,

Li Chen [13] statik kötü amaçlı yazılım analizi yöntemiyle ikili dosyaları vektörleştirdi ve ardından vektörleri görüntülere dönüştürdü. Çalışmasında, bilgisayarla görme alanında yaygın olarak kullanılan Transfer Öğrenme Yöntemi sayesinde Derin Sinir Ağı (DNN) algoritmasının eğitim süresinin kısaldığı ifade edildi.

Dai ve diğerleri [14], bellek dökümü verileri kötü amaçlı yazılım tespiti için yeterli bilgi sağlayacağından, bellek dökümüne dayanıyordu. Kötü amaçlı yazılım Cuckoo sanal alanında yürütüldü ve sanal alan ortamında çalışan kötü amaçlı yazılım, Microsoft tarafından geliştirilen Procdump adlı sysinternal aracı kullanarak bellek dökümü verilerini aldı. Elde edilen bellek dökümü verileri gri tonlamalı görüntülere dönüştürüldü; bu dönüştürme işleminde, döküm verilerinin boyutuna göre genişlik 2048 veya 4096 olarak ayarlandı ve yükseklik dosya boyutuna göre değişmektedir. Görüntüler, doku özelliklerini HOG aracılığıyla çıkardı. Sabit uzunluklu HOG öznitelik vektörleri Çok Katmanlı Algı (MLP), RF ve K-NN sınıflandırıcı (K = 3, K = 5) ile sınıflandırılmış ve en yüksek doğruluk değeri olan %95,2 MLP ile elde edilmiştir.

Halim, Abdullah ve Ariffin [15] çalışmalarında eğitim ve doğrulama için Drebin Veri Kümesini kullandılar ve veri kümesi toplam 179 aileden 129,013 kötü amaçlı yazılımın özelliklerini içermektedir. Bunlar 8 özelliktir: donanım bileşeni, istenen izinler, uygulama bileşenleri, filtrelenmiş amaçlar, kısıtlanmış API çağrıları, kullanılan izinler, şüpheli API çağrıları ve ağ adresleri. BOW, özellik çıkarımı için kullanıldı, ancak BOW yaklaşımı,

kötü amaçlı yazılım modellerindeki uzamsal ve dizi bilgilerini sildiği için bilgi kaybına ve kaba indekslemeye yol açmaktadır. Bu sorunun üstesinden gelmek için Uzun Kısa Süreli Bellek (LSTM) ve Evrişimsel Sinir Ağı (CNN) birleştirilmesi önerilmiştir. Çalışmada Çok Katmanlı Algı (MLP), CNN, LSTM, CNN-LSTM ve LSTM-CNN gibi 5 farklı Sinir Ağı modeli kullanılmış ve sonuçlara göre LSTM-CNN %98,53 doğrulukla üstün performans gösterirken, CNN %87,91 ile en düşük doğruluk değeri. LSTM-CNN birleşik sinir ağı modelinin üstün performansı, hibrit şema sinir ağlarının daha doğru olduğunu göstermektedir.

Dai ve diğerleri, çalışmalarında [16], mevcut dinamik kötü amaçlı yazılım tespit yöntemleri kaçırma saldırılarına karşı yeterli olmadığı için, çok özellikli bir dinamik kötü amaçlı yazılım analiz yöntemi önermiştir. Trough Cuckoo Sandbox, API çağrı sırası, bellek dökümü verileri ve donanım performans sayacı (HPC) özellikleri çıkarılmıştır. API dizisini bir özellik vektörüne dönüştürmek için word2vec yöntemi kullanılmıştır. Bellek dökümü verileri png dosya formatında gri tonlamalı görüntülere dönüştürülmüş ve bellek dökümü verileri farklı boyutlarda olabileceğinden gri tonlamalı görüntüler için bikübik interpolasyon uygulanmış ve böylece tutarlı boyutlarda gri tonlamalı görüntüler oluşturulmuştur. Çalışmada kullanılan son özellik donanım düşük seviye olup, CPU'nun Modele Özgü Kayıtları (MSR) incelenmiştir. Çalışma sonucunda doğruluk %96,9 oranında sağlanmıştır. Elde edilen sonuçlara göre, çok özellikli yöntemin kötü amaçlı yazılım kaçırmayla mücadelede yeterince etkili olduğu gözlemlenmiştir.

Türker ve diğerleri [17] çalışmalarında, zararlı yazılımların ailelerine göre sınıflandırılması ile ilgili bir çalışma yapmıştır. Android uygulamaların istedikleri izinler ile API çağrılarını öznitelik olarak kullanmışlar ve makine öğrenme teknikleri ile sınıflandırma yapmıştır. APKTool kullanarak öznitelik çıkarımında bulunmuştur. Python Scikit- Learn kütüphanesi kullanarak makine öğrenme algoritmaları ile sınıflandırma yapmıştır. Yaptığı testler sonucunda aldığı doğruluk değerleriyle makro ortalamalı duyarlılık, hassasiyet ve F1 skoru değerlerini ayrı ayrı değerlendirmiş ve yorumlamıştır.

Alzaylaee ve diğerleri [18] çalışmalarında, dinamik analiz yöntemi ile zararlı Android uygulamalarını tespit eden DL-Droid adını verdikleri bir çalışma yapmışlardır. Çalışmada Derin Öğrenme metodunu kullanmışlardır. Alzaylaee vd. (2016) tarafından geliştirilen DynaLog isimli uygulamayı kullanarak dinamik öznitelikleri çıkarmışlardır. Gerçek cihazlarda 11,505 zararlı uygulama, 19,620 zararsız uygulama ile testler yapmışlardır. DL-

Droid'in geleneksel makine öğrenme tekniklerini geride bırakan sırasıyla %97,8 (yalnızca dinamik özelliklere sahip) ve %99,6 tespit oranına (dinamik + statik özellikli) tespit oranına eriştiğini belirtmişlerdir.

Dai ve diğerleri [14], bellek dökümü verileri kötü amaçlı yazılım tespiti için yeterli bilgi sağlayacağından, bellek dökümüne dayanıyordu. Kötü amaçlı yazılım Cuckoo sanal alanında yürütüldü ve sanal alan ortamında çalışan kötü amaçlı yazılım, Microsoft tarafından geliştirilen Procdump adlı sysinternal aracı kullanarak bellek dökümü verilerini aldı. Elde edilen bellek dökümü verileri gri tonlamalı görüntülere dönüştürüldü; bu dönüştürme işleminde, döküm verilerinin boyutuna göre genişlik 2048 veya 4096 olarak ayarlandı ve yükseklik dosya boyutuna göre değişiyor. Görüntüler, doku özelliklerini HOG aracılığıyla çıkardı. Sabit uzunluklu HOG öznitelik vektörleri Çok Katmanlı Algı (MLP), RF ve K-NN sınıflandırıcı (K = 3, K = 5) ile sınıflandırılmış ve en yüksek doğruluk değeri olan %95,2 MLP ile elde edilmiştir. Son yıllarda kötü amaçlı yazılım analizi için yapılan birtakım araştırmalar donanım özelliklerinden yararlanmaktadır. Donanım özelliklerine örnek olarak Hardware Performance Counter (HPC) verilebilir. Bu araştırmanın sonuçları, HPC [19] ile yapılan çalışma ile karşılaştırılmış ve bellek dökümü verilerinin kullanılmasının daha yüksek sonuç verdiği; HPC yönteminin kötü amaçlı yazılım davranışını tam olarak açıklamadığı sonucuna varılmıştır.

Halim, Abdullah ve Ariffin [15] çalışmalarında eğitim ve doğrulama için Drebin Veri Kümesini kullandılar ve veri kümesi toplam 179 aileden 129,013 kötü amaçlı yazılımın özelliklerini içeriyor. Bunlar 8 özelliktir: donanım bileşeni, istenen izinler, uygulama bileşenleri, filtrelenmiş amaçlar, kısıtlanmış API çağrıları, kullanılan izinler, şüpheli API çağrıları ve ağ adresleri. BOW, özellik çıkarımı için kullanıldı, ancak BOW yaklaşımı, kötü amaçlı yazılım modellerindeki uzamsal ve dizi bilgilerini sildiği için bilgi kaybına ve kaba indekslemeye yol açar. Bu sorunun üstesinden gelmek için Uzun Kısa Süreli Bellek (LSTM) ve Evrişimsel Sinir Ağı (CNN) birleştirilmesi önerildi. Çalışmada Çok Katmanlı Algı (MLP), CNN, LSTM, CNN-LSTM ve LSTM-CNN gibi 5 farklı Sinir Ağı modeli kullanılmış ve sonuçlara göre LSTM-CNN %98,53 doğrulukla üstün performans gösterirken, CNN %87,91 ile en düşük doğruluk değeri. LSTM-CNN birleşik sinir ağı modelinin üstün performansı, hibrit şema sinir ağlarının daha doğru olduğunu gösterdi.

Dai ve diğerleri, çalışmalarında [16], mevcut dinamik kötü amaçlı yazılım tespit yöntemleri kaçırma saldırılarına karşı yeterli olmadığı için, çok özellikli bir dinamik kötü

amaçlı yazılım analiz yöntemi önermiştir. Trough Cuckoo Sandbox, API çağrı sırası, bellek dökümü verileri ve donanım performans sayacı (HPC) özellikleri çıkarıldı. API dizisini bir özellik vektörüne dönüştürmek için word2vec yöntemi kullanıldı. Bellek dökümü verileri png dosya formatında gri tonlamalı görüntülere dönüştürülmüş ve bellek dökümü verileri farklı boyutlarda olabileceğinden gri tonlamalı görüntüler için bikübik interpolasyon uygulanmış ve böylece tutarlı boyutlarda gri tonlamalı görüntüler oluşturulmuştur. Çalışmada kullanılan son özellik donanım düşük seviye olup, CPU'nun Modele Özgü Kayıtları (MSR) incelenmiştir. Çalışma sonucunda doğruluk %96,9 oranında sağlanmıştır. Elde edilen sonuçlara göre, çok özellikli yöntemin kötü amaçlı yazılım kaçırmayla mücadelede yeterince etkili olduğu gözlemlendi. İlk kötü amaçlı yazılım tespit yöntemleri tamamen yazılım tabanlı ve ilkeldi ve son araştırmalarda kötü amaçlı yazılım tespiti için donanım destekli yöntemler önerildi.

Xu ve diğerleri [20] donanım destekli bir kötü amaçlı yazılım algılama yöntemi önerdi. Çalışmaları, kötü amaçlı yazılımın çalıştırıldığında bellekte parmak izi bıraktığı gerçeğine dayanıyor. Çalışmada çekirdek rootkit tespiti için bellekteki sistem çağrılarının kalıpları kullanılmış ve %99 tespit oranı elde edilmiştir.

Yang ve diğerleri [21], çalışmalarında zararlı davranışları karakterize etmek, zararlı uygulamaları tespit etme oranını artırmak, en alakalı ve etkili özellikleri çıkarmak amacıyla DroidWard adlı bir dinamik analiz yöntemi önermişlerdir. Öznitelikleri 15 ayrı kategoride toplamışlardır 9 tanesini klasik yöntemlerden ve 6 tanesini de yeni olarak belirtmişlerdir. Dinamik özellikleri çıkarırken DroidBox kullanmışlardır. Ancak DroidBox tarafından izlenen verilerin sınırlı olduğunu belirtmişler ve DroidBox'un kaynak kodlarında değişiklik yaparak Monkeyrunner ile hassas API çağrılarını simüle etmişlerdir. 258 zararlı uygulama ve 408 zararsız uygulamadan oluşan bir veri seti ile Destek Vektör Makineleri, Karar Ağacı, Rastgele Orman makine öğrenme tekniklerini kullanmışlardır. Destek Vektör Makineleri yöntemi ile %98,54 doğru sınıflandırma sonucu elde etmişlerdir.

Sugunan ve diğerleri [22], statik ve dinamik analiz kullanarak zararlı ve zararsız uygulamaların davranışları hakkında karşılaştırmalı bir çalışma yapmışlardır. Statik özellikleri APKtool kullanarak çıkarmışlar ve dinamik özellikleri ise Droidbox APIMonitor kullanarak çıkarmışlardır. Çalışmada 150 zararlı uygulama ve 200 zararsız uygulama kullanmışlardır. Statik ve dinamik analizdeki özelliklerin birlikte kullanımı ile makine öğrenme algoritmaları RF, SVM, J48 ve Naive Bayes gibi yöntemlerle yaptıkları

testlerin sonuçlarının statik ve dinamik özelliklerden daha başarılı olduğunu belirtmişlerdir. Peynirci (2018), çalışmasında izinler, API çağrılarını, katar (string) özelliklerini kullanarak statik analiz yapmıştır. Belge sıklığı tabanlı bir model ile sınıflandırma yapmıştır. İki farklı veri seti kullanarak testler yapmıştır. Bunlar MalGenome ve AndroZoo veri setidir. Birçok makine öğrenmesi yöntemi ile WEKA'da testler yapıp sonuçlarını paylaşmıştır.

Wu ve diğerleri [23], Deep Belief Network (DBN) ve ağırlıklı SVM yöntemlerinin beraber kullanıldığı hibrit bir sistem önermişlerdir. Önerilen yöntemde özellik çıkarma işlemi DBN yöntemiyle, özelliklerin iyileştirilmesi ve sınıflandırma işlemi ağırlıklı SVM yöntemiyle gerçekleştirilmiştir. Çalışma sonucunda KDDTest+ verisinde %85,73, KDDTest-21 verisinde ise %70,25 başarı oranı elde edilmiştir.

Liu ve diğerleri [24], CNN tabanlı bir STS önermişlerdir. Çalışmada modelin daha iyi sınıflandırma yapabilmesi için eğitim veri setinde az bulunan saldırı türlerine veri arttırma yapılmıştır. Ayrıca CNN'in etkinliğini arttırmak için özellik vektörleri 2 boyutlu (2D) piksel tabanlı görüntülere dönüştürülmüştür. Sonuç olarak KDDTest+ verisinde %81,30, KDDTest-21 verisinde ise %64,70 başarı oranı elde edilmiştir.

Jiang ve diğerleri [25], C NN i le B LSTM'yi birleştiren derin hiyerarşik ağ modelini önermişlerdir. Bu çalışmada da veri setindeki saldırı türleri arasındaki örnek sayısı dengelenmiştir. Veri setindeki çoğunluk örneklerini azaltmak için One-Side Selection (OSS) ve azınlık örneklerini artırmak için SMOTE tekniği kullanılmıştır. Önerilen yöntem ile KDDTest+ verisinde %83,58 başarı oranı elde edilmiştir.

Tama ve diğ. [26], makine öğrenmesi temelli bir STS önermişlerdir. Çalışmada ilk olarak eğitim veri setinin özellik boyutunu azaltmak için, parçacık sürüsü optimizasyonu, karınca kolonisi algoritması ve genetik algoritma gibi üç yöntemden oluşan bir hibrit özellik seçim tekniği kullanılmıştır. İkinci aşamada ise RF ve torbalama olan iki seviyeli bir sınıflandırıcı önerilmiştir. Önerilen modelle KDDTest+ verisinde %85,79, KDDTest-21 verisinde ise %72,52 başarı oranı elde edilmiştir.

Ieracitano ve diğerleri [27], istatistiksel analiz ve otomatik kodlayıcı tabanlı bir derin öğrenme mimarisi önermişlerdir. Önerilen model eğitilmeden önce veri setindeki aykırı değerler kaldırılmıştır. İkili sınıflandırmada normal ve anormal kategorilerin, çoklu sınıflandırmada ise DoS, R2L ve probe saldırı türlerinin tespiti yapılmıştır. KDDTest+

verisinde ikili sınıflandırmada için %84,21, çoklu sınıflandırma için %87 başarı oranı elde edilmiştir.

2.1. Kötü Amaçlı Yazılım Türleri

Önde gelen anti-virüs şirketlerinden biri olan McAfee'ye göre (McAfee, 2020), "Kötü Amaçlı Yazılım" veya kısaca "Kötü Amaçlı Yazılım" olarak bilinen "Kötü Amaçlı Yazılım", herhangi bir programlanabilir cihaza, sunucuya veya ağa zarar vermek veya bunlardan yararlanmak için tasarlanmış koddur. Bazıları kullanıcı müdahalesi gerektiren, bazıları saklanmak için belirli bir süre bekleyen, bazıları zaman içinde kendilerini diğer ağ bileşenlerine dağıtan birçok farklı kötü amaçlı yazılım türü vardır.

Kötü amaçlı yazılımın birkaç tanımı vardır, Kötü amaçlı yazılım, bilgisayarınızda çalışan ve sisteminizin bir saldırganın yapmasını istediği bir şeyi yapmasını sağlayan bir dizi talimattır. Bu tanıma göre, bu tezde kullandığımız için yürütülebilir olması gerekmez. Donanıma uygulanabileceği için yazılım olması bile gerekmez. Tanımın ikinci kısmı, çok çeşitli senaryolara atıfta bulunabilir. Saldırgan yalnızca örneğin sistemdeki çok sayıda değerli dosyayı silerek zarar vermek isteyebilir. Veya amaç para olabilir, bu nedenle dosyalar şifrelenir ve kurbandan şifre çözme anahtarı için ödeme yapması istenir. Ayrıca, kredi kartı numaraları gibi bilgilerin gizlice dinlenmesi veya çalınması da bir saldırıya neden olabilir [28].

Kötü amaçlı yazılımın tanımı geniş olduğundan, kötü amaçlı yazılım türlerini ve sınırlarını tanımlamak kolay değildir. Birçoğunun sistem başladığında kullanıcıdan gizlenmeyi tercih etmesi, algılanmamak için kullanıcının izni olmadan işletim sistemine entegre olması gibi birçok ortak özelliği vardır. Bu benzerliklere ek olarak, literatürde birçok farklı kötü amaçlı yazılım türü, tanımı ve sınıflandırmasına yol açan birçok farklılık da bulunmaktadır. Bu farklılıklara göre, üzerinde anlaşmaya varılan bazı kötü amaçlı yazılım türleri şunlardır:

2.1.1. Virüs

Bilgisayarlara uygulandığında, "virüs" terimi ilk olarak 1987'de Cohen tarafından tanıtıldı. Virüs, karlı programlar [29]. Diğer programlara eklenirler ve ana bilgisayar programlarına sahip olmanız için arka plan uygulamalarına sahip olmaları gerekir. Üç alandan oluşur: bulaşma eğilimi, tetikleyici ve şarj. Bilgisayar, bilgisayarın bir "reprodüksiyonu" veya biçimidir. İyileşmeyi artırmak veya tamamlamak için şifa. İkincisi kötü bir faaliyettir.

Birkaç virüs türü vardır. Ayrıca, üniversitede gördüğünüz dosyalar ve makrolar gibi virüsleri nasıl gizleyeceksiniz:

a. Şifreli virüs: Bu virüs türü, virüsün geri kalanını rastgele bir anahtarla şifreler. Başka bir tuş varken kopyalama sırasında sabit bir desen gözlemlenemez.

b. Görünmez Virüs: Ana Anahtarlar, fark edilmeyen bir saklanma yeridir. Belki de sadece iyi bir program gibi. İndirimlerden G/Ç rutinlerine kadar, başka birinin diskinin başına gelebilir, sizin başınıza gelebilir, giderek daha da büyüyebilir.

c. Polimorfik virüs: virüs versiyonu için farklı imzalara sahip bit desenleri üretecektir. Gizli virüste olduğu gibi, hedef henüz belirlenmedi.

d. Metamorfik virüs: Polimorfik virüs içeren bir proje benzerdir, ancak görünüş olarak da farklıdır. Bu, yenilerini takmayı daha da zorlaştırır.

2.1.2. Solucan (Worm)

Virüsün aksine bir solucan, diğer sistem cihazlarına bulaşmak için insan yardımına veya etkileşme ihtiyaç duymayan kötü amaçlı yazılım türüdür. Potansiyel istismarları tahmin ederler ve bulaşacak mühendisliği keşfederler. Ağlar arasında kolayca yayılabilir ve diğer sistemlere bulaşabilir. Aslında asıl amacı ağ üzerinden yayılmak ve ağı ele geçirmek, arkadaşlarına sızmaktır. Virüslerin ayırt edici özelliği, belirli bir cihazla değil, bir ağ veya sistemle ilişkili olmalarıdır [30].

2.1.3. Truva atı (Trojan)

Truva atı, bir bilgi işlem sistemine sızan ve fark edilmeden kalmaya çalışan ve bu esnada değerli kaynaklarınıza erişen bir tür kötü amaçlı koddur. Ancak, bu kötü amaçlı yazılım yalnızca bir cihazda engellenir ve engellenmez. Bugün Çanakkale olarak bilinen antik kent, adını ünlü Truva'dan almıştır. Bu karşılaştırmadan da anlaşılacağı gibi Truva atlarının en önemli özelliği, bu kötü amaçlı yazılımın gizlice hedef sisteme sızması ve değerli kaynaklara gizlice erişen ve gerektiğinde onları yeniden yönlendiren bir yazılım olmasıdır [31].

2.1.4. Reklam yazılımı (Adware)

Adware, kullanıcının rızası olmadan kullanıcının zararına reklam veren bir tür kötü amaçlı yazılımdır. Bu tür kötü amaçlı yazılımlar, reklamlar aracılığıyla normal ve kullanışlı bir kullanıcı deneyimini kesintiye uğrattığı, ancak sisteme ciddi zararlar vermediği ve ağ üzerindeki diğer cihazlara yayılmadığı için "en zararsız kötü amaçlı yazılım türü" olarak adlandırılabilir. Bazı saygın ücretsiz yazılımlar yükleme sırasında başka yazılımları yüklemeye çalıştığından, bu tür kötü amaçlı yazılımlar yalnızca agresif reklam olarak kabul edilebilir [32].

2.1.5. Casus yazılım (Spyware):

Casus yazılım, kendisini herhangi bir şekilde sisteme yükleyen ve kurban tarafından gerçekleştirilen eylemleri takip eden zararlı yazılım türüdür. Diğer kötü amaçlı yazılım türlerinin aksine, casus yazılım, etkinlikleri gizli olup olmadığına göre filtrelemez. Böylece ziyaret edilen web siteleri, izlenen videolar, indirilen resimler, genel yorumlar ve hatta oynanan oyunlar casus yazılımlardan etkilenebilir. Adından da anlaşılacağı gibi, bu tür kötü amaçlı yazılımlar arka planda çalışan casus yazılımlar olarak tanımlanabilir [33].

2.1.6. Kök kullanıcı takımı (Rootkit):

Bilindiği üzere tüm sistemler farklı hesap türlerine sahiptir ve genellikle bunlar, sistem üzerinde daha yüksek ayrıcalıklara sahip yönetici (süper- admin) kullanıcılar ve yönetici kullanıcılara kıyasla sistemde bazı belirli işleri yapabilen normal (yerel- local) kullanıcılardan oluşur. Rootkit adı verilen bu kötü niyetli programlar da bundan yararlanır ve ayrıcalıklarını resmi olarak yükseltmenin teknik bir yolunu arar. Böylece sistem içinde aldıkları idari yetkiler ile daha derin ve daha zarar verici eylemlerde bulunurlar [34].

2.1.7. İmzadan Kaçınma:

Tipik olarak, anti-virüs yazılımı, kötü amaçlı yazılımları tespit etmek için imza tabanlı bir yöntem kullanır. Kötü amaçlı yazılım yürütülebilir dosyasında bulunan talimatlar, kötü amaçlı yazılımı tanımlayan benzersiz bir imza elde etmek için ayrıştırılır ve bu daha sonra bilinen kötü amaçlı yazılım imzalarının büyük bir veritabanıyla karşılaştırılır [35]. Yaygın olarak kullanılan tüm montaj talimatları için düğümleri olan bir grafik kullandılar ve ardından kötü amaçlı yazılımları sınıflandırmak için bu grafiğin küçültülmüş bir sürümünü imza olarak kullandılar. Testlerine göre, bu algılama biçimi, grafikler daha büyük

olduğunda (daha büyük yürütülebilir dosyalar için) daha iyi genel algılama doğruluğu ile sonuçlandı.

2.1.8. Kod Gizleme:

Statik kötü amaçlı yazılım analizi, esas olarak semantik analiz ve hesaplamalı kaynak kodu tahmin edilebilir [36]. Kodun semantiğini gizlemek için opak sabitler kullanarak, onu kontrol etmek için bir programı ayrıştırmak için bir yöntem önerdi. Bu, statik kötü amaçlı yazılım analiz tekniklerinde belirsiz olan bir kusuru vurgular; anlamsal analiz, sabitleri gerçek zamanlı olarak programlamak için rastgele değiştirerek güncellenebilir. Bahsedilen yöntemde, kullanıcı veya süreç sırayla temizlik odaları oluşturur ve benzerlerini diğer adreslerdeki adreslerde depolar. Sabitler kodda oluşturulacağından bu, NP-hard tanıtımının girişinde tartışılmaktadır. Örneğin, kod bir 3SAT görevindeyse, o bölüm kodunun girişi her zaman statik bir sınıfta (kısaca 0) uyarlanabilirdir. Bu, 3SAT herhangi bir atamada çalışma zamanında 0 değeri döndürdüğünde oluşur. Bir insan okuyucu için bunu öğretmek kolay olsa da bir anlamsal analizcinin tüm bu olası çıktıları tanımlaması ve sonuçtan zamanının 0 olduğunu belirlemesi çok zordur çünkü bu bir polinom algoritmasıdır. Bu, Cristodorescu ve Jah tarafından, çoklu ikili şifreleme kullanılarak kod kullanımı ve güvenlik için bir pakette tartışılmıştır. Bu tür bir inceleme, geleceğin yörüngesi boyunca kolayca yakalanabilir, ancak dosyanın inceliği, yörüngenin analizinde ve dinamiklerde belirgindir. Preda ve diğerleri, orijinal kötü amaçlı yazılım tarafından karmaşık hale getirilmiş bir dizi ev aracı için anlambilim testi. Ayrıca NOP enjeksiyonu, boru hattı montajı ve yeniden düzenlemenin (NP sabit programlama veya benzer enjeksiyonlar) kötü amaçlı yazılım koduna dahil edilebileceğine dair değerlendirmelerini tartıştılar. Ancak yürütülemeyen uygulama tam olarak yürütülmemiştir. Arama analizi ve teşvikler konusunda olumsuz bir şeyler olacak. Bununla birlikte, bu amaçlar için kullanımı ucuzdur ve kötü amaçlı yazılımların yerleştirilebileceği ve kapsamlı bir şekilde analiz edilebileceği bir sanal alan uygulanmalıdır.

2.1.9. Yazılım Paketleme

Dosya paketleme, büyük yazılımları küçük, kompakt bir pakette bir araya getirirken kullanılan yaygın bir tekniktir [27]. Bu tür paketleme teknikleri genellikle kötü amaçlı yazılımın kolay tanımlanmasını potansiyel olarak engelleyebilecek bir tür şifreleme içerir. PolyPack adı verilen bu tür bir araç, paketleyicilerin virüsten ve kötü amaçlı yazılımdan kaçmak için etkili bir yöntem olduğunu kanıtlamak için özel olarak tasarlanmıştır [37].

Kendilerine sağlanan verileri bağımsız olarak paketleyen 10 paketleyici sağlarlar ve ardından paketlenmiş verileri 10 iyi bilinen anti-virüs tarayıcısı ile tararlar. En iyi sonucu alan paketleyici seçilir. Çalışmaları, bunun çoğu virüsten koruma yazılımına karşı kaçınma oranlarını 2,58 kat artırdığını ortaya koydu.

2.1.10. Tuş kaydedici (Keylogger)

Bir tuş kaydedici, klavyede yaptığınız her eylemi takip edebilen bir kötü amaçlı yazılımdır ve bu nedenle başta şifreler olmak üzere hassas ve kritik tüm verileriniz ortaya çıkar. Bu hassas veriler yalnızca parolaları değil, aynı zamanda tuş vuruşlarını, tıklamaları, ekran görüntülerini, anlık görüntüleri, ses kayıtlarını vb. içerir. Ayrıca olası hareketleri de kontrol edin. Ayrıca, tek tıklamayla bulaşma olasılığı göz önüne alındığında, bu tür kötü amaçlı yazılımlar oldukça tehlikelidir. Keylogger'ları tespit etmek kolay değildir, ancak bazı yazılımlar tarama sırasında diğer kötü amaçlı yazılımları tespit edebilir. Çünkü keylogger'lar, kullanıcı eylemlerini gerçekleştirmek ve işletim sistemleriyle doğrudan entegrasyon içinde çalışmak için işletim sistemleri tarafından sağlanan adli hizmetleri kullanır. Bu tür yazılımların tanınamayacağı düşünüldüğünde, biçimlendirme, yeni işletim sistemi kurma gibi onarılamaz hasarlara neden olacak işlemler yapılmadığı sürece hassas verilerin uzun süreler boyunca izlenebileceğini söylemeye gerek yok [38].

2.1.11. Fidye yazılım (Ransomware)

Fidye yazılımları, saldırı nedeniyle kullanılamaz hale getirdiği şifreli dosyaların şifresini çözmek veya saldırı yoluyla elde edilen hassas veya kişisel verileri yayınlamamak karşılığında kurbandan para talep eden kötü amaçlı yazılımlardır. Bu kötü amaçlı programlar, bilinen, hacklenmesi kolay algoritmalar yerine kendi şifreleme algoritmalarını kullanır. Bu nedenle, bu dosyaların kullanıcı veya başka herhangi bir yazılımın şifresini çözmesi kolay olmadığından, bir saldırganın boyun eğmesi gerekebilir. Gizlilik nedeniyle, kullanıcıdan geleneksel yöntemler yerine kripto para birimleri ile ödeme yapması isteniyor, bu yüzden polis bunu tespit edemiyor [39].

2.1.12. İndirici (Downloader)

İndiriciler, damlalıklar gibi farklı yapıya sahip başka bir kötü amaçlı yazılım türüdür. Kendileri kötü amaçlı yazılım değillerdir, ancak İnternet üzerinden belirli kaynaklardan kötü amaçlı yazılım indirmek için kodları vardır. Bu kötü amaçlı yazılımın geliştiricileri

için ana dezavantajı, İnternet bağlantısına bağımlılığıdır. İnternet bağlantısı olmadığında sisteme zarar veren herhangi bir ek yazılım indirilemez ve bu nedenle kötü amaçlı yazılım geliştiricileri tarafından yönlendirilen kötü niyetli eylemler gerçekleştirilemez. Kendileri kötü amaçlı yazılım olmadıkları için, onları normal kötü amaçlı yazılımlardan ayıran özelliklere sahiptirler ve bu nedenle geleneksel yöntemleri kullanan diğer kötü amaçlı yazılım türlerinden daha zor tespit edilirler [40].

2.1.13. Uzak yönetim araçları (Remote administration tools)

Uzaktan Yönetim Aracı olarak adlandırılan yazılımlar aslında, ağları veya cihazları uzaktan yönetmek için kullanılan resmi araçlardır. Bu resmi araçların makul güvenlik gereksinimleri gerektirmesi şaşırtıcı değildir. Ancak, kötü niyetli eylemler için bu araçlar, güvenlik koşulları atlanarak istismar araçları olarak kullanılabilir. "RAT" olarak kısaltılan bu yazılım, saldırganların, kötü amaçla kullanılması durumunda, kurbanın bilgisayarında yönetici ayrıcalıklarıyla komut dosyaları, kabuklar veya diğer kötü niyetli eylemleri çalıştırmalarına olanak tanır [41].

2.2. KÖTÜ AMAÇLI YAZILIM ANALİZ YÖNTEMLERİ

2.2.1. Statik Analiz Yöntemi

Statik analiz yöntemi, uygulamanın APK dosyasından çıkarılan kaynak kodun çeşitli tersine mühendislik yöntemleri kullanılarak analiz edilmesini içerir. Statik analiz yönteminde uygulamalar bir emülatör veya cihaz üzerinde çalıştırılmadan analiz edilir. APK dosyaları sıkıştırılmış biçimdedir. Winzip, Winrar gibi sıkıştırma programları ile açılabilir ve dex, manifest ve kaynak dosyalara erişebilir. Bu dosyaları görüntülendikten sonra okunabilir hale getirmek için çeşitli araçlar geliştirilmiştir. Kaynak kodları, Java ile kodlanıp derlenen ve artık DVM üzerinde çalışabilen Classes.dex dosyasını bir sınıf dosya formatına dönüştüren dex2jar gibi araçlarla analiz edilebilir. Benzer şekilde, apktool'u kullanarak kaynak koda dönüştürebilirsiniz. AXML2jar, AXMLPrinter2.jar gibi uygulamalar, dosyayı okunabilir hale getirmek için bildirim dosyasını ayrıştırmak için kullanılır. Statik analiz yöntemi, tersine mühendislik yöntemlerine dayandığından, yani kaynak kod doğrulama ilkesini kullandığından, gerçek veya sanal bir cihaz üzerinde çalıştırılmasına gerek yoktur. Bu nedenle kötü amaçlı bir uygulama aranırken uygulama cihaza zarar vermez. Bu, statik analiz yönteminin bir avantajı olarak kabul edilir. Ancak,

uygulamada kodların anlaşılmasını zorlaştıracak gizleme gibi teknikler kullanılırsa, analiz yönteminin bir dezavantajı olarak görülen statik analiz yöntemi ile başarı elde edilemeyebilir [42].

2.2.2. Dinamik Analiz Yöntemi

Dinamik analiz yöntemi, gerçek bir cihaz veya sanal bir cihaz kullanarak uygulamaların çalıştırılmasına ve yürütme sırasında uygulamanın davranışının izlenmesine dayanır. Uygulamayı çalıştırdıktan sonra uygulamanın sistem üzerindeki etkisini, ağ üzerindeki davranışını, API çağrılarını, pil kullanımını izleyerek geliştirilen yazılımı analiz edebilirsiniz. Statik bir ayrıştırma yönteminde ayrıştırmayı karmaşıklaştırmak için kullanılan kod gizleme tekniği, dinamik bir ayrıştırma yönteminde etkisini kaybeder ve ayrıştırma sırasında hiçbir etkisi olmaz. Bu anlamda dinamik analiz yöntemi, statik analiz yöntemine göre bir avantaja sahiptir. Ancak dinamik tarama yönteminde uygulama çalıştırılarak uygulama tarandığından, uygulama kötü niyetli ise gerçek cihazlarda başlatıldığında cihaz etkilenecek ve zarar görecektir. Bu nedenle uygulamaların sandbox (sanal alan-sanal aygıt-emülatör) adı verilen sistemlerde çalıştırılması daha uygun görülmemektedir. Öte yandan sanal ve izole sistemler oluşturmak maliyetli ve zaman alıcı bir yöntemdir. Aynı zamanda dinamik analiz yöntemi, statik analizden daha fazla zaman gerektirir. Analiz edilen sanal platformlarda performans isteniyorsa bunun da bir bedeli vardır. Yani Genymotion, sanal cihazlar oluşturmak ve kullanmak için tasarlanmış bir yazılımdır ve bu yazılım, hız ve esneklik açısından faydalı bir yazılım olarak kabul edilmektedir. Bu yazılım ücretli olarak sunulmaktadır [43].

3. MATERYAL VE YÖNTEM

3.1. Veri Madenciliği ve Makine Öğrenme

Modern teknolojiler hızla gelişiyor ve geliştikçe yetenekleri de her geçen gün artıyor. Bu kadar hızlı ilerlemeyle birlikte, insanların bilgisayar teknolojisini kullanmaları sonucunda yeryüzünde sürekli olarak çok büyük miktarda veri oluşmaktadır. Bilgisayarların bilgi saklama kabiliyetindeki artışın yanı sıra bilgilerin kaydedilebildiği alanların sayısında da artış görülmektedir. Ancak bilgisayar sistemleri tarafından üretilen ve tarlalarınızda saklanan bu bilgiler, onunla bir şeyler yapılıncaya kadar anlamsız kalacaktır. Bu anlamsız veriler, belirli amaçlar için işlendiğinde anlam kazanacaktır. Ham ve işe yaramaz verilerin işlenmeden önce bilgi ve anlama dönüştürülmesi sürecine veri madenciliği denir [44]. Veri çeşitliliğinin ve hacminin artmasıyla birlikte "büyük veri", yapay zekâ sistemleri, makine öğrenmesi yöntemleri, veri madenciliği yöntemleri gibi kavramlar bu kavramlara çözüm bulmak için kullanılmış ve sonuca varılmaya çalışılmıştır büyük verilerin keşfi [45]. Bu büyük veri setleri üzerinde yapılan analizler sonucunda stratejik olarak alınması gereken pazarlama kararlarının doğru ve zamanında verilmesi mümkün olup, bu sayede birçok alanda kazanımlar elde edilebilmektedir. Bu nedenle araştırmacılara bu konularda yardımcı olacak veri madenciliği, makine öğrenmesi ve yapay zekâ gibi sistemler önem kazanmış ve çalışmaya değer hale gelmiştir. Eğitimsel veri madenciliği, eğitimsel veri madenciliği topluluğu tarafından, eğitim ortamından gelen benzersiz veri türlerini keşfetmek için yöntemler geliştirmek ve bu yöntemleri öğrencileri ve öğrendikleri ortamı daha iyi anlamak için kullanmaktan kaynaklanan bir disiplin sorunu olarak tanımlanmaktadır. Eğitim ortamında madencilik, eğitimsel veri madenciliği olarak adlandırılır. Eğitim veri tabanından bilgi çıkarmak için yeni yöntemler geliştirmekle ilgileniyor. Eğitimsel veri madenciliği, eğitim sisteminin bu zorlukların üstesinden gelmesine yardımcı olabilecek ve böylece ülkenin geleceği üzerinde etkisi olacak bir alan haline gelebilecek bir dizi yöntem sunmaktadır. Bu kavramlara ek olarak zaman içinde "Veri Hazırlama" adında bir konudan bahsetmek gerekiyor. Veri hazırlama, veri analizinde en temel adımdır. Bu nedenle, ham verileri temizleme sürecinde genellikle araştırma ve geliştirme vardır. Bu konudaki araştırmalar incelendiğinde, metin ve web sayfası madenciliği, duygu madenciliği ve duygu analizi gibi konularda sınıflandırma, kümeleme ve yapay sinir ağlarının kullanıldığı görülmektedir. Bir perakende işletmesi için, operatörün müşteriye göre özelleştirilmiş satış hareketlerini içeren bir veri tabanı ile detaylı ve göreceli ölçüm sonuçlarını içeren bir satış

gerçekleştiren bir sistem, karar ağaçları ile tasarlanır ve büyük bir etki yaratması beklenir iş pazarlama stratejileri hakkında. Konuşma tanıma, sinyal tanıma, nesne algılama, kelime işleme, görsel nesne tanıma, ilaç keşfi gibi daha birçok alanda çok ileri teknolojilere sahiptir. Geliştirilen bu sistemler, günümüzün en son teknolojilerinden biri olan derin öğrenme kavramı ile ilgili yöntemler kullanmıştır. Başka bir duygu analizi çalışmasında, makine öğrenimi yöntemleri, izleyicilerin tutumları, davranışları ve duyguları gibi öznel bilgileri çıkarmak için üç ana yolla araştırıldı. Öğreticiler, yarı öğreticiler ve öğretici olmayanlar olarak ayrılan bu üç ana konuya makine öğrenmesi yöntemleri uygulanmakta ve bunların güçlü ve zayıf yönleri analiz edilmektedir.

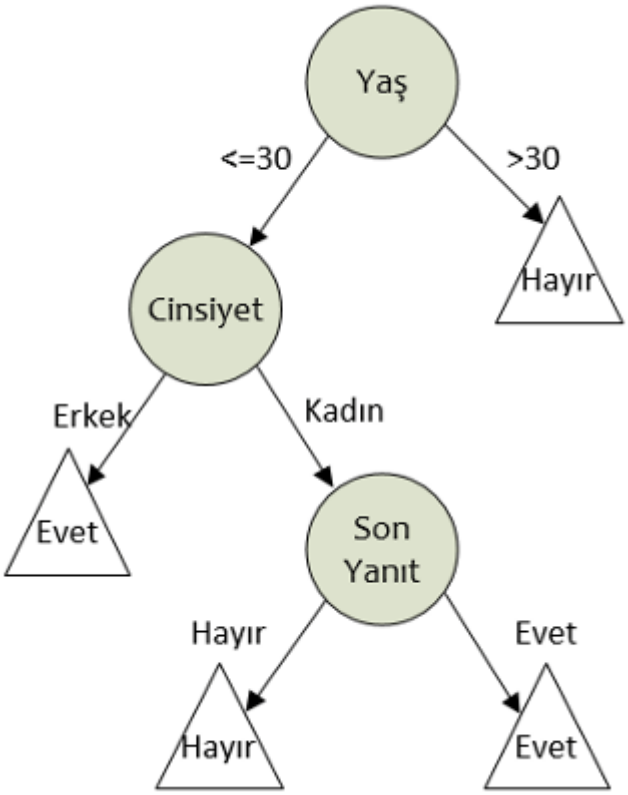
3.2. Makine Öğrenmesi

Bir makine öğrenmesi algoritması türü olan denetimli öğrenme algoritmaları, giriş ve çıkışları tanımlayan önceden etiketlenmiş sınıflara sahip olmalıdır. Eğitim sürecinde elde edilen bilgiler, daha sonra test verilerinin hangi sınıfa ait olduğunu belirlemek için kullanılır. Eğitim süreci, etiketlenmiş bir eğitim veri seti gerektirir. Denetimli öğrenmede, "usta", verilen bir uyarana istenen tepkinin ne olması gerektiğini belirterek ağı eğitmek için kullanılır [46]. Denetimli öğrenme, bir dizi eğitim verisi gerektirir. Sınıflandırıcı, bu ön bilgi kullanılarak tasarlanmıştır. Denetimli öğrenme algoritmalarının örnekleri arasında regresyon analizi, karar ağaçları, k-komşular, sinir ağları ve destek vektör makineleri bulunur. Denetimli öğrenme görevi, önceden etiketlenmiş G/Ç çiftlerine sahip olmalıdır. Sistemin öğrenebileceği bu eğitim örnekleri, verilerdeki kalıpları veya eğilimleri eşleştirmek için eğitim sisteminden gelen bilgileri kullanır. Eğitim başarılı olursa, yeni girdilere maruz kaldığında, eğitilen sistem daha önce öğrenilen davranışı taklit etmek için kendi çıktılarını oluşturabilir. Her eğitim örneği, bir çift girdi ve çıktı değerinden oluşur. Sınıflandırma probleminde, her eğitim örneği bir öznitelik vektörü üretir. Denetimli öğrenme, bu özelliklerle eşleşen bir çıktı sınıfı oluşturur.

3.2.1. Karar Ağaçları

Karar ağacı algoritmasının amacı veri matrisini belirli kurallar silsilesinde homojen alt gruplara ayırmaktır. Karar ağacı algoritması böl ve yönet stratejisini uygulayan hiyerarşik bir veri yapısıdır. Parametrik olmayan kural tabanlı bir tekniktir. Karar ağacı algoritması makine öğrenmesi algoritmaları içerisinde en basit yöntem olarak değerlendirilebilir. Karar ağacı algoritması sürekli veri seti olmayan sınıflandırmalar için daha uygundur. Veriyi

özniteliklere göre belirlediği eşik değeri temel alacak ,şekilde ikiye bölerek bir sınıflandırma işlemi gerçekleştirerek bir ağaç yapısı oluşturmaktadır. Bu algoritma esasında kök düğüm, orta düğümler ve yapraklardan oluşmaktadır. Her dal belli bir sonuca ulaşarak sınıflandırma işlemini ifade etmektedir. Her bir karar düğümü, dalları etiketleyen ayrık sonuçlara sahip bir test fonksiyonu uygulamaktadır. Bir girdi verildiğinde, her düğümde bir test uygulanır ve sonuca bağlı olarak dallardan birine yöneliniz. Bu işlem kökten başlayarak yapraklara ulaşıncaya kadar devam etmektedir. Yaprakta yazan değer çıktıyı vermektedir. Karmaşık bir problem basit ve seri yapılara ayrıştırılmaktadır [47].



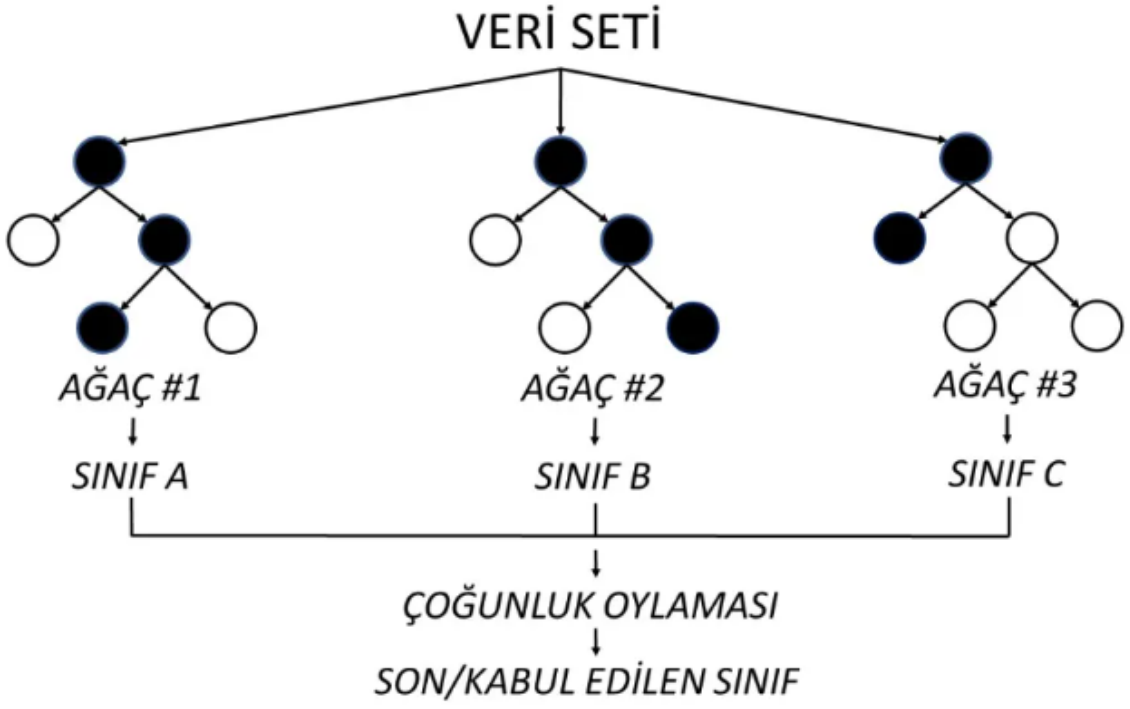
Şekil 3.1. Karar ağacı algoritması [48].

Sınıflandırma işlemi için bir yaprak, düğüm, giriş alanından bu bölgeye düsen örneklerin aynı etiketlere sahip olduğu bir bölge tanımlanmaktadır. Karar mekanizmasını oluşturan sınırlar kökten yaprak düğüme giden yol üzerindeki dâhili düğümlerde kodlanan diskriminantlar tarafından tanımlanmaktadır. Öğrenme sırasında verinin içerdiği problemin karmaşıklığına bağlı olarak ağaç büyür, yeni dallar ve yapraklar eklenmektedir. Karar ağaçları, öğrenme örneğini daha küçük ve daha küçük parçalara bölen bir dizi soru ile temsil edilmektedir. Bu kurallara göre hiyerarşik yapıya sahip bir ağaç yapısı oluşturulmaktadır. Ağaçlar, öznitelik değerlerini izin verilen bir hipotezi temsil eden bir

dizi karar sınıfına esleyen bir işlevi temsil etmektedir. Tüm düğümler, seçilen bölme kriterlerine göre oluşturulan öznitelikler üzerinde kurallar içermektedir. Kurallar verileri o verinin öznitelik değerlerine göre bölen veri bölünmesini belirlemenin yolunu temsil etmektedir. Kökten yapraklara doğru giderken dallarda özniteliklerin ayrılması için sonuçlar verilmektedir. Tüm kural sonuçları dallarla temsil edilmektedir. Yapraklar ayrışmanın gerçekleştirildiği son basamaklardır.

3.2.2. Rastgele Orman

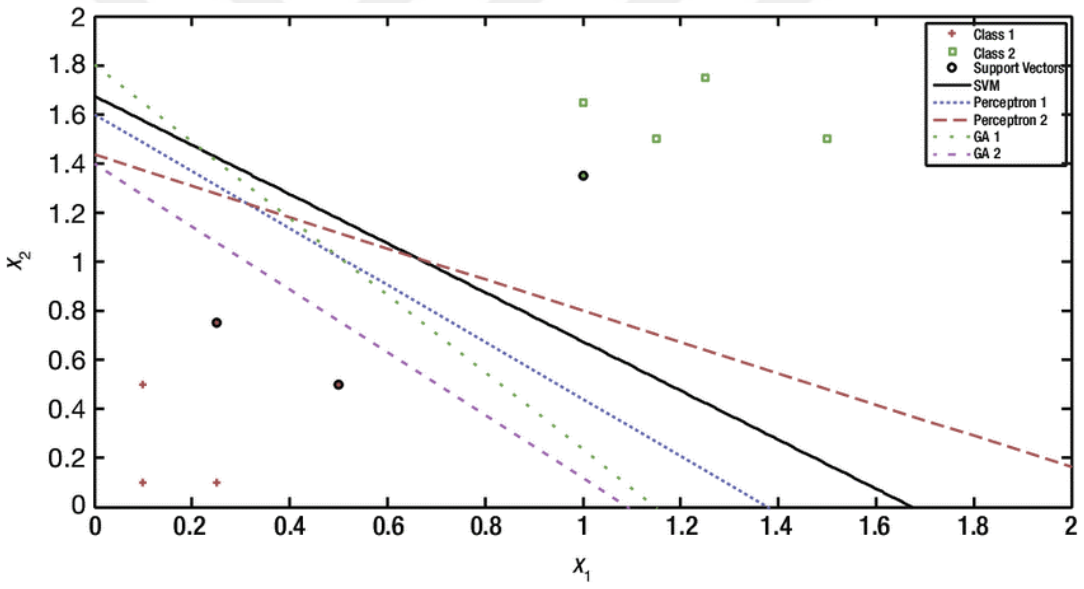
Rastgele ormanlar, her biri rastgele bir eğitim kümesinin alt kümesini veya girdi özelliklerinin rastgele bir alt kümesini kullanan bir değil birden çok karar ağacının eğitilip sonuçlarının her birinin bir oy hakkı olacak ,şekilde sınıflandırma işlemini gerçekleştirmesidir [49]. Karar ağaçlarının bir kombinasyonudur. Rastgele orman algoritmasının genelleme hatası, ormandaki tek tek ağaçlara ve bu ağaçlar aralarındaki korelasyona bağlıdır. Karar ağaçları aşağıdaki şekilde sunulmuştur.



Şekil 3.2. Rastgele orman [50].

3.2.3. Destek Vektör Makineleri Algoritması

Destek vektör makineleri ikili sınıflandırma için çok fazla tercih edilen özellikle çekirdek fonksiyon aracılığıyla yüksek boyutlu uzaya geçiş sağlanarak doğrusal olmayan sınıfların ayrımında da kullanılan başarılı bir yöntemdir. SVM sınıflandırmayı iki sınıfı ayıran optimum hiperdüzlemi bularak optimizasyon problemine çözüm bulacak ,şekilde gerçekleştirmektedir. Destek vektör makineleri için iki yaklaşım bulunmaktadır. Sorunun doğasına bağlı olarak doğrusal ve doğrusal olmayan çekirdekler kullanılabilir. Genel olarak, SVM'ler veri sınıflarını bir hiper düzlem kullanarak sınıflar arasındaki en büyük uzaklığa göre ayıran bir sınıflandırma işlevi bulmaktadır. Uygulamadan bağımsız olarak tüm makine öğrenme algoritmalarında ayıran temel fark, en uygun ayırma alt düzleminin hesaplanmasında yer alan matematiksel işlemlerdir. Optimal hiper düzleme yakın veri noktaları "destek vektörleri" olarak adlandırılmaktadır [51].

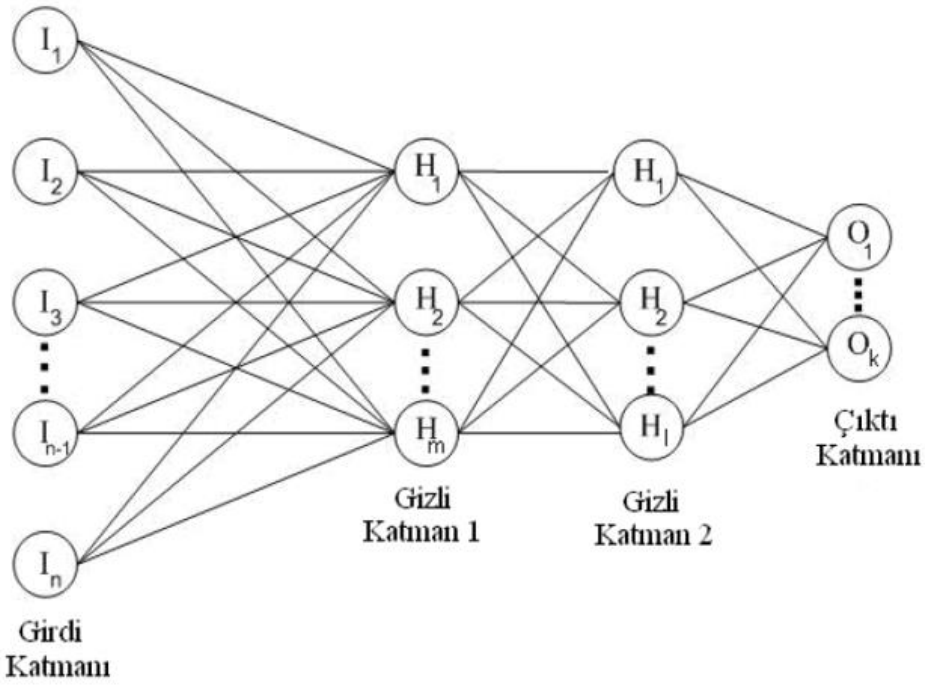


Şekil 3.3. Destek Vektör Makineleri Algoritması [52].

3.2.4. Yapay Sinir Ağları

İnsan beyninden esinlenerek tasarlanan sinir ağları sınıflandırma ve regresyon işlemleri için kullanılmaktadır. Basit bir sinir ağı bir girdi, bir gizli katman, bir çıktı ve katmanlarda bulunan nöronlardan oluşmaktadır. Nöron sayısı öznitelik sayısına, sınıf sayısına ve gizli katman sayısına göre değişebilmektedir. Doğrusal olmayan aktivasyon fonksiyonları, ağırlıklar ve bayesler ile bir katmandaki tüm nöronlar birbirleri ile ilintili durumdadır. Derin öğrenmenin de temeli olan gizli katmandaki nöron sayısının artması kimi zaman öğrenme kapasitesini artırırken kimi zaman azaltmaktadır. Bu performans temelde

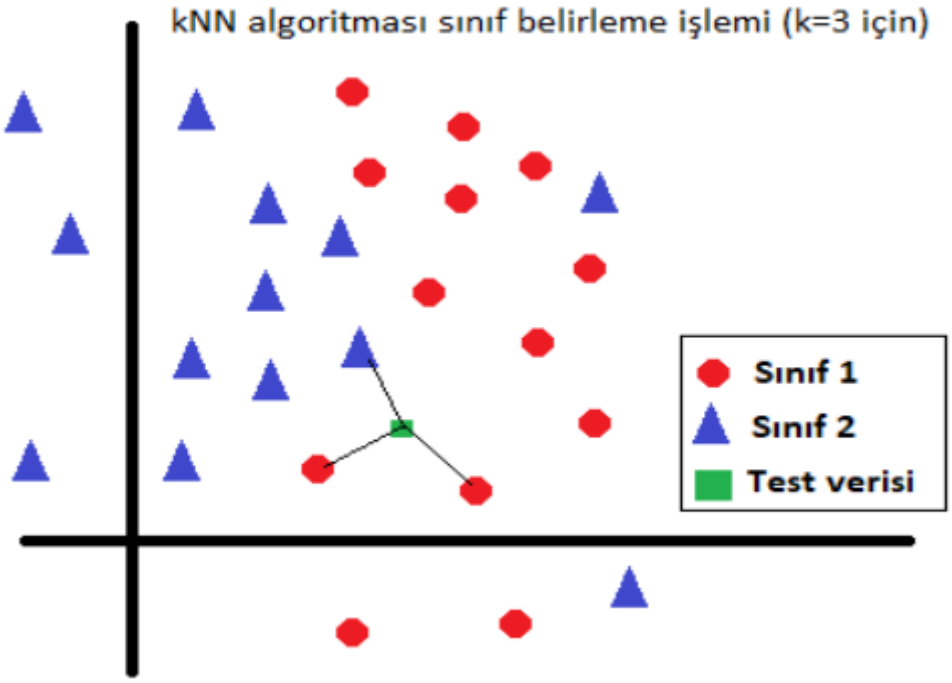
problemin cinsine bağlıdır. Sınıflandırma ya da öğrenme problemine bağlı olarak en optimum gizli katman sayısı belirlenmelidir. Sinir ağları uygulamakta ki temel amaç, öğrenme sistemini uyarlamak için bir dizi girdi verisi ve istenen sistem yanıtlarının kullanılmasıdır [53]. Bu sistem yanıtlarının basarım performanslarının artırılması için ağırlıkları güncellenmiş birçok iterasyon yürütülebilmektedir. Öğrenme sisteminin çıktısı ile bilgi uzmanından istenen yanıt arasındaki hata ölçülmektedir. Hata sinyali daha sonra öğrenme sisteminin yanıtını değiştirmek için kullanılmaktadır, ağırlıkları sinir ağları için uyarlar, böylece yanıtı bilgi uzmanının yanıtıyla daha yakından eşleşmektedir. Geri yayılım algoritmaları ile her bir iterasyonda güncellenen parametreler sayesinde hata oranı azaltılmış bir eğitme işleminin elde edilmesi amaçlanmaktadır. Örüntü tanıma, sınıflandırma, optimizasyon, kümeleme gibi problemlerde sinir ağları çözümleri etkin sonuçlar sunabilmektedir. Sinir ağları çözümünde her nöron kendisine verilen görevi yürütmek ve arzulanan çıktıyı vermek için tasarlanmaktadır. Nöronların içerisinde gerçekleştirilen ve çıktının belirli bir değer arasında sıkıştırılmasını amaçlayan fonksiyonlara, aktivasyon fonksiyonları denilmektedir. En yaygın aktivasyon fonksiyonları lojistik sigmoid, hiperbolik tanjant, Gauss ve doğrusal aktivasyon fonksiyonudur. Bu fonksiyonların girdi üzerinde yarattıkları etki her birine özgüdür ve çıktının başarım performansının gereksinimlerine bağlı olarak tasarımcı tarafından özellikle seçilmektedir. Sınıflandırma problemleri için doğrusal olmayan aktivasyon fonksiyonları daha iyi sonuçlar üretmektedir [54].



Şekil 3.4. Yapay Sinir Ağları [55].

3.2.5. K-en Yakın Komşu

K en yakın komşu mevcut tüm durumları saklayan ve yeni durumları bir benzerlik ölçütüne, örneğin mesafe ölçütlerine göre sınıflandıran basit bir algoritmadır. KNN, 1970'lerin başında parametrik olmayan bir teknik olarak istatistiksel tahmin ve örüntü tanımada kullanılmıştır. KNN algoritması hem sınıflandırma hem de regresyon problemlerini çözmek için kullanılabilen basit, uygulaması kolay denetimli bir makine öğrenme algoritmasıdır[56]. K-En Yakın-Komşu yaklaşımı, ham veri seti hakkında herhangi bir varsayımda bulunmazken, parametrik olmayan bir sınıflandırma algoritmasıdır. Aynı anda hem basit hem de verimli olmasıyla ünlüdür. Etiketlenmemiş verilerin ait olduğu sınıfı belirlemek için sınıflandırmada farklı kriterler kullanılmaktadır. Belirli bir alandaki son veya son eğitim örneklerine dayalı olarak verileri kategorize etmek için kullanılır. Bu yöntem, yürütme kolaylığı ve düşük hesaplama süresi nedeniyle kullanılır. Sürekli veriler için en yakın komşuları hesaplamak için Öklid mesafesini kullanır. Algoritmanın verimliliği, onu yöneten faktörleri değiştirerek geliştirilebilir. Bu algoritmayı daha verimli hale getirmek için KNN'nin birçok varyantı daha önce incelenmiştir, bu için değişkenler şunları içerir: Yerel Olarak Uyarlanabilir KNN, Ağırlıklı KNN, Metin Kategorizasyonu Optimize Edilmiş KNN Uyarlanabilir KNN, Paylaşılan En Yakın Komşularla KNN, K-Ortalamalarla KNN, DVM, KNN, Mahalanobis Metric ile KNN, Genelleştirilmiş KNN, Bilgilendirici KNN ve Bayes KNN [57].



Şekil 3.5. K en yakın komşu [58].

3.3. Veri Madenciliği

Veri madenciliği, bu bilginin ne olduğu hakkında önceden kabul edilmiş varsayımlar olmaksızın veri bilgisini aramanın bilgisayarlı ve manuel sürecidir. Veri madenciliği, nicel verileri analiz etme, verileri veri sahibi için anlaşılabilir ve yararlı yeni bir şekilde özetleyen mantıksal bir ilişki bulma süreci olarak da tanımlanmaktadır. Veri madenciliği teknikleri ve araçları, şirketlerin gelecekteki eğilimleri tahmin etmelerini ve daha bilinçli iş kararları almalarını sağlar. Bu açılımla birlikte kullanışlı ve çalışılabilir veriler olduğu gibi anlamsız veriler de oluşur. Bilgisayar sistemlerinin ürettiği veriler, diğer verilerden ayrı tutulduğunda anlamsızdır. Bu veriler çıplak gözle bakıldığında anlamsızdır, ancak bu veriler belirli bir amaç için işlendiğinde anlamlı hale gelmektedir [59]. Ham verilerin işlenmesi sonucunda değerli verilerin elde edilmesi sürecine veri madenciliği denir. Ham verilerden değerli içgörüler elde edilerek geleceğe yönelik doğru tahminlerde bulunulabilir ve bu tahminlerle kurum, kuruluş veya birey olarak kazanç elde edilebilir. Veri madenciliği veri boyutu, veri madenciliği uygulamamız için bir engel değildir. Veri miktarı arttıkça veya azaldıkça, çıktı veya işlem süresi artabilir veya azalabilir, ancak veri madenciliği işlemleri yine de çalışabilir. Böylece bireylerin tek başına ya da ekip olarak

yapamayacakları işlemler bir teknoloji geliştirme ekibi yardımıyla tamamlanabilmekte ve veri madenciliği sonucunda tüm dünyada büyük bir verimlilik elde edilebilmektedir.

Ham verinin ve elde edilmek istenilen verinin büyüklüğüne bağlı olarak gerçekleşen süreçler zaman zaman değişiklik gösterse de genel olarak aşağıdaki gibidir:

Veri Seçimi: Çözüm sağlamak istenen probleme ilişkin veriler seçilmelidir ve seçilen veriler iyi analiz edilmelidir. Seçim tamamlandıktan sonra gerçekleştirilen tüm aşamalar bu seçim üzerinden yürüyeceğinden dolayı kritik bir adımdır.

Ön İşleme ve Veri Temizleme: Bu aşama daima uygulanan bir aşama olmamakla birlikte tutarsız ve gürültülü verilerin işlenmesi adımıdır. Tutarsız veya gürültülü veri, veri seti içerisinde amaca uygun fakat bilgiyi elde etmeyi zorlaştırıcı unsurlar taşıyan verilere denilmektedir. Bu verilerin temizlenmesi aşamasında isteğe bağlı olarak bu veri kümesi tamamen atılabilmektedir. Ayrıca verinin veri kümesinden tamamen atılmasının istenmediği durumlarda yerine genel bir sabit veya ortalama değerler konulabileceği gibi ilgili değerin tahmini de yine farklı veri madenciliği yöntemleriyle yapılabilmektedir.

Veri İndirgeme: Günümüzde teknoloji oldukça gelişmektedir fakat teknolojinin gelişimi zaman zaman verinin büyüklüğünden dolayı ilgili veri setine veri madenciliği uygulamaya yeterli olmayabilmektedir. Bu gibi büyük veri setlerine veri madenciliği uygularken sonuca etkisinin önemsenmeyecek derecede az olduğu düşünülen veri veya niteliklerin sayıları azaltılabilmektedir.

Veri Bütünleştirme: Veri madenciliği uygulanan veri setleri her zaman tek kaynaktan toplanmayabilmektedir. Farklı kaynaklardan toplanarak işlenmek üzere oluşturulan veri ambarına verilerin aktarılması safhasında aynı verinin farklı şekillerde ifade edilmesiyle karşılaşılabilmektedir. Örnek olarak aynı veri ambarına kaydedilmek üzere farklı kaynaklardan alınan cinsiyet değerleri “K”, “Kadın”, “Female” olarak farklı şekillerde kaydedilmiş olabilmektedir. Bu duruma çözüm olarak uygulanan Veri Bütünleştirme işleminde verilerin tamamı aynı şekilde kaydedilmelidir. Bu sayede bu şekilde eklenen veriler birbirinden farklı değil birbiriyle aynı değerleri ifade edecektir.

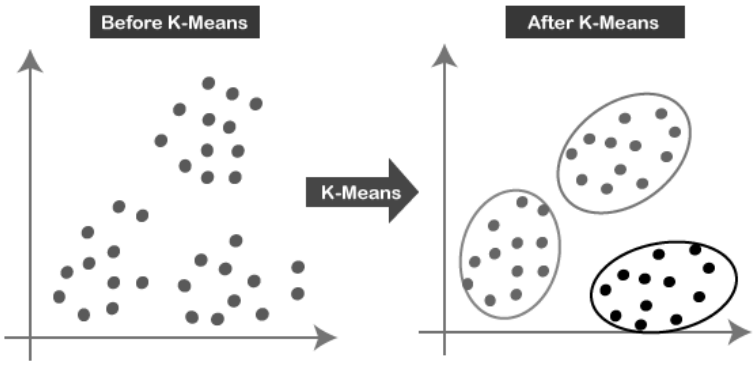
Veri Dönüştürme: Zaman zaman bazı verilere veri madenciliğini direk uygulamak uygun olmayabilmektedir. Değişkenlerin ortalama ve varyant değerlerinin birbirlerinden önemli derecede farklı olması durumunda, ortalama ve varyant değerleri büyük olan verilerin tüm veri seti üzerindeki etkisi önemli derecede dengesizlik sağlamaktadır. Ayrıca benzer bir

şekilde veri seti içerisindeki büyük veya küçük uç değerler sonuçların sağlıklı olmamasına yol açabilmektedir. Bu nedenlerden dolayı verinin tipine göre istenilen normalleştirme veya standartlaştırma algoritması kullanılarak dönüşümünün sağlanması ve bu şekilde veri madenciliğine devam edilmesi gerekmektedir.

Veri Madenciliği Aşaması: Sürecin önceki adımları sonucu ortaya çıkan veri seti kullanılarak, veri setinin tipi ve çalışmada ulaşılmak istenen amaca yönelik veri madenciliği yöntemi uygulanarak sonuca ulaşılmaktadır. Yöntemler çeşitli algoritmalar yardımı sonucu sonuç vermektedir, her algoritma kendine has karakteristik özelliklere ve veri giriş çıkış tipine sahiptir.

3.3.1. K-Ortalamalar Algoritması

K-Ortalamalar Algoritması K-ortalamalar yöntemi, aynı küme içerisindeki noktalar arasında ortalama kare mesafesini bularak çalışır. Ortalama kare mesafesini en aza indirgemeyi amaçlayan bu kümeleme yöntemi genellikle yaygın kullanılan bir yöntemdir. Kümeleme yöntemlerinin genelinde olduğu gibi doğruluk konusunda garanti sunmayan K-ortalamalar Algoritması, basit ve hızlı olduğundan dolayı pratikte kullanıcılara çok cazip gelmektedir [60]. Kullanılan matematiksel yönteme göre bir merkez belirlenmektedir. Belirlenen bu merkez her sınıfa uzaklığı ölçülerek, uzaklık değerine göre yeni kümeye yerleştirme işlemi gerçekleştirilmektedir.

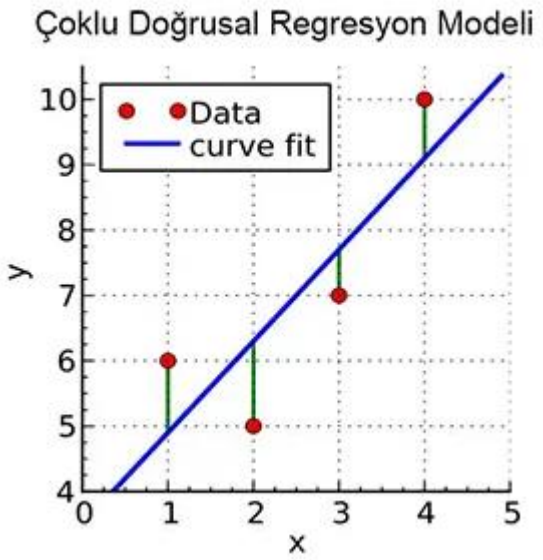


Şekil 3.6. K-Ortalamalar Algoritması [61].

3.3.2. Doğrusal Regresyon

Regresyon analizi, sebep-sonuç ilişkisiyle birbirine bağlı olarak bulunan değişkenler arasındaki ilişkiyi, o konuda tahmin veya kestirimler yapmak amacıyla tasarlanan istatistiksel analiz tekniğidir. Regresyon analizinin en önemli bölümü, model

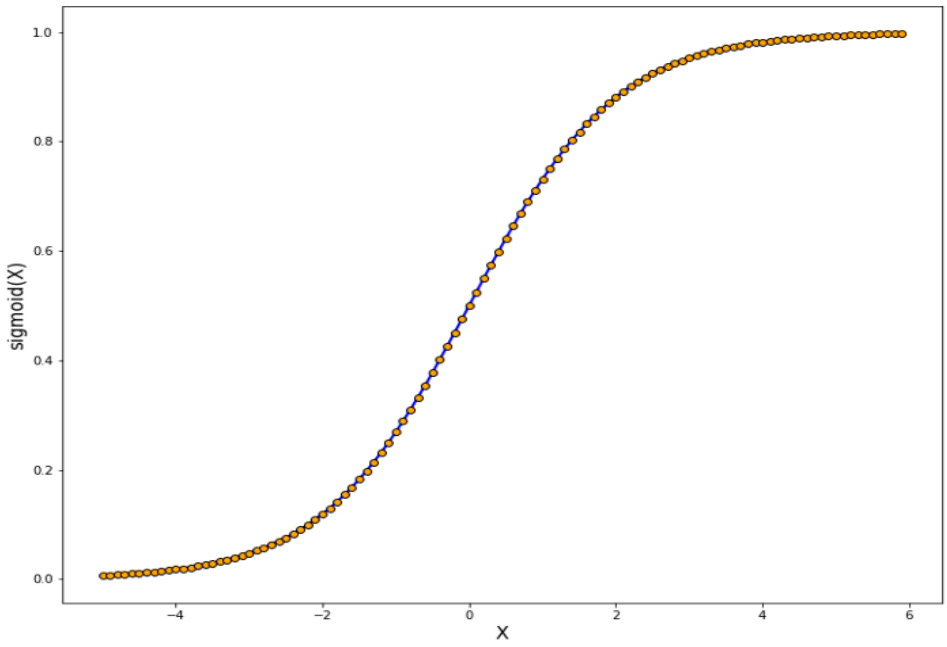
oluşturulduktan sonra modelin yeterli olup olmadığının kontrol aşamasıdır. Regresyon analizinin tüm varsayımları sağlaması doğru modele yeterli derecede yaklaştığını garanti etmektedir [62].



Şekil 3.7. Doğrusal Regresyon [63].

3.3.3. Lojistik Regresyon

Regresyon analizi, bireylerin farklı gruplara ayrılarak sınıflandırılmasını amaçlamaktadır. Bireylerin hangi gruba üye olduğunu kestirmeyi hedeflemek için bir regresyon denklemi oluşturan bu yöntem, bağımsız değişkenlerin sürekli veya süreksiz olmalarına yönelik bir kısıtlama getirmemektedir. Aynı zamanda lojistik modele uygun oluşturulmuş denklemlerin analizini gerçekleştirecek SPSS (Statistical Package for the Social Sciences), SAS (Statistical Analysis System) gibi birçok paket program bulunmaktadır. Bu paket programlar sayesinde özellikle bankacılık ve sigortacılık alanlarında modeller kolaylıkla incelenebilmekte ve yorumlanabilmektedir [64].



Şekil 3.8. Lojistik Regresyon [65].

3.3.4. Saf Bayes sınıflandırıcıları

Bayes olasılık modeli tabanlı bir algoritmadır. NB algoritmasında sınıflandırma işlemi güçlü bir bağımsızlık varsayımı üzerinde çalışır. Bu durum bir özelliğin olasılığının diğerinin olasılığını etkilemediği anlamına gelir. NB algoritmasının başarısı eğitim verisindeki gürültü, sapma ve varyansa bağlı değişmektedir. NB algoritması, önceden tanımlanmış bir kategoriye ait yeni bir gözlemin olasılığını Bayes teorisine göre tanımlanan bir olasılık modeli kullanarak tahmin etmeyi amaçlar. Algoritmada her bir kategorinin önceki olasılığı, bir dizi değişkenle tanımlanan geniş bir eğitim verileri kümesine dayanarak değerlendirilir. Sınıflandırma, koşullu olasılık yoğunluk fonksiyonu ve posteriori olasılığı hesaplanarak tahmin edilmektedir. Sınıflandırılacak veri setinin x=x1, x2 …, xn olduğu varsayıldığında NB algoritmasının işlem adımları aşağıda maddeler halinde sıralanmıştır[66].

3.3.5. Topluluk öğrenme algoritmaları

Tek bir süper doğru modeli öğrenmeye çalışmak yerine, çok sayıda düşük doğruluklu modeli eğitmeye ve ardından yüksek doğruluklu bir meta-model elde etmek için bu düşük doğruluklu modeller tarafından verilen tahminleri birleştirmeye odaklanan öğrenme yöntemine topluluk öğrenme denilmektedir. Düşük doğruluklu modeller genellikle karmaşık modelleri öğrenemeyen ve bu nedenle genellikle eğitimde ve tahmin zamanında

hızlı olan öğrenme algoritmalarıdır. Bu modellerde elde edilen ağaçlar sığdır. Ancak toplu öğrenmenin arkasındaki fikir, eğer ağaçlar aynı değilse ve her ağaç rastgele tahmin etmekten en azından biraz daha iyiyse, bu tür çok sayıda ağacı birleştirerek yüksek doğruluk elde edilebilmesidir. Topluluk öğrenmede torbalama ve güçlendirme olmak üzere iki yöntem vardır. Torbalama, eğitim verilerinin birbirinden farklı olan birçok kopyasını oluşturmaktan ve ardından düşük doğruluklu modelleri her kopyaya uygulayarak bunları birleştirmekten oluşur. Birleştirme, her düşük doğruluklu modele bir çeşit ağırlıklı oylama verilerek yapılmaktadır. Güçlendirme yönteminin torbalama yönteminden farkı, öğrenme işleminin bağımsız olarak değil, sıralı olarak gerçekleşmesidir. Güçlendirmede, denetimli öğrenmeye dayalı olarak ağırlıklar art arda ayarlanır ve birden çok öğrenme sonucu aranır. Bu sonuçlar daha sonra genel doğruluğu artırmak için birleştirilir. Torbalama yöntemiyle çalışan en yaygın algoritmalarından biri RF'dir. RF'de kullanılan ağaçlar, ikili yinelemeli bölümleme ağaçlarına dayanmaktadır. Bu ağaçlar, her bir değişken üzerinde ikili bölmeler kullanarak tahmin alanını böler. Ağacın kök düğümü tüm tahmin alanını kapsar. Bölünmemiş düğümlere terminal düğüm denilmektedir ve bu düğümler tahmin uzayının son bölümünü oluşturur. Her bölümlenmiş düğüm biri solda diğeri sağda olmak üzere iki alt düğüme ayrılır. Sürekli bir tahmin değişkeni için ayrım bir bölme noktası ile belirlenir. Tahmincinin bölünme noktasından daha küçük olduğu noktalar sola, büyük olanlar ise sağa gider [67].

3.4. Önerilen Yöntem

Bu çalışmada, kötü amaçlı yazılımların tespiti için K-NN ve ızgara arama dayalı yeni uygulama sunulmuştur. En Yakın Komşu, Denetimli Öğrenme tekniğine dayanan en basit Makine Öğrenimi algoritmalarından biridir. Yeni vaka / veri ile mevcut vakalar arasındaki benzerliği varsayar ve yeni vakayı mevcut kategorilere en çok benzeyen kategoriye yerleştirir. K-NN algoritması mevcut tüm verileri saklar ve yeni bir veri noktasını benzerliğe göre sınıflandırır. Bu, yeni veriler göründüğünde K-NN algoritması kullanılarak kolayca bir süit kategorisine sınıflandırılabileceği anlamına gelir. Sınıflandırma yapmadan önce alınması gereken iki önemli karar vardır. Birincisi kullanılacak K değeri; buna isteğe bağlı olarak karar verilebilir veya en uygun değeri bulmak için çapraz doğrulamayı deneyebilirsiniz. Bir sonraki ve en karmaşık olan, kullanılacak mesafe metriğidir. Mesafeyi hesaplamanın birçok farklı yolu vardır, çünkü oldukça belirsiz bir kavramdır ve kullanılacak uygun metrik her zaman veri kümesi ve sınıflandırma görevi tarafından

belirlenecektir. Bununla birlikte, iki popüler olan Öklid mesafesi ve Kosinüs benzerliğidir. Bu çalışmada, K-en yakın komşu algoritmasının optimum hiper parametrelerini bulmak için ızgara arama algoritması sunulmuştur. Önerilen yöntemimiz birkaç ardışık adıma dayanmaktadır. Çalışmamızda kullandığımız veri setine gelince, İnternetten hazır bir veri seti kullandık [68]. Bu veriler (373 örnek) ve her örnek (531) özellikten oluşmaktadır.

- İlk adımda veri seti dosyası okunur ve sonra girdi özelliklerini ve çıktı işaretlerini iki ayrı matrise ayırmaya çalıştı, (X) matrisinin giriş özelliklerini içerdiği ve matrisin (Y) çıkış işaretlerini içerdiği ve değerlerinin (0,1) olduğu yerde, aşağıdaki kodda olduğu gibi.

  Data = pd. Read_csv('Path/uci_malware_detection1.csv')
  X = Data.drop(columns=['target'])
  Y= Data['target']

- Bir önceki adımdaki verileri hazırladıktan sonra, ikinci adım, ızgara arama algoritmasını ve k-en yakın komşu algoritmasını kullanan eğitim sürecini temsil eder. Algoritma eğitimi yapılmadan önce, Grid Search algoritması üç farklı parametre ile donatılmıştır: İlk parametre (KNN algoritması). İkinci parametreye hiperparametreler denir, üç parametrenin (leaf_size, n_neighbors ve p) değerlerini içeren bir sözlük biçimindedir. (N_neighbors) k-komşu sorguları için kullanılacak komşu sayısını temsil eder ve (leaf_size) her düğümdeki nokta sayısını temsil eder, (P) parametresine gelince, (P = 1) değeri Manhattan mesafesini kullanırsa ve (P = 2) ise Öklid mesafesi kullanılır. Üçüncü parametre ise (CV), bu parametre, veri kümesinde gerçekleşen çapraz doğrulama bölme stratejisini belirler. Bir değer atandığı için (CV=10). algoritmaları gerekli değerlerle hazırladıktan sonra, algoritmaların veriler üzerinde eğitilmesi süreci başlar ve aşağıdaki kodda olduğu gibi en iyi sonuçlara yol açan en iyi parametreler seçilir.

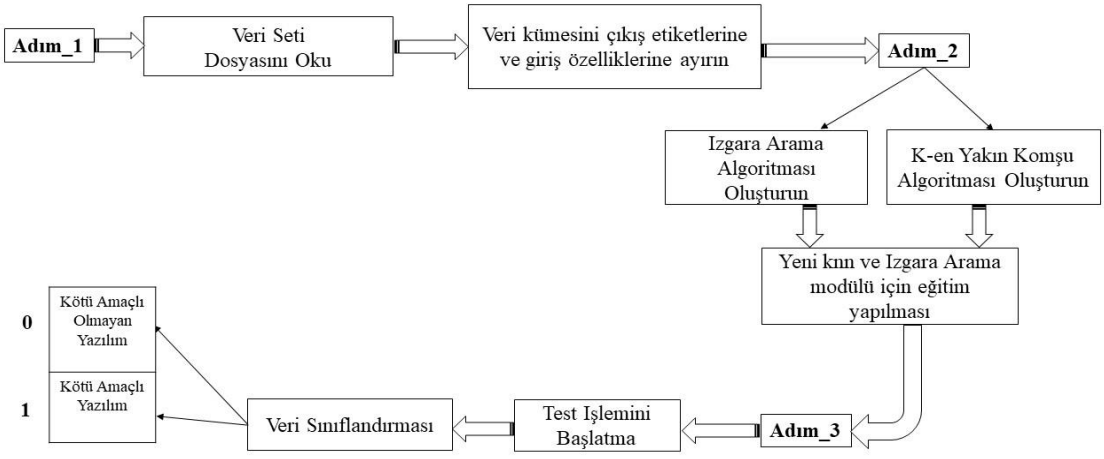
  Leaf_size = list (range (1, 5))
  N_neighbors = list (range (1, 4))
  P= [1, 2]
  Hyperparameters = dict (leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
  Knn_2 = KNeighborsClassifier ()
  Clf = GridSearchCV (knn_2, hyperparameters, CV=10)
  Best_model = clf.fit (x, y)
  Print ('Best leaf_size:' best_model. Best_estimator_. Get_params () ['leaf_size'])
  Print ('Best p:' best_model.best_estimator_.get_params () ['p'])

Print ('Best n_neighbors:' best_model. best_estimator_.get_params () ['n_neighbors']).

- Üçüncü adım, bu adımda, eğitim sürecinden üretilen tüm optimize edilmiş hiperparametreler kullanılarak eğitilmiş algoritmalar üzerinde test süreci gerçekleştirilir**.** Test verileri üzerinde bir test prosedürü gerçekleştirdikten sonra, veriler iki kategoriye ayrılır, kötü amaçlı yazılım (hedef = 1) veya kötü amaçlı olmayan programlar (hedef = 0). Aşağıdaki kodda olduğu gibi.

Dict=best_model.cv_results_ ["params"]
For Params_Value in Dict:
Ls=Params_Value['leaf_size']
N=Params_Value['n_neighbors']
P1=Params_Value['p']
PA= dict (estimator__leaf_size=ls, estimator__n_neighbors=n, estimato r__p=p1)
*Test_Model=best_model.set_params (**PA)*
Y_pred = Test_Model.predict (x_test)
Print (classification_report (y_test, y_pred))

Şekil 3.9, çalışmamızda önerilen yöntemin adımlarını basitleştirilmiş bir şekilde sunmaktadır.



Şekil 3.9. Önerilen Yöntemin Adımları.

4. SONUÇLAR

4.1. Doğrulama Tekniği (Validation Technique)

Veri, makine öğrenimine güç veren destektir. Bir makine öğrenimi ve derin öğrenme modeli ne kadar güçlü olursa olsun, kötü verilerle hiçbir şey yapamazsınız. Rastgele gürültü modele zarar verebilir. Doğrulama süreci neyin yanlış olduğunu doğrudan belirleyemese de bazen bize modelin kararlılığında bir sorun olduğunu gösterebilir.

En basit veri doğrulama yöntemi, eğitim/doğrulama/veri test işlemlerini gerçekleştirmektir. Bunun için tipik bir oran 80/20 olacaktır. Modeli eğitim seti ile eğittikten sonra, kullanıcı tatmin edici bir performans metriği elde edilene kadar sonuçları doğrulamaya ve seti ve doğrulama parametrelerini ayarlamaya devam etmelidir. Bu adımı tamamladıktan sonra kullanıcı, performansını tahmin etmek ve değerlendirmek için bir dizi test kullanarak modeli doğrular.

Monte Carlo çapraz doğrulama olarak da bilinen rastgele alt örnekleme, verilerin rastgele alt kümelerine dayanır. Böylece kullanıcı alt kümelerin boyutunu belirler. Verilerin rastgele bölümlenmesi herhangi bir sayıda tekrar edilebilir. Son olarak, test sayısı toplanır ve toplam test sayısına bölünür. Böylece fazla takılma sorunu ortadan kalkar.

4.2. Karışıklık matrisi (Confusion Matrix)

Karışıklık matrisi, bir sınıflandırma modelinin performansını değerlendirmek için kullanılan N x N boyutlu bir matristir. Burada N sınıf sayısıdır. Bu matrisi kullanarak gerçek hedef değerleri makine öğrenme modelinin öngördüğü değerlerle karşılaştırır. Böylece sınıflandırma modelinin nasıl çalıştığına ve ne gibi hatalar yaptığına dair bütüncül bir bakış açısı vardır. Karışıklık matrisinde birden fazla parametre bulunmaktadır. Açıklamalar aşağıdadır.

Gerçek Pozitif (TP): Tahmin edilen değer, gerçek değerle eşleşir. Gerçek değer pozitifti ve model pozitif bir değer öngördü şeklinde yorumlanır.

Gerçek Negatif (TN): Tahmin edilen değer, gerçek değerle eşleşir. Gerçek değer negatifti ve model negatif bir değer öngördü şeklinde yorumlanır.

Yanlış Pozitif (FP): Tahmin edilen değer yanlış tahmin edildi. Gerçek değer negatifti ancak model pozitif bir değer öngördü şeklinde yorumlanır.

Yanlış Negatif (FN): Tahmin edilen değer yanlış tahmin edildi. Gerçek değer pozitifti ancak model negatif bir değer öngördü şeklinde yorumlanır.

		TAHMİN		
		YOK	VAR	TOPLAM
GERÇEK	YOK	TN 100	FP 20	120
	VAR	FN 10	TP 200	210
	TOPLAM	110	220	

Şekil 4.1. Karmaşıklık Matrisi.

Doğruluk (ACC), Denklem 4.1'de verildiği gibi temsil edilir.

$$\text{Doğruluk (Acc)} = \frac{(\text{TP} + \text{TN})}{(\text{P} + \text{N})} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \tag{4.1}$$

$$\text{Duyarlılık (TPR)} = \frac{TP}{(TP + FN)} \tag{4.2}$$

$$\text{Özgüllük (SPC)} = \frac{TN}{(FP + TN)} \tag{4.3}$$

Makine öğrenme teknikleri ile geliştirilen uygulama aşağıda belirtilen kısımlarda test edilmiştir. Veri kümesi eğitim ve test gruplarına ayırılmıştır. Önce önerilen yöntem için eğitim gerçekleştirilmiştir ve sonra test işlemi yapılmıştır. Çalışmada sınıflama için KNN ile hibrit oluşturulan izgara arama (Grid Search) kullanılır.

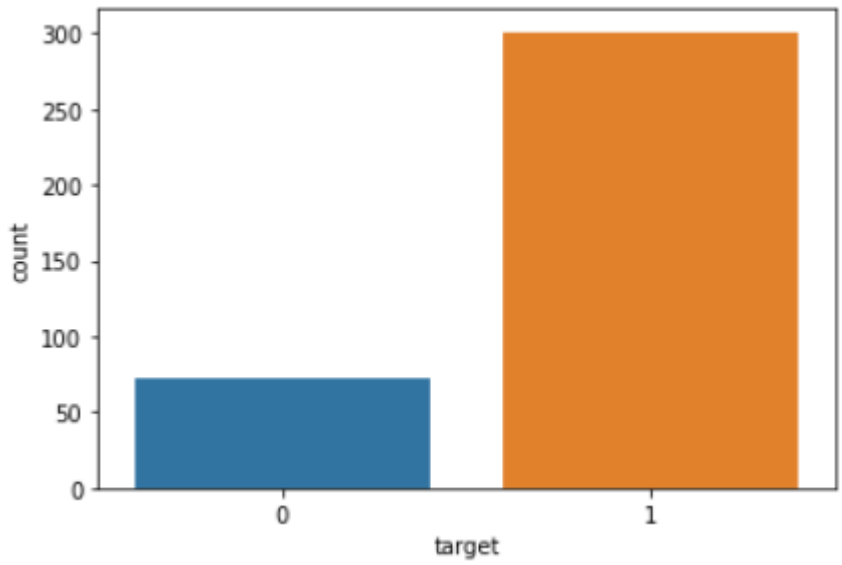
4.3. Kullanılan Hibrit Algoritmaların Sonuçları

Bu bölümde önerilen metot jupyter notebookta python kodu ile yazıldı. Kullanılan makine i7 RAM 8 GB. İlk başta veri seti python ile okundu giriş ve etiket şekil 4.2' de sunuldu.

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_10	...	F_523	F_524	F_525	F_526	F_527	F_528	F_529	F_530	F_531	target
0	1	0	1	0	1	0	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	1	0	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	1	0	1	0	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0
4	1	0	1	0	1	0	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0

Şekil 4.2. Malware Veri Seti python ile okundu.

İkinci adımda, giriş özellikleri ile ve çıkış etiketi ayrı matrislere ayırıldı ve kötü huylu yazılım var ise 1 ya da yok ise 0 iki etiket şekil 4.3 de gösterildi.

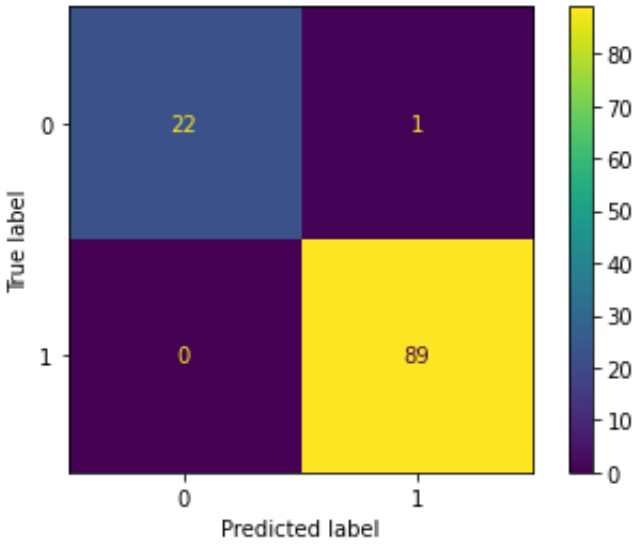


Şekil 4.3. Kategoriler.

Daha sonra veri seti eğitim seti (%70) ve test seti (%30) olarak ikiye ayrılmıştır. İlk başta sadece KNN algoritması ve parametresi (minkowski) ile sınıflandırma işlemi yapıldı. Önerilen çalışmada KNN'in üç parametresi ızgara arama (grid search) algoritması ile optimize edildi bu parametreler (Yaprak boyutu = 1, optimal mesafe = 1, K sayısı =2, ve mesafe_parametresi = Manhattan) olarak ızgara yöntemi ile bulundu, Sonuçlar Tablo 4.1' de sunuldu.

Tablo 4.1. Önerilen yöntemin sonuçları.

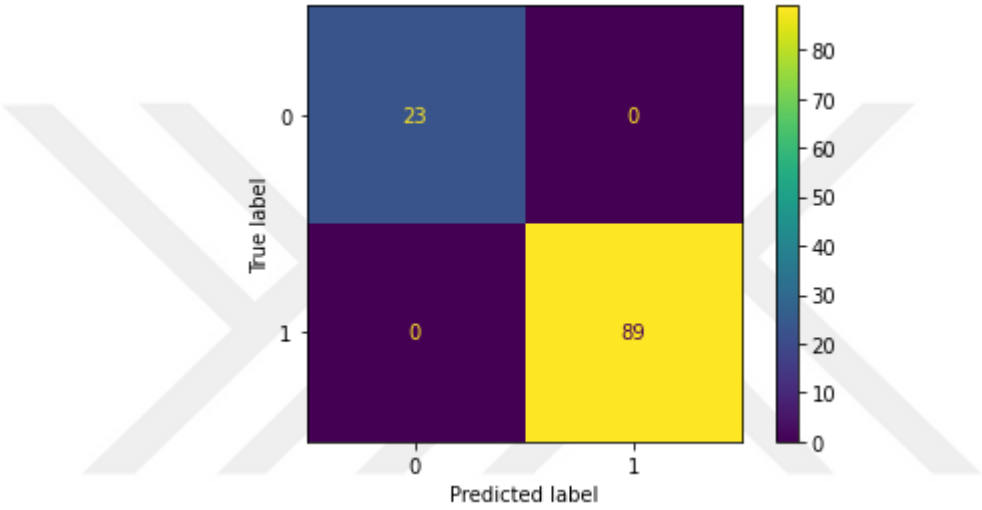
Yaprak boyutu	Optimal mesafe	K sayısı	Mesafe Parametresi	En iyi puan	Hassasiyet	Geri Çağırma	Doğruluk
1	1	1	Manhattan	98.6	1.0	1.0	1.0
1	1	2	Öklid	98.6	1.0	1.0	1.0
1	2	1	Manhattan	98.9	1.0	1.0	1.0
1	2	2	Öklid	98.9	1.0	1.0	1.0
1	3	1	Manhattan	98.4	1.0	1.0	1.0
1	3	2	Öklid	98.4	1.0	1.0	1.0
2	1	1	Manhattan	98.6	1.0	1.0	1.0
2	1	2	Öklid	98.6	1.0	1.0	1.0
2	2	1	Manhattan	98.9	1.0	1.0	1.0
2	2	2	Öklid	98.9	1.0	1.0	1.0
2	3	1	Manhattan	98.4	1.0	1.0	1.0
2	3	2	Öklid	98.4	1.0	1.0	1.0
3	1	1	Manhattan	98.6	1.0	1.0	1.0
3	1	2	Öklid	98.6	1.0	1.0	1.0
3	2	1	Manhattan	98.9	1.0	1.0	1.0
3	2	2	Öklid	98.9	1.0	1.0	1.0
3	3	1	Manhattan	98.4	1.0	1.0	1.0
3	3	2	Öklid	98.4	1.0	1.0	1.0
4	1	1	Manhattan	98.6	1.0	1.0	1.0
4	1	2	Öklid	98.6	1.0	1.0	1.0
4	2	1	Manhattan	98.9	1.0	1.0	1.0
4	2	2	Öklid	98.9	1.0	1.0	1.0
4	3	1	Manhattan	98.4	1.0	1.0	1.0
4	3	2	Öklid	98.4	1.0	1.0	1.0



Şekil 4.4. Klasik K-NN ile karmaşık matris.

Şekil 4.4, klasik KNN algoritmasını ve minkowski parametresini kullanarak sonuçları sunar, Şekil 4.5, optimize edilmiş KNN algoritmasını ve Manhattan parametresini kullanarak sonuçları sunar.

Klasik ve optimal k-nn algoritması arasındaki fark, klasik k-nn algoritmasındaki parametre değerlerinin ya varsayılan değerler ya da kullanıcı tanımlı değerler olmasıdır. Optimum k-nn algoritmasında, k-komşu algoritmasının optimal hiperparametreleri ağ arama algoritması kullanılarak aranır ve bu yöntem, k-komşu algoritmasının veri madenciliğinde çalışmasını iyileştirmesini sağlar, özellikle veri sınıflandırmasında.



Şekil 4.5. İyileştirilmiş (Optimize edilmiş) K-NN ile karmaşık matris.

Yapılan çalışmada hiper parametreler optimize edilmiş KNN ile klasik KNN arasında karşılaştırma yapıldı ve sonuçlar Tablo 4.2' de sunuldu. Yapılan karşılaştırmada iki yöntem için üç önemli parametre hesaplanmıştır Hassasiyet (presicion), Geri Çağırma (recall) ve Doğruluk (accuracy).

Tablo 4.2. Karşılaştırma tablosu.

Parametreler	**KNN**	**KNN ile ızgara arama (grid search)**
Hassasiyet (presicion),	0,98	1,00
Geri Çağırma (recall)	1,00	1,00
Doğruluk (accuracy).	0,99	1,00

Ayrıca, bu bölümde önerilen yöntem birden fazla ilgili çalışma ile karşılaştırılmıştır. Yapılan karşılaştırma bu çalışmada sunulan çalışmanın daha üstün sonuçlar elde ettiğini ispatlamıştır. Tablo 4.3’ de karşılaştırma sunulmuştur.

Tablo 4.3. İlgili çalışmalar ile karşılaştırma.

Kaynak	**Yöntem**	**Performans (%)**
[8]	CNN	97,10
[9]	ResNet	88,36
[9]	GoogleNet	74,5
[10]	LSTM	97,26
[11]	LSTM	99,01
Bu çalışma	KNN- ızgara arama algoritması	100

Kötü amaçlı yazılım tespit yöntemleri, bilgi sistemlerinin ilk günlerinden beri çalışılmaktadır. İlk olarak 1971'de Creeper System [12] adlı bir kötü amaçlı yazılım çeşidi görüyoruz. Creeper System, rakibi olan başka bir kötü amaçlı yazılım olan Reaper [13]'in yaratılmasına ve mevcut kötü amaçlı yazılımları kaldırmak için başka bir kötü amaçlı yazılım dağıtmak yerine kötü amaçlı yazılımları tespit etmek için sistematik bir yaklaşıma ihtiyaç duymasına neden oldu. Erken AI tabanlı çalışmalar, kötü amaçlı ve iyi huylu yazılımlardan elde edilen verileri sınıflandırmak için Makine öğrenimi (ML) sınıflandırma algoritmalarını kullandılar. ML sınıflandırma algoritmaları, öznitelik seçimi ve çıkarımı için zaman ve çaba gerektirdiğinden, tam otomatik yöntemler sağlamazlar. Bu nedenle, son çalışmalarda, derin sinir ağları öğrenme sürecini daha iyi simüle ettiğinden ve daha akıllı ve daha hızlı aracılar sağladığından, odak derin öğrenme (DL) tabanlı yöntemlere kaydırıldı. Derin öğrenme dayalı uygulamalar maliyetli, yavaş ve çok veri gerektirir. Bu nedenle, bu çalışmada sunulan K-NN algoritması çok verimli ve iyi sonuçlar sundu. Önerilen çalışma, 100% doğruluk elde etmiştir ve bu oran önceki çalışmalarla karşılaştırdığımızda çok daha iyidir.

Öneri olarak ızgara arama algoritması yerine başka iyileştirme ve optimizasyon algoritmaları kullanılabilir.

KAYNAKLAR

[1]. Rathore, H., Samavedhi, A., Sahay, S. K., & Sewak, M. (2021). Robust Malware Detection Models: Learning from Adversarial Attacks and Defenses. *Forensic Science International: Digital Investigation, 37*. https://doi.org/10.1016/j.fsidi.2021.301183

[2]. Darem, A., Abawajy, J., Makkar, A., Alhashmi, A., & Alanazi, S. (2021). Visualization and deep-learning-based malware variant detection using OpCode-level features. Future Generation Computer Systems, 125. https://doi.org/10.1016/j.future.2021.06.032

[3]. Bhat, P., & Dutta, K. (2021). A multi-tiered feature selection model for android malware detection based on Feature discrimination and Information Gain. Journal of King Saud University- Computer and Information Sciences. https://doi.org/10.1016/j.jksuci.2021.11.004

[4]. Ou, F., & Xu, J. (2022). S3Feature: A static sensitive subgraph-based feature for android malware detection. Computers and Security, 112. https://doi.org/10.1016/j.cose.2021.102513

[5]. Wen, Q., & Chow, K. P. (2021). CNN based zero-day malware detection using small binary segments. Forensic Science International: Digital Investigation, 38. https://doi.org/10.1016/j.fsidi.2021.301128

[6]. Bala, N., Ahmar, A., Li, W., Tovar, F., Battu, A., & Bambarkar, P. (2022). DroidEnemy: Battling adversarial example attacks for Android malware detection. Digital Communications and Networks. https://doi.org/10.1016/j.dcan.2021.11.001

[7]. Tang, J., Li, R., Jiang, Y., Gu, X., & Li, Y. (2022). Android malware obfuscation variants detection method based on multi-granularity opcode features. Future Generation Computer Systems, 129. https://doi.org/10.1016/j.future.2021.11.005

[8]. Kumar, R., Xiaosong, Z., Khan, R. U., Ahad, I., & Kumar, J. (2018). Malicious code detection based on image processing using deep learning. ACM International Conference Proceeding Series, 81–85. https://doi.org/10.1145/3194452.3194459

[9]. Khan, R. U., Zhang, X., & Kumar, R. (2019). Analysis of ResNet and GoogleNet models for malware detection. Journal of Computer Virology and Hacking Techniques, 15(1). https://doi.org/10.1007/s11416-018-0324-z

[10]. Lu, R. (2019). Malware Detection with LSTM using Opcode Language. http://arxiv.org/abs/1906.04593

[11]. Jahromi, A. N., Hashemi, S., Dehghantanha, A., Parizi, R. M., & Choo, K. K. R. (2020). An Enhanced Stacked LSTM Method with No Random Initialization for Malware Threat Hunting in Safety and Time-Critical Systems. IEEE Transactions on Emerging Topics in Computational Intelligence, 4(5). https://doi.org/10.1109/TETCI.2019.2910243

[12]. Sewak, M., Sahay, S. K., & Rathore, H. (2018). Comparison of deep learning and the classical machine learning algorithm for the malware detection. Proceedings-2018 IEEE/ACIS 19th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2018. https://doi.org/10.1109/SNPD.2018.8441123

[13]. Chen, L. (2018). Deep Transfer Learning for Static Malware Classification. http://arxiv.org/abs/1812.07606

[14]. Dai, Y., Li, H., Qian, Y., & Lu, X. (2018). A malware classification method based on memory dump grayscale image. Digital Investigation, 27. https://doi.org/10.1016/j.diin.2018.09.006

[15]. Halim, M. A., Abdullah, A., Akram, K., & Ariffin, Z. (2019). Recurrent Neural Network for Malware Detection. Int. J. Advance Soft Compu. Appl, 11(1).
[16]. Dai, Y., Li, H., Qian, Y., Yang, R., & Zheng, M. (2019). SMASH: A Malware Detection Method Based on Multi-Feature Ensemble Learning. IEEE Access, 7. https://doi.org/10.1109/ACCESS.2019.2934012
[17]. Türker, S. (2019). Zararlı Android Yazılımlarının Makine Öğrenmesi ile Ailelerine Göre Sınıflandırılması.
[18]. Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2016). DynaLog: An automated dynamic analysis framework for characterizing android applications. 2016 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2016. https://doi.org/10.1109/CyberSecPODS.2016.7502337
[19]. Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., & Stolfo, S. (2013). On the feasibility of online malware detection with performance counters. Proceedings- International Symposium on Computer Architecture. https://doi.org/10.1145/2485922.2485970
[20]. Xu, Z., Ray, S., Subramanyan, P., & Malik, S. (2017). Malware detection using machine learning based analysis of virtual memory access patterns. Proceedings of the 2017 Design, Automation and Test in Europe, DATE 2017. https://doi.org/10.23919/DATE.2017.7926977
[21]. Yang, Y., Wei, Z., Xu, Y., He, H., & Wang, W. (2018). Droidward: An effective dynamic analysis method for vetting android applications. Cluster Computing, 21(1). https://doi.org/10.1007/s10586-016-0703-5
[22]. Sugunan, K., Gireesh Kumar, T., & Dhanya, K. A. (2018). Static and dynamic analysis for android malware detection. In Advances in Intelligent Systems and Computing (Vol. 645). https://doi.org/10.1007/978-981-10-7200-0_13
[23]. Wu, Y., Lee, W. W., Xu, Z., & Ni, M. (2020). Large-scale and robust intrusion detection model combining improved deep belief network with feature-weighted svm. IEEE Access, 8, 98600–98611. https://doi.org/10.1109/ACCESS.2020.2994947
[24]. Liu, X., Di, X., Ding, Q., Liu, W., Qi, H., Li, J., & Yang, H. (2020). NADS-RA: Network Anomaly Detection Scheme Based on Feature Representation and Data Augmentation. IEEE Access, 8, 214781–214800. https://doi.org/10.1109/ACCESS.2020.3040510
[25]. Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network Intrusion Detection Combined Hybrid Sampling with Deep Hierarchical Network. IEEE Access, 8, 32464–32476. https://doi.org/10.1109/ACCESS.2020.2973730
[26]. Tama, B. A., Comuzzi, M., & Rhee, K. H. (2019). TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System. *IEEE Access*, *7*, 94497–94507. https://doi.org/10.1109/ACCESS.2019.2928048
[27]. Ieracitano, C., Adeel, A., Morabito, F. C., & Hussain, A. (2020). A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing*, *387*. https://doi.org/10.1016/j.neucom.2019.11.016
[28]. AV-Test-Institute. (2020, April 5). *New malware*. Https://Www.Av-Test.Org/En/Statistics/Malware/.
[29]. Kakisim, A. G., Nar, M., & Sogukpinar, I. (2020). Metamorphic malware identification using engine-specific patterns based on co-opcode graphs. *Computer Standards and Interfaces*, *71*. https://doi.org/10.1016/j.csi.2020.103443
[30]. Damodaran, A., Troia, F. di, Visaggio, C. A., Austin, T. H., & Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, *13*(1). https://doi.org/10.1007/s11416-015-0261-z

[31]. Zhang, J., Qin, Z., Yin, H., Ou, L., & Zhang, K. (2019). A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding. *Computers and Security*, *84*. https://doi.org/10.1016/j.cose.2019.04.005
[32]. Alazab, M. (2015). Profiling and classifying the behavior of malicious codes. *Journal of Systems and Software*, *100*. https://doi.org/10.1016/j.jss.2014.10.031
[33]. Ali, M., Shiaeles, S., Bendiab, G., & Ghita, B. (2020). Malgra: Machine learning and N-GRAM malware feature extraction and detection system. *Electronics (Switzerland)*, *9*(11), 1–20. https://doi.org/10.3390/electronics9111777
[34]. Cho, I. K., Kim, T. G., Shim, Y. J., Ryu, M., & Im, E. G. (2016). Malware Analysis and Classification Using Sequence Alignments. *Intelligent Automation and Soft Computing*, *22*(3). https://doi.org/10.1080/10798587.2015.1118916
[35]. Alam, S., Traore, I., & Sogukpinar, I. (2014). Annotated Control Flow Graph for Metamorphic Malware Detection. *Computer Journal*, *58*(10). https://doi.org/10.1093/comjnl/bxu148
[36]. Sung, Y., Jang, S., Jeong, Y. S., & Park, J. H. (James J.). (2020). Malware classification algorithm using advanced Word2vec-based Bi-LSTM for ground control stations. *Computer Communications*, *153*. https://doi.org/10.1016/j.comcom.2020.02.005
[37]. Yan, J., Qi, Y., & Rao, Q. (2018). LSTM-Based Hierarchical Denoising Network for Android Malware Detection. *Security and Communication Networks*, *2018*. https://doi.org/10.1155/2018/5249190
[38]. Bilar, D. (2007). Opcodes as predictor for malware. *International Journal of Electronic Security and Digital Forensics*, *1*(2). https://doi.org/10.1504/IJESDF.2007.016865
[39]. Runwal, N., Low, R. M., & Stamp, M. (2012). Opcode graph similarity and metamorphic detection. *Journal in Computer Virology*, *8*(1–2). https://doi.org/10.1007/s11416-012-0160-5
[40]. Yewale, A., & Singh, M. (2017). Malware detection based on opcode frequency. *Proceedings of 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016*. https://doi.org/10.1109/ICACCCT.2016.7831719
[41]. Khalilian, A., Nourazar, A., Vahidi-Asl, M., & Haghighi, H. (2018). G3MD: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families. *Expert Systems with Applications*, *112*. https://doi.org/10.1016/j.eswa.2018.06.012
[42]. Kang, J., Jang, S., Li, S., Jeong, Y. S., & Sung, Y. (2019). Long short-term memory-based Malware classification method for information security. *Computers and Electrical Engineering*, *77*. https://doi.org/10.1016/j.compeleceng.2019.06.014
[43]. Hashemi, H., Azmoodeh, A., Hamzeh, A., & Hashemi, S. (2017). Graph embedding as a new approach for unknown malware detection. *Journal of Computer Virology and Hacking Techniques*, *13*(3). https://doi.org/10.1007/s11416-016-0278-y
[44]. Morid, M. A., Lau, M., & del Fiol, G. (2021). Predictive analytics for step-up therapy: Supervised or semi-supervised learning? *Journal of Biomedical Informatics*, *119*. https://doi.org/10.1016/j.jbi.2021.103842
[45]. Lagani, G., Falchi, F., Gennaro, C., & Amato, G. (2021). Hebbian semi-supervised learning in a sample efficiency setting. *Neural Networks*, *143*. https://doi.org/10.1016/j.neunet.2021.08.003
[46]. Hu, L., Liu, H., Zhang, J., & Liu, A. (2021). KR-DBSCAN: A density-based clustering algorithm based on reverse nearest neighbor and influence space. *Expert Systems with Applications*, *186*. https://doi.org/10.1016/j.eswa.2021.115763

[47]. Moros, J., Cabalín, L. M., & Laserna, J. J. (2022). Refractory residues classification strategy using emission spectroscopy of laser-induced plasmas in tandem with a decision tree-based algorithm. *Analytica Chimica Acta*, *1191*. https://doi.org/10.1016/j.aca.2021.339294

[48]. Aktif Group. (2021, March 27). *Data Mining Decision Trees*. Https://Aktif.Net/En/Data-Mining-Decision-Trees/.

[49]. Haleem, N., Bustreo, M., & del Bue, A. (2021). A computer vision based online quality control system for textile yarns. *Computers in Industry*, *133*. https://doi.org/10.1016/j.compind.2021.103550

[50]. Miraç ÖZTÜRK. (2022, June 13). *Python ile Sınıflandırma Analizleri – Rastgele Orman (Random Forest) Algoritması*. Https://Miracozturk.Com/Python-Ile-Siniflandirma-Analizleri-Rastgele-Orman-Random-Forest-Algoritmasi/.

[51]. Paneru, S., & Jeelani, I. (2021). Computer vision applications in construction: Current state, opportunities & challenges. In *Automation in Construction* (Vol. 132). https://doi.org/10.1016/j.autcon.2021.103940

[52]. Awad, M., & Khanna, R. (2015). *Support Vector Machines for Classification* (pp. 39–66). https://doi.org/10.1007/978-1-4302-5990-9_3

[53]. Minz, P. S., & Saini, C. S. (2021). Comparison of computer vision system and colour spectrophotometer for colour measurement of mozzarella cheese. *Applied Food Research*, *1*(2). https://doi.org/10.1016/j.afres.2021.100020

[54]. Bai, X., Wang, X., Liu, X., Liu, Q., Song, J., Sebe, N., & Kim, B. (2021). Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments. In *Pattern Recognition* (Vol. 120). https://doi.org/10.1016/j.patcog.2021.108102

[55]. Utku Kubilay ÇINAR. (2018, August 13). *Yapay Sinir Ağlarının Bazı Avantajları*. Https://Www.Veribilimiokulu.Com/Yapay-Sinir-Aglari/.

[56]. Sultan, H. H., Salem, N. M., & Al-Atabany, W. (2019). Multi-Classification of Brain Tumor Images Using Deep Neural Network. *IEEE Access*, *7*, 69215–69225. https://doi.org/10.1109/ACCESS.2019.2919122

[57]. Taunk, K., De, S., Verma, S., & Swetapadma, A. (2019). A brief review of nearest neighbor algorithm for learning and classification. *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*. https://doi.org/10.1109/ICCS45141.2019.9065747

[58]. COŞAR, M., & DENİZ, E. (2021). Makine Öğrenimi Algoritmaları Kullanarak Kalp Hastalıklarının Tespit Edilmesi. *European Journal of Science and Technology*. https://doi.org/10.31590/ejosat.1012986

[59]. Kabir Anaraki, A., Ayati, M., & Kazemi, F. (2019). Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. *Biocybernetics and Biomedical Engineering*, *39*(1). https://doi.org/10.1016/j.bbe.2018.10.004

[60]. Sajjad, M., Khan, S., Muhammad, K., Wu, W., Ullah, A., & Baik, S. W. (2019). Multi-grade brain tumor classification using deep CNN with extensive data augmentation. *Journal of Computational Science*, *30*, 174–182. https://doi.org/10.1016/j.jocs.2018.12.003

[61]. Pranshu Sharma. (2021, April 26). *K Means Clustering Simplified in Python*. Https://Www.Analyticsvidhya.Com/Blog/2021/04/k-Means-Clustering-Simplified-in-Python/.

[62]. Cheng, J., Huang, W., Cao, S., Yang, R., Yang, W., Yun, Z., Wang, Z., & Feng, Q. (2015). Enhanced performance of brain tumor classification via tumor region

augmentation and partition. *PLoS ONE*, *10*(10). https://doi.org/10.1371/journal.pone.0140381

[63]. NAZ DOĞDU. (2019, January 18). *Çoklu Doğrusal Regresyon Analizi*. Https://Bilimselanketler.Com/Coklu-Dogrusal-Regresyon-Analizi.

[64]. Shin, H. C., Orton, M. R., Collins, D. J., Doran, S. J., & Leach, M. O. (2013). Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8). https://doi.org/10.1109/TPAMI.2012.277

[65]. z_ai. (2020, February 9). *Logistic Regression Explained*. Https://Towardsdatascience.Com/Logistic-Regression-Explained-9ee73cede081.

[66]. Zhang, B. (2010). *Computer vision vs. human vision*. https://doi.org/10.1109/coginf.2010.5599750

[67]. Alvarez-Valera, H. H., Bolivar-Vilca, E., Cervantes-Jilaja, C., Cuadros-Zegarra, E. E., Barrios-Aranibar, D., & Patino-Escarcina, R. (2016). Automation of Chestnuts Selection Process Using Computer Vision in Real Time. *Proceedings- International Conference of the Chilean Computer Science Society, SCCC*, *2016-September*. https://doi.org/10.1109/SCCC.2014.21

[68]. *Malware Executable Detection*. (n.d.). Https://Www.Kaggle.Com/Datasets/Piyushrumao/Malware-Executable-Detection.

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Ruaa Hussein Hyara
Uyruğu	❒ T.C. ☑ Diğer:
E-Posta Adresi	
Web Adresi	

Eğitim Bilgileri	
Lisans	
Üniversite	Dhi Kar Üniversitesi
Fakülte	Bilim Fakültesi
Bölümü	Bilgisayar Bilimi
Mezuniyet Yılı	2009

Yüksek Lisans	
Üniversite	Kırşehir Ahi Evran Üniversitesi
Enstitü Adı	Fen Bilimleri Enstitüsü
Anabilim Dalı	İleri Teknolojiler Ana Bilim Dalı
Programı	İleri Teknolojiler Tezli Yüksek Lisans
Mezuniyet Tarihi	2022