Yapay Zeka ile Malware Tespiti Final Raporu

Mustafa Akbaba

June 2024

Abstract

Bu Projede makine öğremi ve statik analiz yöntemlerini kullanarak malware tespiti amaçlanmıştır. Kullanılan veri seti ile PE dosyalarını analiz ederek iyi huylu mu kötü huylumu olduğuna karar verir.

1 Giriş

İnternetin hızla gelişmesiyle birlikte kötü amaçlı yazılımlar günümüzde en büyük siber tehditlerden biri haline geldi.Bilgi çalma, casusluk vb. dahil olmak üzere kötü amaçlı eylemler gerçekleştiren herhangi bir yazılım, kötü amaçlı yazılım olarak adlandırılabilir.
Kötü amaçlı yazılımların çeşitliliği artarken, anti-virüs tarayıcıları koruma ihtiyaçlarını karşılayamamakta ve milyonlarca ana bilgisayarın saldırıya uğramasına neden olmaktadır. Kaspersky Labs'a (2023) göre[1], 2023'te Kaspersky'nin sistemleri toplamda yaklaşık 125 milyon zararlı dosya tespit etti. Windows, siber saldırılar için birincil hedef olmaya devam etti ve günlük olarak tespit edilen tüm kötü amaçlı yazılım dolu verilerin yüzde 88'ini oluşturdu. Çeşitli komut dosyaları ve farklı belge formatları aracılığıyla yayılan kötü amaçlı dosyalar, günlük olarak tespit edilen tüm kötü amaçlı dosyaların yüzde 10'unu oluşturarak ilk üç tehdit arasında yer aldı.

2 Literatür Araştırması

Sınıflandırma Algoritması kullanımında Machine Learning approach for Malware Detection(2016) projesinden yardım alınmıştır .[2]Bu yaklaşım, sonuçlarını karşılaştırarak tahmin için hangisinin kullanılacağına karar vermeden önce 6 farklı sınıflandırma algoritmasını dener.
Yapay zekayı kod içinde nasıl kullanılıacağını anlamak için Malware Detection by Machine Learning and Deep Learning(2021) projesinden yardım alınmıştır.[3]Bu çalışmada kötü amaçlı yazlım tespiti için machine learning ve deep learning yöntemleri kullanılmıştır.
Android Malware Detection Using Machine Learning(2022) projesinde konuyu daha iyi kavramak açısından incelemek için kullanılmıştır.[4]Bu projede, Android kötü amaçlı yazılım tespit sorununu çözmeye yönelik farklı yaklaşımlar sunulmakta ve gösterilmektedir.Bu, makine öğrenimi modelleri aracılığıyla elde edilir. En Önemli Makine Öğrenimi algoritmaları andorid haritalama veri kümelerine uygulanır ve ardından android uygulamasının kötü amaçlı yazılım içerip içermediğini tespit etmemizi sağlar.

3 Veri Seti

3.1 Ağ Trafiği Analizi

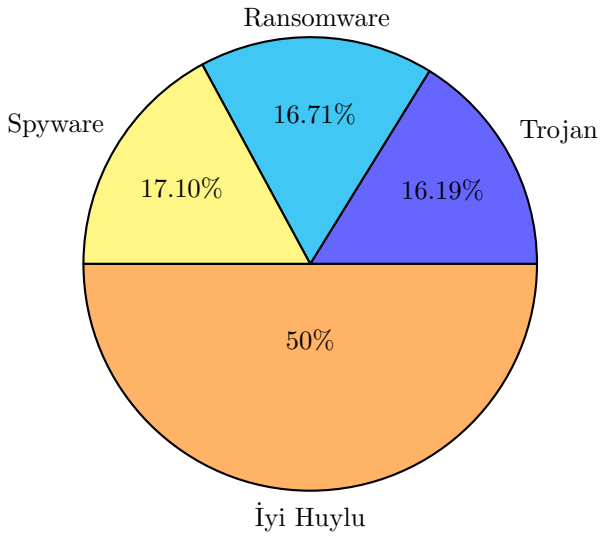
Bu veri seti içeriği ağ trafiği analizi için tipik özellikleri içeriyor. Bu tür veri setleri genellikle ağ trafiği analizi, siber güvenlik tehditlerinin tespiti ve ağ performansı optimizasyonu gibi alanlarda kullanılır.
Veri setinizdeki sütunların içeriği

- Kaynak ve hedef IP adresleri ve bağlantı noktaları
- İletim protokolü
- Zaman damgaları
- İletim süreleri ve paket sayıları
- İletim ve alım yönünde paket boyutları
- İletim ve alım yönünde sürelerin istatistikleri
- TCP kontrol bayrakları (FIN, SYN, RST, vb.)
- Aktivite ve boşta kalma süreleri

Veri setindeki özelliklerin istatistiksel analizi, kötü amaçlı trafiği tanımlamak için tipik örüntülerin belirlenmesine yardımcı olabilir. Örneğin, belirli bir kaynak portundan gelen çok sayıda bağlantı veya belirli bir hedefe yönlendirilen anormal büyüklükteki veri paketleri gibi belirgin örüntüler, potansiyel olarak zararlı trafiği işaret edebilir.[**1**]

3.2 Bellek Analizi

Bu veri seti, bir tür sistem izleme veya güvenlik taraması sonucu elde edilmiş. Her bir özellik, farklı sistem bileşenleri veya davranışlarının belirli özelliklerini temsil ediyor.



Genel olarak, veri seti, bilgisayar sistemlerindeki işlemler, DLL'ler, tanıtıcılar, modüller, hizmetler ve geri aramalar gibi çeşitli bileşenlerin özelliklerini içeriyor.Bu veri seti, bilgisayar sistemlerindeki çeşitli bileşenlerin özelliklerini içerir ve bu özellikler, potansiyel olarak zararlı faaliyetlerin belirlenmesinde ve güvenlik tehditlerinin sınıflandırılmasında kullanılabilir. Örneğin, işlem sayısı, işlem başına tanıtıcı sayısı, DLL sayısı gibi özellikler, sistemin normal veya anormal davranışlarını tanımlamak için kullanılabilir.[5]

4 Yöntem

Kötü amaçlı yazılım tespit teknikleri imza tabanlı ve davranış tabanlı yöntemler olarak ikiye ayrılır. Bu yöntemlere geçmeden önce, kötü amaçlı yazılım analizi için kullanılan yaklaşımının temellerini anlamak önemlidir: statik ve dinamik kötü amaçlı yazılım analizi. Adından da anlaşılacağı üzere, statik analiz "statik olarak", yani dosya yürütülmeden gerçekleştirilir. Buna karşılık, dinamik analiz, dosya yürütülürken, örneğin sanal makinede gerçekleştirilir.

1. İmza Tabanlı Kötü Amaçlı Yazılım Tespiti Kötü amaçlı yazılımları tanımlamak için virüs kodlarını/hash'lerini kullanır. Kötü amaçlı yazılım, onu tanımlamak için kullanılan benzersiz bir kod taşır. Bir dosya bilgisayara ulaştığında, kötü amaçlı yazılım tarayıcısı kodu toplar ve bulut tabanlı bir veritabanına gönderir. Veritabanı geniş bir virüs kodu koleksiyonuna sahiptir. Dosya kodu listede bulunursa, veritabanı dosyanın kötü amaçlı yazılım olduğuna

dair bir karar verir. Anti-malware dosyayı bilgisayardan reddeder ve siler. Yeni bir kötü amaçlı yazılım keşfedilirse, kodu listeye eklenir. ,,

- Artıları: Bu analizi yöntemi, internette yayılan popüler olan kötü amaçlı yazılım türleri söz konusu olduğunda en hızlı ve en doğru yöntemdir.
- Eksileri: Popüler olmayan kötü amaçlı yazılımlar söz konusu olduğunda en doğru yöntem değildir. Doğruluğu kullandığı veri setine bağlıdır.

2. Davranış Temelli Kötü Amaçlı Yazılım Tespiti Kötü amaçlı yazılımları davranışa dayalı olarak tanımlayarak adının hakkını verir. Kötü amaçlı yazılımlar genellikle yasal yazılımlardan farklı davranır. Kötü amaçlı yazılım kendini çalıştırmadan önce bile antivirüs ürünlerine kimliğini açıklayabilecek davranışlar sergileyebilir. Davranış tabanlı tespit, bir yazılımın kötü amaçlı olup olmadığını belirlemek için bu davranışların taranmasını içerir[6].

   - Artıları: Popüler olmayan kötü amaçlı yazılımlar için en iyi sonucu verir.
   - Eksileri: Taramaya gerek kalmadan mümkün olan en kısa sürede kaldırılması gereken kötü amaçlı yazılımları durdurma yeteneğinden yoksundur.

4.1 Makine Öğrenimi ve Derin Öğrenme

4.1.1 Makine Öğrenimi

Makine öğrenmesi, çeşitli algoritmik teknikler kullanılarak farklı makine öğrenmesi modelleri türlerinden oluşur. Verilerin niteliğine ve istenilen sonuca bağlı olarak üç öğrenme modelinden biri kullanılabilir:

- Denetimli:
  Denetimli öğrenmede bağımsız değişkenlerin işaret ettiği bağımlı değişkenin varlığından bahsettiğimiz, bağımsız değişkenleri eğiterek buradan çıkan sonuca referans veren bir yöntemdir.

- Denetimsiz:
  Denetimsiz öğrenmede, algoritma verilerin kendisinin kullanıldığı anlamlı sonuçlar bulmaya çalışır. Sonuç, örneğin, her kümede ilişkilendirilebilecek bir dizi veri noktası kümesi olabilir. Kümeler örtüşmediğinde daha iyi sonuç verir. Birbirine benzer verileri aynı küme altında gruplara ayırır.

- Yarı Denetimli

Bu modellerin her birinde, kullanılan veri kümelerine ve amaçlanan sonuçlara göre bir veya daha fazla algoritmik teknik uygulanabilir. Makine öğrenmesi algoritmaları temel olarak olayları sınıflandırmak, örnekler bulmak, sonuçları tahmin etmek ve bilinçli kararlar vermek için tasarlanmıştır. Algoritmalar karmaşık ve daha öngörülemeyen veriler söz konusu olduğunda mümkün olan en iyi doğruluğu elde etmek için tek seferde bir veya bir arada kullanılabilir[7].

4.1.2 Yaygın Makine Öğrenim Algoritmaları

- Random Forests, (sınıflandırma ve regresyon için)
- Linear regression, en küçük kareler yöntemi(sayısal veriler için),
- Logistic regression (ikili sınıflandırma için)
- Karar ağaçları (sınıflandırma ve regresyon için)

4.2 Feature Engineering

Feature engineering (Özellik Mühendisliği), ham verileri seçme, değiştirme ve denetimli öğrenmede kullanılabilecek özelliklere dönüştürme ve veriyi makine öğrenmesi modellerine hazırlama sürecidir. Feature(Özellik), tahmine dayalı bir modelde kullanılabilecek herhangi bir ölçülebilir girdidir. Doğru olarak işletildiğinde modellerin tahmin gücünün artmasına yardımcı olur[8].

4.2.1 Encoding

Bazı makine öğrenmesi algoritmaları kategorik veriler üzerinde çalışamadığı için sayısal verilere veya vektörlere dönüşmesi gerekmektedir. One hot encoding ve label encoding olarak yöntemleri mevcuttur.

4.2.2 Correlation

Korelasyon analizi; değişkenler arasındaki ilişki, bu ilişkinin yönü ve şiddeti ile ilgili bilgiler sağlayan istatiksel bir yöntemdir. Korelasyon katsayısı, bağımlı değişken ile bağımsız değişkenler arasındaki ilişkinin gücünü gösteren bir katsayıdır. Örneğin; öğrencinin ders çalışma süresi ile aldığı istatistik notu arasında ilişki olup olmadığını veya borsada işlem gören bir hisse senedinin belli bir dönemdeki günlük getirisi (X) ile içinde yer aldığı bir endeksin günlük getirisi (Y) arasındaki ilişki korelasyon katsayısı ile incelenebilir. Korelasyon katsayısı değişkenlerin yönü ve etkileşimlerin nasıl olduğu hakkında bilgi verir[9].

Korelasyon Aralığı	İlişki Düzeyi
(-0,25)-00 ve 0,00-0,25	Çok Zayıf
(-0,49)-(-0,26) ve 0,26-0,49	Zayıf
(-0,69)-(-0,50) ve 0,50-0,69	Orta
(-0,89)-(-0,70) ve 0,70-0,89	Yüksek
(-1,00)-(-0,90) ve 0,90-1,00	Çok Yüksek

4.3 Methodlar

4.3.1 StandartScaler

Standardscaler, veri setinin her bir öznitelik değerinin ortalaması sıfır ve standart sapması bir olacak şekilde yeniden ölçeklendirilmesi işlemidir. Bu işlem, veri setindeki özniteliklerin farklı ölçeklerde olması durumunda, öğrenme algoritmalarının doğru bir şekilde çalışmasına olanak tanır.

Standardscaler, verilerin daha tutarlı bir şekilde işlenmesine olanak tanır ve makine öğrenmesi modellerinin daha iyi performans göstermesini sağlar. Özellikle, çok sayıda öznitelik içeren veri setlerinde, Standardscaler kullanımı önemlidir[10]

4.3.2 .fit()

Bu methodu veri setinde dönüşüm yapılacağında, label encoding yapılacağında veya bir model kurulacağında kullanırız. Bu yöntemlerden her birini uygulamak için bazı parametreler gerekir. Örnek vermek gerekirse eğer bir modelde kullanacaksak veri setinin (ortalama), (standart sapma) değerlerine ihtiyaç vardır. Başka bir örnek, eğer veri setini 0 ile 1 arasında bir normalizasyon işlemi yapılacaksa veri setindeki minimum ve maksimum değerler gerekir. .fit() methodu bu parametleri arka planda hesaplar ancak bu parametrelerle bir işlem yapmaz. Yani bu methodun kendi başına bir çıktısı yoktur.

4.3.3 .transform()

Bu method adından da anlaşılacağı gibi dönüşüm işlemini gerçekleştirir. Ancak bir veri setini fit etmeden transform işlemi uygulanamaz çünkü dönüşüm yapılırken kullanılacak olan parametreler hesaplanmamıştır[11].

4.4 Algoritmalar

4.4.1 Decision Tree

Decision tree algoritması, basitliği ve yorumlanabilirliği nedeniyle sınıflandırma görevleri için popülerdir. Verileri farklı niteliklere göre alt kümelere bölerek ve ağaç benzeri bir karar modeli oluşturarak çalışır. Algoritma, örnekleri sınıflandırmak için if-else koşullarının hiyerarşik yapısını kullanır. Decision tree, finans, sağlık ve pazarlama gibi çeşitli alanlarda yaygın olarak kullanılmaktadır.

4.4.2 Random Forest

Random forest, tahminlerin doğruluğunu artırmak için birden fazla karar ağacını birleştiren bir toplu öğrenme yöntemidir. Birçok karar ağacı oluşturarak ve çıktılarını oylama yoluyla toplayarak çalışır. Ormandaki her karar ağacı, eğitim verilerinin rastgele bir alt kümesi ve girdi özelliklerinin rastgele bir alt kümesi kullanılarak oluşturulur. Random Forest, sağlamlığı ve yüksek boyutlu özelliklerle büyük veri kümelerini işleme yeteneği ile bilinir.

4.4.3 Support Vector Machines

SVM, sınıflandırma ve regresyon görevleri için güçlü bir algoritmadır. SVM, sınıfları maksimum düzeyde ayıran yüksek boyutlu bir uzayda optimal bir hiper düzlem bulur. Girdi verilerini daha yüksek boyutlu bir öznitelik uzayına dönüştürür ve en büyük kenar boşluğuna sahip optimum ayırıcı hiper düzlemi bulur. SVM'ler görüntü tanıma, metin sınıflandırma ve biyoinformatikte yaygın olarak kullanılır.

4.4.4 K-Nearest Neighbors (KNN)

KNN, sınıflandırma için kullanılan parametrik olmayan ve tembel bir öğrenme yöntemidir. Örnekleri, eğitim verilerindeki k en yakın komşuya olan yakınlıklarına göre sınıflandırır. KNN basit ama etkilidir ve temel veri dağılımı hakkında herhangi bir varsayımda bulunmaz. Genellikle tavsiye sistemlerinde, anormallik tespitinde ve örüntü tanımada kullanılır.

4.4.5 Gradient Boosting

XGBoost ve LightGBM gibi Gradient Boosting algoritmaları, zayıf öğrenicileri birleştirerek güçlü bir tahmine dayalı model oluşturan toplu öğrenme yöntemleridir. Önceki modellerin yaptığı hataları düzelten modelleri sırayla ekleyerek iteratif olarak çalışırlar. Gradient Boosting algoritmaları, web arama sıralaması, kredi riski analizi ve dolandırıcılık tespiti dahil olmak üzere çeşitli alanlarda mükemmeldir.

4.5 Model Performan Değerlendirmesi

4.5.1 Doğruluk (Accuracy)

Modelin doğru tahmin ettiği örneklerin oranıdır. Doğruluğun maksimum değeri 1 olabilir. Örneğin, 100 örneğin 80'inin doğru sınıflandırıldığı bir modelin doğruluk skoru 0.8 olacaktır.
Formül:

Doğruluk = (TP + TN) / (TP + TN + FP + FN)

4.5.2 Hassasiyet (Precision)

Pozitif olarak sınıflandırılan örneklerin ne kadarının gerçekten pozitif olduğunu gösterir. Hassasiyet modelin pozitif sınıfı doğru sınıflandırma yeteneğini ölçmektedir. Aşağıdaki formülden de anlaşılacağı üzere hassasiyet pozitif olarak doğru bilinen tahminlerin tüm pozitif tahminlere oranı olarak ifade edilebilir.
Formül:
Hassasiyet = TP / (TP + FP)

4.5.3 Duyarlılık (Recall)

Gerçek pozitif örneklerin ne kadarının pozitif olarak sınıflandırıldığını gösterir. Aşağıdaki formül incelendiğinde duyarlılık, pozitif olarak doğru tahmin edilenlerin gerçek pozitiflere oranı olarak ifade edilebilir.
Formül:
Duyarlılık = TP / (TP + FN)

4.5.4 F1 Skor (F1 Score)

F1 Skor ise Hassasiyet (Precision) ve Duyarlılık (Recall) skorlarının harmonik ortalamasıdır.
Formül:
F1 Skor = 2 * ((Precision * Recall) / (Precision + Recall))[12]

4.6 Portable Executable

Portable Executable (kısaca PE), Windows' un çalıştırılabilir dosya formatıdır. Dosya formatları , programların fonksiyonlarla alakalı bir takım bilgileri analiz yapan kişiye sunabilmektedir. PE dosya formatı Windows executable ve DLL dosyaları tarafından kullanılabilmektedir. PE dosyaları başlık bilgileri ile başlar ve bu başlık bilgileri program için gereken kütüphane fonksiyonlarını uygulama tipleri gibi bilgileri içermektedir.Bağlı kütüphaneler ve fonksiyonlardan yararlanılarak analiz için bilgiler edilebilir. PE dosya başlığı program tarafından yüklenen kütüphane ve fonksiyonlar ile alakalı bilgileri tutar. [13]

4.6.1 File Header

Bu başlık, PE dosyasına ait önemli birkaç bigiyi tutuyor.

- Machine
- SizeOfOptionalHeader
- Characteristics

4.6.2 Optional Header

programın entry pointi, işletim sisteminin versiyonu gibi çok kritik blgiler taşır. Bu kısmın 32 bit ve 64 bit versiyonları var fakat sadece boyut olarak farklılıklar barındırıyor.

- MajorLinkerVersion
- MinorLinkerVersion
- SizeOfCode
- SizeOfInitializedData

4.6.3 Section

PE dosyası sectionlardan oluşur. Bu sectionlar içindeki verinin nasıl bir yapıda olduğunu belirtir.Sadece global ve statik veriler section'larda saklanıyor. Lokal değişkenler .data, .bss gibi section'larda saklanmıyor. Bunun yerine ptogramı execute ettiğimizde çalışma zamanında oluşturulup stack te depolanır ve gerektiğinde stack'ten okunup gerekli işlem yapılır.

4.6.4 File Import

Bildiğiniz üzere yazılan programlarda başka .dll' lerde ki fonksiyonları kullanıyoruz. Bu fonksiyonları kullanabilmek için kullandığımız dll'lerin kayıtlarının PE dosya formatında bir yerde bulunması gerekiyor. Aksi halde bu fonksiyonları kullanamayız. İşte bu kayıtlar ".idata" section'ında yer alıyor. Loader(PE dosyamızı rame map eden program) .idata section'ındaki bilgileri kullanarak gerekli dll'lerin ve fonksiyonların adreslerini executable image'ye bağlar.

4.6.5 File Export

Biz normalde başka .dll'lerin içerisindeki fonksiyonları kullanıyoruz (import), export bunun tam tersi bizim dışarıya kullanılabilecek bir fonksiyon vermemiz denebilir.

4.6.6 File Resources

Portable Executable (PE) dosya formatındaki kaynaklar, bir uygulamanın veya yürütülebilir dosyanın içinde bulunan metin, grafik, ses, ikonlar ve diğer öğeleri içeren veri parçacıklarıdır. Bu kaynaklar, bir uygulamanın görsel ve işitsel unsurlarını, dil seçeneklerini, simgelerini ve diğer öğelerini içerir. PE dosya formatında kaynakları temsil etmek için özel bir bölüm vardır ve bu bölüm "RESOURCE" adını taşır.PE dosyalarındaki kaynaklar, çeşitli özelliklere ve türlerde verilere sahip olabilir. Bu kaynaklar, uygulamanın kullanıcı arayüzünü, dil seçeneklerini, ikonlarını, resimlerini, metin kaynaklarını ve daha fazlasını içerebilir. RESOURCE bölümü, kaynakların düzenli bir şekilde depolanmasını sağlar ve uygulamaların bu kaynaklara erişimini kolaylaştırır. [14]

5 Kod Bölümleri

5.1 get entropy

Shannon entropy, bir veri dizisinin düzensizliğini veya belirsizliğini ölçen bir metriktir. Daha yüksek bir entropi, verinin daha fazla belirsizlik veya düzensizlik içerdiği anlamına gelir.

- 0 * 256: Bir liste oluşturur ve bu listenin her elemanı 0 olan 256 eleman içerdiği anlamına gelir.
- isinstance(x, int): x değişkeninin bir tamsayı olup olmadığını kontrol eder.
- ord(x): Eğer x bir tamsayı değilse, yani bir karakter (char) ise, ord() fonksiyonu kullanılarak karakterin ASCII değerine dönüştürülür.

5.2 get resources

Verilen PE (Portable Executable) dosyasından kaynak bilgilerini (resources) çıkarır. Bu kaynaklar, dosyanın içerdiği gömülü verileri temsil eder.

5.3 get version info

PE dosyasının sürüm bilgilerini çıkarır. Dosyanın sürüm numarası, imzası ve diğer özelliklerini içerir.

5.4 extract info

PE dosyasının çeşitli özelliklerini çıkarır. Bu özellikler, dosyanın boyutu, bölümleri, import ve export sayıları, kaynakları ve sürüm bilgilerini içerir.

5.5 İçe Aktarım

Bir yazılım bileşeninin, kendi kodunda başka bir bileşeni kullanmak için dışarıdan içe aktarmasıdır. Bu, genellikle bir yürütülebilir dosyanın veya bir DLL'nin bir işlevini veya hizmetini kullanmak için diğer bir yürütülebilir dosya veya DLL'yi içe aktarması anlamına gelir. Örneğin, bir programın kullanıcı arayüzü için Windows API işlevlerini kullanması veya bir DLL'nin işlevlerini çağırması, içe aktarımı temsil eder.

5.6 Dışa Aktarım

Bir yazılım bileşeninin, kendi kodunda bulunan bir işlevi veya hizmeti diğer bileşenlere sunmasıdır. Bu, genellikle bir yürütülebilir dosyanın veya bir DLL'nin, kendi işlevlerini diğer bileşenlerin kullanımına sunması anlamına gelir. Örneğin, bir DLL'nin belirli işlevlerini diğer programlar kullanabilmek için dışa aktarması, dışa aktarımı temsil eder[15][16][17][18][19][20].

6 Bulgu ve Tartışma

Tablolarda görülen sonuçlar yapay zeka modelinin farklı parametlerini değiştirerek doğruluk oranı arttırılmıştır. Bu parametreler şunlardır:

6.1 n Estimators

n Estimators parametresi, Random Forest ve Gradient Boosting gibi ağaç tabanlı ensemble öğrenme modellerinde kullanılan bir parametredir. Bu parametre, modele dahil edilecek karar ağaçlarının sayısını kontrol eder.
n Estimators seçiminde göz önünde bulundurulması gereken bazı faktörler şunlardır:

- Veri kümesinin boyutu: Daha büyük veri kümeleri, daha fazla karar ağacına izin verebilir.
- Veri kümesinin karmaşıklığı: Daha karmaşık veri kümeleri, daha fazla karar ağacı gerektirebilir.
- Modeli kullanılacak görev: Sınıflandırma görevleri, regresyon görevlerinden daha fazla karar ağacı gerektirebilir.

n Estimators'ın bazı yaygın değerleri şunlardır:

- 100: Bu, birçok uygulama için iyi bir başlangıç noktasıdır.
- 1000: Bu, daha karmaşık veri kümeleri için daha uygundur.
- 10000: Bu, çok karmaşık veri kümeleri için kullanılabilir.

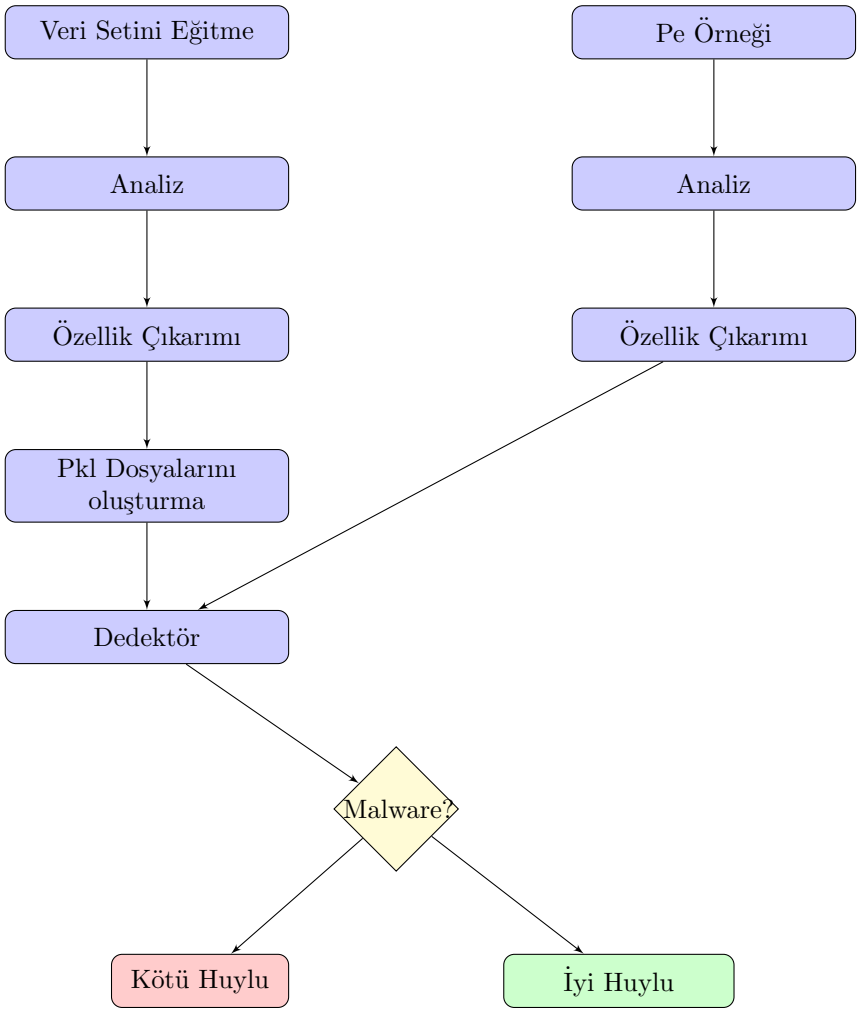


Figure 1: Kodun Çalışma Mantığı

Table 1: Uygulama Test Sonuçları Model Versiyon 1

	Test Edilen Uygulama	Doğru Sonuç	Doğruluk
İyi Huylu	50	33	66
Malware	18	18	100

Table 2: Uygulama Test Sonuçları Versiyon 2

	Test Edilen Uygulama	Doğru Sonuç	Doğruluk
İyi Huylu	50	38	76
Malware	18	18	100

6.2 max depth

max depth, karar ağaçları ve rastgele ormanlar gibi ensemble öğrenme modellerinde kullanılan bir hiperparametredir. Bu parametre, bir karar ağacının veya alt ağacın maksimum derinliğini kontrol eder. Derinlik, bir düğümden kök düğüme giden en uzun yol uzunluğu ile ölçülür.
max depth'in bazı yaygın değerleri şunlardır:

- 2: Bu, birçok uygulama için iyi bir başlangıç noktasıdır.
- 5: Bu, daha karmaşık veri kümeleri için daha uygundur.
- 10: Bu, çok karmaşık veri kümeleri için kullanılabilir.

6.3 n jobs

n jobs, Python'daki joblib kütüphanesi tarafından kullanılan bir parametredir. Bu parametre, birden fazla CPU çekirdeği veya işlemciyi kullanarak kodun paralel olarak yürütülmesine izin verir.
n jobs'u kullanmanın faydaları:

- Hesaplama süresini önemli ölçüde kısaltabilir. Özellikle büyük veri kümeleri üzerinde çalışırken, paralel işleme önemli bir hız artışı sağlayabilir.
- Bilgisayarınızın kaynaklarını daha verimli kullanabilirsiniz. n jobs, tüm işlemci çekirdeklerinizi kullanarak kodunuzu çalıştırarak bilgisayarınızın tüm potansiyelinden yararlanmanızı sağlar.
- Bazı karmaşık hesaplamaları daha pratik hale getirebilir. Bazı işlemler tek bir işlemci çekirdeğinde çok uzun sürebilir. n jobs, bu işlemleri birden fazla çekirdeğe bölerek daha kısa sürede tamamlanmasını sağlayabilir..

Projede doğruluk oranını arttırmak için model parametlerini daha detaylı bir şekilde incelemek ve verisetini daha fazla sütun eklemek işe yarayacaktır.

Projenin başlıngıcından itibaren planlanan süreçlerde bazı değişiklikler oldu. Kod kısmı beklenenden daha fazla süre aldı.

7 Sonuç

Bu proje , makine öğrenimi teknikleri kullanılarak malware tespiti gerçekleştirilmiştir. Çeşitli modeller arasından en yüksek performansı gösteren RandomForest modeli, 94/100 doğruluk oranına ulaştı. Özellikle max depth, n jobs ve n estimator gibi özellikler model performansını artırmıştır.

Modelin eğitim ve test verilerindeki tutarlı performansı, genelleme yeteneğinin yüksek olduğunu göstermektedir. Ancak, belirli malware türlerinde iyileştirme gerekmektedir. Gelecekte, daha büyük ve çeşitli veri kümeleri ile derin öğrenme tekniklerinin kullanılması, performansı daha da artırabilir.

Sonuç olarak, makine öğrenimi teknikleri, malware tespiti konusunda etkili bir çözüm sunmaktadır ve bu alanda daha ileri çalışmalar için temel oluşturulmuştur.

References

[1] Kaspersky, *Rising Threat*, Kaspersky, **2023**.

[2] Surajr, *Machine Learning approach for Malware Detection*, Github, **2016**.

[3] A. Gull, *Malware Detection by Machine Learning and Deep Learning*, Youtube, **2021**.

[4] Vatshayan, *Android Malware Detection Using Machine Learning*, Github-Youtube, **2022**.

[5] T. Carrier, *Malware Memory Analysis*, Kaggle, **2022**.

[6] K. Dhruw, *Anomaly Based Malware Detection – 1*, KIIOCITY, **2022**.

[7] SAP, *Makine öğrenmesi nedir?*, SAP, **2021**.

[8] N. S. Özakca, *Feature Engineering Nedir? Kısa Bakış*, Medium, **2022**.

[9] E. E. ÖZTÜRK, *Korelasyon Analizi(r) Nedir?*, VBO, **2020**.

[10] A. Asutay, *Standardscaler Nedir?*, Aslı Asutay, **2023**.

[11] Ramiscanyakar, *Sci-Kit Learn .fit(), .transform(), .fit$_t$ransform()MethodlarıFark*ı, Medium, **2020**.

[12] M. Erdogan, *Makine Öğrenmesi Sınıflandırma Modelleri: Accuracy, Precision, Recall, F1-score, Log Loss and Micro-Macro-Weighted Avg*, Medium, **2023**.

[13] A. PAYASLIOĞLU, *Portable Executable Dosya Formatı Üzerinden Malware Analizi*, Medium,https://ahmetpayaslioglu.medium.com/pe-portable-executable-dosya-format, **2020**.

[14] wintersoldier1903, *PE (Portable Executable) Nedir ?*, Turk Hack Team,https://www.turkhackteam.org/ko portable-executable-nedir.2055135, **2024**.

[15] umasolution, *ML-malware-detection*, Github,https://github.com/umasolution/ML-malware-detection/blob/master/checkmanalyzer.py.org, **2019**.

[16] wintersoldier1903, *PE (Portable Executable) Nedir ?*, Turk Hack Team,https://www.turkhackteam.org/ko portable-executable-nedir.2055135, **2024**.

[17] E. Carrera, *Pefile*, Github,https://github.com/erocarrera/pefile/blob/master/pefile.py, **2023**.

[18] Stanislav-Povolotsky, *peanalyzer*, Github,https://github.com/blacknbunny/peanalyzer/blob/master/pea **2018**.

[19] A. Swanda, *pe-static*, Github,https://www.turkhackteam.org/konular/pe-portable-executable-nedir.2055135, **2018**.

[20] M. Akbaba, *ChatGPT*, OpenAI,Version 3.5, **2024**.