Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi
Bilecik Seyh Edebali University Journal of Science
10(1), 140-152, 2023
DOİ: 10.35193/bseufbd.1194460
e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

Araştırma Makalesi - Research Article

XGBoost Algoritması ile İkili Parçacık Sürü Optimizasyonu Öznitelik Seçme Tabanlı Jar Kötü Amaçlı Yazılımlarının Tespiti

Jar Malware Detection with XGBoost Algorithm Based on Binary Particle Swarm Optimization Feature Selection

Mahmut Tokmak[1*]

Geliş / Received: 25/10/2022 ***Revize / Revised: 05/12/2022*** ***Kabul / Accepted: 27/01/2023***

ÖZ

Java dilini kullanan kötü amaçlı yazılımlarla gerçekleştirilen saldırılar, geçtiğimiz yıllarda hızla artış göstermeye başlamıştır. Bu artışlarla birlikte kötü amaçlı yazılımların kişilere ve kurumlara verebileceği zararlar araştırmacıları otomatik algılama sistemlerini geliştirerek güçlendirmek için farklı makine öğrenme teknikleri geliştirmeye ve test etmeye yöneltmiştir. Bu çalışmada kötü amaçlı Jar dosyalarının tespiti için ikili parçacık sürü optimizasyonu tabanlı öznitelik seçimi ve XGBoost algoritması ile sınıflandırma yapan hibrit bir sistem önerilmiştir. İkili parçacık sürü optimizasyonu algoritmasında minimizasyon sağlanırken kullanılan uygunluk fonksiyonunda rastgele orman algoritması kullanılmıştır. Öznitelik seçimi ile sınıflandırma algoritmasının üzerine düşen hesaplama yükü azaltılarak hız ve performans artırımı hedeflenmiştir. Önerilen modelde 10 kat çapraz doğrulama yapılarak eğitim ve testler gerçekleştirilmiştir. XGBoost algoritması ile yapılan tespit mekanizmasında doğruluk, kesinlik, F1-Skoru, duyarlılık metrikleri ile kurulan modelin performansı ortaya konulmuştur. Önerilen modelin performansının değerlendirilmesi amacıyla AdaBoost, Gradient Boosting, Destek Vektör Makineleri, Yapay Sinir Ağları, Naive Bayes yöntemleri ile testler yapılmış ve sonuçlar karşılaştırılmıştır. Deneysel sonuçlar, önerilen ikili parçacık sürü optimizasyonu tabanlı öznitelik seçimi ve XGBoost algoritması ile sınıflandırma yapan hibrit modelin kötü amaçlı Jar yazılım tespitinde %98.04 doğruluk oranı ile karşılaştırılan modellere göre daha başarılı olduğunu göstermiştir.

Anahtar Kelimeler- Jar Malware Tespiti, XGBoost, İkili Parçacık Sürü Optimizasyonu, Öznitelik Seçme

ABSTRACT

Attacks with malware using the Java language have increased rapidly in recent years. With these increases, the damage that malware can cause to people and institutions has led researchers to develop and test different machine learning techniques to improve and strengthen automatic detection systems. In this study, a hybrid system is proposed for the detection of malicious Jar files, which uses binary particle swarm optimization based feature selection and classification with XGBoost algorithm. While minimizing is achieved in the binary particle swarm optimization algorithm, the random forest algorithm is used in the fitness function. With feature selection, it is

[1]*Sorumlu yazar iletişim: mahmuttokmak@mehmetakif.edu.tr (https://orcid.org/0000-0003-0632-4308)
Yönetim Bilişim Sistemleri, Burdur Mehmet Akif Ersoy Üniversitesi, Bucak Zeliha Tolunay Uygulamalı Teknoloji ve İşletmecilik Yüksekokulu, Burdur, Türkiye

aimed to increase speed and performance by reducing computational load on classification algorithm. In proposed model, training and tests were carried out by performing 10-fold cross validation. In detection mechanism made with the XGBoost algorithm, the performance of the model established with Accuracy, Precision, F1-Score, Recall metrics has been demonstrated. In order to evaluate performance of the proposed model, tests were made with AdaBoost, Gradient Boosting, Support Vector Machines, Artificial Neural Networks, Naïve Bayes methods and the results were compared. Experimental results showed that the proposed binary particle swarm optimization-based feature selection and hybrid model classifying with the XGBoost algorithm was more successful in detecting Jar malware than the compared models with an accuracy rate of 98.04%.

Keywords- Jar Malware Detection, XGBoost, Binary Particle Swarm Optimization, Feature Selection

I. GİRİŞ

Java programlama dili, dünya çapında şirketler tarafından dahili ve ticari uygulamalar geliştirmek için en çok tercih edilen dillerden biridir [1]. TIOBE Index olarak da bilinen TIOBE programlama topluluğu endeksine göre, Mayıs 2022'de Java %10.99'luk bir derecelendirme ile en yüksek puan alan programlama dillerinden biri olarak değerlendirilmektedir [2]. Java teknolojisi, çeşitli işletim sistemlerinde ve bilgisayar kontrollü cihazlarda yürütülebilecek uygulamaların geliştirilmesine odaklanmaktadır. Bu nedenden dolayı Java uygulamaları işletim sisteminden bağımsızdır ve taşınabilir özelliktedir. Java uygulamaları JVM (Java Sanal Makinesi) içeren herhangi bir işletim sisteminde yürütülebilmektedirler [3]. Java'nın bu özellikleri kullanım oranını ve popülaritesini artırmaktadır. Java'nın kurumsal web sitesinde bildirdiğine göre; şirket masaüstü bilgisayarlarının %97'sinde bulunan Java'nın dünya genelinde 9 milyon geliştiricisi bulunmaktadır. Bunun yanı sıra 3 milyar cep telefonunda ve 125 milyon TV cihazında Java bulunmaktadır. Dizüstü bilgisayarlardan veri depolama merkezlerine, oyun konsollarından süper bilgisayarlara, cep telefonlarından internete kadar geniş bir yelpazede kullanılıyor olması popülaritesini destekleyen göstergeler olarak ifade edilebilmektedir [4].

Java'nın kullanım istatistiklerden de gözlemlendiği üzere Java oldukça popüler bir programlama dilidir. Bu nedenle, diğer popüler programlama dilleri gibi, yazılım saldırganları tarafından kötü amaçlı yazılımlar oluşturmak için de yaygın olarak kullanılmaktadır [1]. Özellikle son yıllarda Java tabanlı kötü amaçlı yazılım kullanan saldırı olayları artış göstermektedir [2]. Java kötü amaçlı yazılımları genellikle teknik olarak uç nokta denilen kişisel bilgisayarları ve cihazları hedeflemektedirler. Enfekte olduğu uç noktada, internette gezinme veya yardımcı programları kullanma gibi günlük etkinliklerden kötü niyetli olarak yararlanabilmektedirler. Bu siber zararlılar nedeniyle banka şifreleri, sosyal ağlar, fotoğraflar veya özel videolar ele geçirilebilmektedir [2]. Bu saldırılara karşı koymak için kullanılabilecek güvenlik yazılımlarının kişisel olarak kullanımının yanında büyük şirketlerde konuşlandırıldığı dikkate alındığında, bu tür teknolojilerin çok hassas, yüksek algılama oranı ve sıfır hatalı pozitif olması gerekmektedir. Zira bu tür şirketlerde, zararlı olmayan bir dosyaya erişiminin kötü amaçlı olarak algılanarak engellenmesi, iş akışını veya üretimini kesintiye uğratabilir. Bunu bilen kötü amaçlı yazılım geliştiricileri, zararlı olmayan dosyalara çok benzeyen dosyalar geliştirerek bu tür saldırıların kesin olarak tespit edilmesinin zorluğunu artırmaktadırlar [1].

Son zamanlarda, Java tabanlı kötü amaçlı yazılımları içeren saldırılar, kötü amaçlı eklentiler veya oltalama e-postaları aracılığıyla gerçekleştirilmektedir [1,3]. Bu kötü amaçlı yazılım dosyaları, virüslü sistemde çalıştırılabilen Jar (Java Arşivi) dosyalarıdır. Jar dosyaları, ZIP formatına dayalı çapraz platform paket dosya formatıdır. Jar formatı, farklı proje bileşenlerini (Java sınıfı dosyaları, kaynaklar ve ilişkili meta veriler) uygun dağıtım için tek bir arşivde toplamayı amaçlar. Bir Jar dosyası, o platformda Java Runtime Environment (JRE) kurulu olduğu sürece o platformda çalışabilir [1,3,4].

2020 gibi yakın bir geçmişte bilgisayar korsanları, bilgisayarlara Java tabanlı kötü amaçlı yazılımları bulaştırmak için en popüler etkileşimli COVID-19 güncelleme haritalarından birini kullanmışlardır [3]. Birkaç siber suç forumunun üyesi Java tabanlı bir kötü amaçlı yazılım dağıtım planının bir parçası olarak Hopkins etkileşimli haritasını kullanan dijital bir Koronavirüs enfeksiyon kiti satmaya başlamışlardı. Satıcı "Koronavirüs bulaşmış bölgelerin ve diğer verilerin tamamen çalışan çevrimiçi haritası" diye kullanıcıya açıklama getiriyordu [5]. Bu uygulamanın zararsız bir harita olduğunu düşünen kullanıcılar uygulamayı çalıştırdıklarında Jar dosyaları altındaki kötü niyetli Java uygulamaları kullanıcıların bilgisayarlarına yükleniyor ve viral olmuş oluyorlardı. Bu dosyalar, Koronavirüs için etkileşimli, gerçek zamanlı bir veri haritası ve virüslü sistemlerden hassas bilgileri çalmak için bir payload içermekteydi [3,5].Bahse konu olan nedenlerden dolayı, kötü amaçlı bir Java derlenmiş örneğini tespit etmek için çeşitli yöntemler geliştirmek bir zorunluluk haline gelmiştir. Kötü amaçlı yazılımdan koruma endüstrisinde veya virüsten koruma yazılımlarında kötü amaçlı yazılım algılamaya yönelik geleneksel yaklaşımlar, kötü amaçlı yazılım dosyalarını algılamak için bilinen kötü amaçlı yazılım örneklerinin imzalarını kullanmaktadırlar [6,7]. Ancak günümüzde geleneksel yöntemlerin yerine, makine öğrenme (Machine Learning:

ML) yöntemlerinin kullanıldığı kötü amaçlı yazılım tespit mekanizmaları temel olarak statik analiz, dinamik analiz ve hibrit yöntemler başlıkları altında kategorize edilmektedir [3,8].

Statik analiz yöntemi, tersine mühendislik tekniklerine dayandığı yani kaynak kodların incelenmesi ilkesini benimsediği için herhangi bir gerçek cihaz ya da sanal cihazda çalıştırılmasına gerek duyulmamaktadır. Bu sayede zararlı bir uygulama incelemesi esnasında uygulamanın cihaza zarar verme durumu olmamaktadır. Bu statik analiz yönteminin bir avantajı olarak görülmektedir. Ancak uygulamada kodların anlaşılabilirliğini zorlaştırmak için kod gizleme (obfuscation) gibi yöntemler kullanılmış ise statik analiz yöntemi ile başarı elde edilememektedir ki bu da statik analiz yönteminin dezavantajı olarak görülmektedir [9,10]

Dinamik analiz yöntemi uygulamaların gerçek bir cihaz ya da bir sanal cihaz kullanılarak çalıştırılması ve uygulamanın çalışma zamanındaki davranışlarının takip edilmesi esasına dayanmaktadır. Uygulama çalıştırıldıktan sonra uygulamanın sistem üzerindeki etkileri, ağ üzerindeki davranışları, API çağrıları, batarya kullanımı gibi hareketleri takip edilerek geliştirilen yazılımın analizi mümkün olabilmektedir. Statik analiz yönteminde analizi zorlaştırmak için kullanılan kod gizleme yöntemi dinamik analiz yöntemi ile etkisini yitirmiş olacak, analiz esnasında herhangi bir etkisi kalmayacaktır. Bu açıdan dinamik analiz yöntemi statik analiz yöntemine göre avantaj sağlamaktadır. Ancak dinamik analiz yönteminde uygulama çalıştırılarak analiz edildiğinden eğer uygulama zararlı bir uygulama ise gerçek cihazlarda çalıştırıldığında cihaz bundan etkilenecek, zarar görecektir. Bu nedenle uygulamalar sandbox (kum havuzu-sanal cihaz-emülator) adı verilen sistemlerde çalıştırılması daha uygun görülmektedir. Sanal, izole edilmiş sistemlerin oluşturulması ise maliyet ve zahmet gerektiren bir yöntem olarak karşımıza çıkmaktadır. Aynı zamanda dinamik analiz yöntemi statik analize göre süre açısından uzun sürmektedir [9,11].

Hibrit analiz yöntemi, statik analiz yöntemi ve dinamik analiz yönteminin beraber kullanıldığı bir yöntemdir. Hibrit analiz yöntemi genel olarak iki aşamadan oluşmaktadır. Bu aşamalar, statik özelliklerin çıkarıldığı ilk aşama ve dinamik olarak çıkarılan özelliklerin elde edildiği ikinci aşama şeklindedir [12]. Literatürdeki çalışmalar ışığında örneklendirilecek olursa; izin ve API tabanlı bir çalışma, hibrit analiz yöntemi ile analiz yapacaksa; birinci aşamada manifest dosyasından izinleri statik olarak çıkarmakta ve ikinci aşamada emülatörde uygulama çalıştırılarak API çağrılarını dinamik olarak elde etmektedir. Hibrit analiz yöntemi ile hedeflenen ise statik analiz yöntemi ve dinamik analiz yönteminin dezavantajlarından kaçınarak doğru sonuçlara ulaşabilmektir.

Yapılan Java tabanlı kötü amaçlı yazılım tespit çalışmalarında, ML algoritmaları sıkça kullanılmıştır. Kumar ve Vaishakh [4] kod gizleme yöntemini kullanan Java kötü amaçlı yazılımlarının tespitinde statik analiz yöntemini benimseyerek Destek Vektör Makineleri (Support Vector Machine: SVM), Naive Bayes (NB), C4.5, Random Forest (RF) algoritmalarını kullanmışlardır. Balan ve Popescu [1] kötü amaçlı Jar dosyalarının tespiti için statik analiz yapmışlar ve hiyerarşik kümeleme algoritmasını kullanmışlardır. Obaidat vd. [3] statik analiz ile image tabanlı, derin öğrenme yöntemlerinden Evrişimsel Sinir Ağlarını (ESA) kullanmışlardır. Pinheiro vd. [13] kötü amaçlı Jar dosyalarının tespiti için davranışsal yaklaşımla öznitelikleri elde etmişler ve En İyi Eşlenik Gradyan algoritmasıyla sınıflandırma yapmışlardır. Aynı zamanda kullanmış oldukları veri setini paylaşarak yaptıkları diğer çalışmada ise Aşırı Öğrenme Makinelerini kullanmışlardır [2]. Herrera ve Cheney [14], dinamik analize dayanan kötü amaçlı yazılım tespit sistemi önermişlerdir. Önerilen metodoloji, Java tabanlı kötü amaçlı yazılım tespiti için sembolik yürütme, enstrümantasyon ve dinamik analiz olmak üzere üç tekniği birleştirmiştir. Jha vd. [15] Deepmal4j adını verdikleri çalışmasında derin öğrenme tabanlı tespit sistemi önermişlerdir. Uzun kısa süreli bellek (Long short-term memory: LSTM) ve geçitli tekrarlayan birim mimarisi (Gated Recurrent Units: GRU) kullanarak Java kötü amaçlı yazılımlarını tespit etmeye çalışmışlardır.

Öznitelik seçimi, bir uzman sistemin sınıflandırma kısmında önemli bir ön işleme yöntemidir [16]. Genel olarak, öznitelik seçimleri, orijinal veri kümesinin bir alt kümesinin optimizasyon problemine bir çözümle temsil edildiği optimizasyon problemleri olarak kabul edilebilir ve bu problemler kapsamlı ve sezgisel arama yaklaşımları ile çözülebilir [17]. Bununla birlikte, özellikle yüksek veri boyutlarına sahip veri kümesi için kapsamlı arama yöntemleri kullanıldığında hesaplama maliyetleri genellikle kabul edilemez. Bu nedenle, öznitelik seçimi problemlerini çözmek için sezgisel yaklaşımlar daha makul yöntemler olarak kabul görmektedir. Sürü zekası (Swarm Intelligence: SI) algoritmaları, sarmalayıcı tabanlı öznitelik seçim problemleri için verimli buluşsal arama yöntemleridir [18]. ML yöntemlerinde, "değişkenler ne kadar fazlaysa, performans o kadar iyi" inancı bir süredir kabul edilebilir değildir; bu nedenle, öznitelik seçim tekniklerinin uygulanması veri madenciliği alanında hızla popülerlik kazanmaktadır. Makine öğreniminde farklı teknikler ve çeşitli yaklaşımlar (denetimli sınıflandırma veya regresyon, denetimsiz kümeleme, vb.) olmasına rağmen, bir dizi özniteliğin (veya özelliğin) değerleriyle temsil edilen örnekler veri kümesi ortaktır. ML algoritmalarının asıl amacı, verideki örüntüleri arayarak özelliklerin değerleri ile istenen sonuç (sınıflandırma veya regresyon modeli, küme kümesi vb.) arasında mümkün olan en iyi eşleştirmeyi bulmaktır. Verinin boyutu arttığında, hesaplama maliyeti de genellikle üstel olarak artar.

Bu sorunun üstesinden gelmek için, dikkate alınan öznitelik sayısını azaltmanın bir yolunu bulmak gerekir. Genellikle iki teknik sıklıkla kullanılır: öznitelik seçimi ve öznitelik çıkarma. Öznitelik çıkarma, boyutluluğu azaltmak için diğerlerinin kombinasyonları olarak yeni değişkenler yaratır. Öznitelik seçme ise ilgili olmayan veya gereksiz özellikleri kaldırarak çalışır. Bu şekilde, bilgiyi mümkün olduğunca koruyan ve genellikle ML algoritmalarının daha iyi eşlemeleri daha verimli bulmasını sağlayan daha az özniteliğin üzerine verilerin bir projeksiyonunu arar [17]. Son yıllarda popüler hale gelen sarmalayıcı algoritmaların bir sınıfı, SI algoritmalarıdır. SI algoritmaları arasında, parçacık sürü optimizasyonu (PSO), öznitelik seçimi için en yaygın olarak kullanılan algoritmadır. Kötü amaçlı yazılım sınıflandırma alanında dikkate değer bir başarıyla uygulanmıştır [18–21] .

Bu çalışmada, kötü amaçlı Jar zararlılarının tahmini için XGBoost algoritması seçilmiştir. Çünkü XGBoost, gradyan artırma makinesi algoritmaları içinde en popüler artırma ağacı algoritmalarından biridir. Problem çözmedeki yüksek performansı ve özellik mühendisliği için minimum gereksinimi nedeniyle endüstride yaygın olarak kullanılmaktadır. Günümüzde GPU kullanımına gereksinim duyabilen derin öğrenme algoritmalarıyla karşılaştırıldığında, XGBoost'un CPU üzerinde çalışan veri kümeleri için kullanımının daha kolay olduğu kabul görmüştür [22]. Modelleme çalışmasından gereksiz parametreleri filtreleyerek doğru matematiksel modeller oluşturmak için öznitelik seçimi çok önemlidir [23]. Bu amaçla çalışma kapsamında İkili Parçacık Sürü Optimizasyonu (İPSO) temelli öznitelik seçimi uygulanmıştır. Algoritmada uygunluk fonksiyonunda belirleyici parametre olarak, RF algoritması kullanılarak elde edilen doğruluk değeri kullanılmıştır. Önerilen modelin performansının değerlendirilmesi amacıyla AdaBoost, Gradient Boosting, SVM, Yapay Sinir Ağları (Artificial Neural Network: ANN), NB, XGBoost yöntemleri ile testler yapılmış ve sonuçlar karşılaştırılmıştır.

II. MATERYAL VE METOT

A. Veri Seti

Bu çalışmada UCI Machine Learning Repository'de yer alan REJAFADA veri seti kullanılmıştır [24]. REJAFADA isimli veri seti 998 kötü amaçlı Jar yazılım örneğinden ve 998 kötü amaçlı olmayan Jar yazılım örneğinden oluşmaktadır. REJAFADA veri setinde kötü amaçlı Jar dosyaları, kötü amaçlı yazılım örnekleri deposu olan VirusShare'den edinilmiştir. Kötü amaçlı olmayan Jar dosyaları ise Java2s.com ve findar.com gibi uygulama havuzlarından edinilmiştir. Daha sonra kötü amaçlı olmayan dosyaların tamamı VirusTotal tarafından taratılarak veri setine dahil edilmiştir. Veri setini oluşturan öznitelikler Jar dosyalarının dinamik analizi sonucunda elde edilmiştir. Dinamik analizler, Windows 7 sistemi üzerine kurulan bir sanal makine vasıtasıyla Cuckoo kum havuzu (Cuckoo Sandbox) tarafından, Jar dosyalarının gerçek zamanlı olarak çalıştırılması yoluyla yapılmış ve toplamda 6284 öznitelik çıkarılmıştır. Bu öznitelikler kötü amaçlı yazılımların davranışlarını ortaya koymayı hedeflemektedir ve aşağıdaki kategorilerde başlıklandırılmıştır :[2]

- Sanal makinelerle ilgili öznitelikler
- Kötü amaçlı yazılımlarla ilgili öznitelikler
- Bitcoin ile ilgili öznitelikler
- Botlarla ilgili öznitelikler
- Tarayıcılarla ilgili öznitelikler
- Güvenlik Duvarı ile ilgili öznitelikler
- Bulut bilişim ile ilgili öznitelikler
- Windows 7 işletim sistemi ve diğer yardımcı programları devre dışı bırakmaya çalışan öznitelikler
- Ağ trafiğiyle ilişkili öznitelikler
- Windows 7 İşletim Sistemi (Regedit) ile ilgili öznitelikler
- Kum havuzu kullanımıyla ilgili öznitelikler
- Fidye Yazılımı ile ilgili öznitelikler
- Exploit ile ilgili öznitelikler
- Bilgi hırsızlarıyla ilgili öznitelikler

B. İkili Parçacık Sürü Optimizasyonu

PSO sürü davranışlarını esas alan bir eniyileme algoritmasıdır. Algoritma sürüdeki bireylerin, zengin kaynaklara ulaşmak için hedefe en yakın bireyi takip ederek konum ve hız bilgilerini bu bireyi baz alarak

güncellemesi prensibine göre çalışmaktadır PSO'da her çözüm bir parçacık olarak temsil edilir ve her parçacığın bir konumu vardır [25]. Bir parçacığın uygunluk değeri, optimize edilecek olan uygunluk fonksiyonu tarafından değerlendirilir. 1997'de Kennedy ve Eberhart, ayrık ikili değişkenler üzerinde çalışmak için PSO algoritmasının ikili versiyonunu geliştirmiştir. İPSO'da her parçacık, ikili değerlerdeki konumunu temsil eder. Bir parçacığın hızı, parçacığın kendi deneyimine ve diğer parçacıkların deneyimine göre belirlenir. Spesifik olarak, her parçacığın hızı önceki hıza, parçacığın yerel en iyisine (pBestPos) ve global en iyisine (gBestPos) göre hesaplanır. Parçacığın yerel en iyi konumu, şimdiye kadar ziyaret ettiği en iyi konumdur, oysa global en iyi konum popülasyondaki herhangi bir parçacığın ziyaret ettiği en iyi konum olarak tanımlanmaktadır. Hız bir kayan nokta vektörü olarak temsil edildiğinden, konumları güncellemek için hız değerlerini olasılık değerlerine eşlemek için bir transfer fonksiyonu gereklidir. Hızlar, 0 veya 1 değerini alan bitin olasılığını temsil ederler. Eşitlik 4'teki sigmoid fonksiyon, hızların tüm gerçek değerlerini [0, 1] aralığında olasılık değerlerine dönüştürmek için kullanılır. Hızları olasılık değerleriyle değiştirdikten sonra, konum vektörleri Eşitlik 5'teki gibi hızlarının olasılığı ile güncellenebilir. [16]

$$v_{ij}^{t+1} = wv_{ij}^{t} + c1 \times rand \times (pBestPos_{ij}^{t} - x_{ij}^{t}) + c2 \times rand \times \left(gBestPos^{t} - x_{ij}^{t}\right) \quad (1)$$

$$pBestPos_{i}^{t+1} = \begin{cases} pBestPos_{i}^{t} & if\ f(x_i^{t+1}) \geq f\left(p\ BestPos_{i}^{t}\right) \\ x_i^{t+1} & else \end{cases} \quad (2)$$

$$gBestPos^{t+1} = \begin{cases} gBestP^{t} & if\ f(x_i^{t+1}) \geq f\left(g\ BestPos^{t}\right) \\ x_i^{t+1} & else \end{cases} \quad (3)$$

$$T\left(v_{ij}^{t}\right) = \frac{1}{1 + e^{-v_{ij}^{t}}} \quad (4)$$

$$x_{ij}^{t+1} = \begin{cases} 1\ if\ rand < T\left(v_{ij}^{t+1}\right) \\ 0\ if\ rand > T\left(v_{ij}^{t+1}\right) \end{cases} \quad (5)$$

Eşitlik 1'de v_i^t parçacığın t yinelemesindeki hızıdır, w bir eylemsizlik ağırlığıdır, c1 ve c2 hızlanma sabitleridir, *rand* 0 ile 1 arasında bir rastgele sayıdır, x_i^t t yinelemesinde i parçacığının mevcut konumudur, f uygunluk fonksiyonudur, $pBestPos_i$, i. parçacığın şimdiye kadar elde ettiği en iyi çözümdür ve $gBestPos_i$, t yinelemesinde o ana kadar elde edilen en iyi çözümü gösterir. Ayrıca, maksimum hız (V_{max}) üzerinde bir sınıra izin verilerek, optimizasyon sürecinin yakınsama oranı kontrol edilebilir. İPSO prosedürü aşağıdaki gibidir:

1. Parçacıkları rastgele konum ve hızlarla başlat.
2. Popülasyondaki her parçacığın uygunluk değerini hesapla
3. i parçacığının uygunluk değeri *pBestPos* değerinden küçükse, *pBestPos* değerini i parçacığının konumuna ayarla (Eşitlik 2)
4. Herhangi bir *pBestPos* güncellenirse ve uygunluk değeri mevcut *gBestPos* değerinden küçükse, *gBestPos*'u i parçacığının mevcut *pBestPos*'u olarak ayarla (Eşitlik 3)
5. En iyi uygunluk değeri veya maksimum iterasyon şartı sağlanmışsa yinelemeyi durdur, aksi takdirde 2. adıma geri dön

C. XGBoost Algoritması

XGBoost, Chen ve Guestrin tarafından geliştirilen bir topluluk ağacı algoritmasıdır [26]. Friedman'ın gradyan artırma algoritmasına dayalı olarak geliştirilmiştir [27,28]. XGBoost, tahmin performansı tek başına kullanılan bireysel tekniklerden daha iyi olan, birleştirilmiş bir model üretmek için karar ağaçlarının verimli bir şekilde uygulanmasından oluşan kollektif bir modeldir. Model karmaşıklığını azaltmak, aşırı öğrenmeyi önlemek ve öğrenme sürecini daha hızlı hale getirmek için amaç fonksiyonunda normalizasyon kullanılmaktadır [29].

XGBoost algoritmasının açıklaması aşağıdaki gibidir:

$D_1 = \{(x_i, y_i)\}$ n örnek ve m özellikten ($|D_1| = n, x_i \in R^m, y_i \in R$) oluşan bir veri kümesi olsun. Kollektif ağaç modeli (Eşitlik 6), çıkışı tahmin etmek için K toplamsal fonksiyonunu kullanır [30].

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(y_i), f_k \in \mathcal{F} \tag{6}$$

Eşitlik 6'da F, regresyon ağaçlarının uzayıdır ve $F = \{f(x) = \omega_{q(x)}\}(q: R^m \to T, \omega \in R^T)$ şeklinde tanımlanır. Burada q ağaçların yapısını, w yaprağın ağırlığını ve T ağaçtaki yaprak sayısını gösterir. Ayrıca f_k terimi, bağımsız bir ağaçla ilgili q ve w'ye karşılık gelen bir fonksiyondur. XGBoost'un amaç fonksiyonu şu şekilde minimize edilir:

$$\begin{aligned} \mathcal{L}(\phi) &= \sum_{l} l(\hat{y}_i, y_i) + \sum_{k} \Omega(f_k) \\ \Omega(f_k) &= \gamma T + \frac{1}{2}\alpha\|\omega\|^2 \end{aligned} \tag{7}$$

Eşitlik 7'de l, ölçülen ve tahmin edilen değerler arasındaki farkı hesaplamak için kullanılan dışbükey bir kayıp fonksiyonudur, $\hat{y}_i$ tahmin edilen değer, y_i ölçülen değerdir, t hataları en aza indirmek için yineleme sayısıdır, Ω aşırı öğrenmeyi önlemek için modelin karmaşıklığını cezalandıran bir düzenleme terimidir [31].

Eşitlik 7'deki ağaç topluluğu modeli, parametre olarak fonksiyonları içerir ve Öklid uzayında geleneksel optimizasyon yöntemleri kullanılarak optimize edilememektedir. Bunun yerine, model eklemeli bir şekilde eğitilir. Biçimsel olarak, $\hat{y}_i^{(t)}$ t. yinelemede i. örneğin tahmini kabul edilirse, Eşitlik 8'deki amaç fonksiyonunu minimize etmek için Chen ve Guestrin f_t'nin eklenmesini önermişlerdir [26].

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{8}$$

$$\mathcal{L}^{(t)} \cong \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i{}^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{9}$$

Eşitlik 9'da $g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ ve $h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ kayıp fonksiyonu üzerine birinci ve ikinci dereceden gradyan istatistikleridir. Adım t'de Eşitlik 10'daki basitleştirilmiş hedefi elde etmek için sabit terimler kaldırılabilir.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{l=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{10}$$

$I_j = \{i \mid q(\mathbf{x}_i) = j\} j$ yaprağının örnek kümesi olarak tanımlandığında; Eşitlik 10'daki Ω, Eşitlik 11'deki gibi genişletilerek yazılabilir.

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T \end{aligned} \tag{11}$$

Sabit bir q(x) yapısı için, j yaprağının optimal w_i^* ağırlığı Eşitlik 12'de gösterildiği şekilde hesaplanabilir.

$$w_i^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_l + \lambda} \tag{12}$$

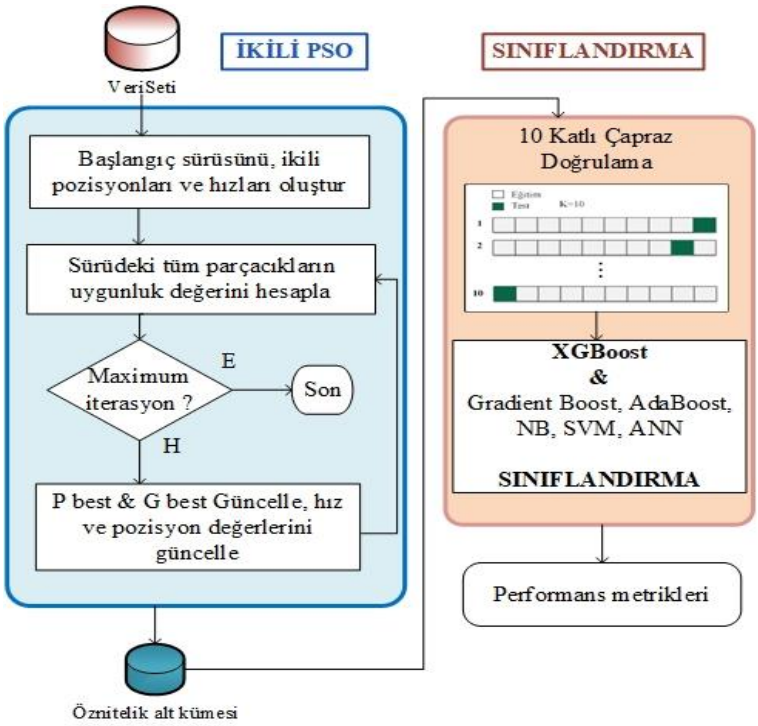
Optimal değer ise Eşitlik 13'e göre hesaplanabilir.

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{13}$$

Eşitlik 13, bir ağaç yapısının q kalitesini ölçmek için bir puanlama fonksiyonu olarak kullanılabilir. Bu puan, daha geniş bir amaç fonksiyonları yelpazesi için türetilmesi dışında, karar ağaçlarını değerlendirmek için kullanılan safsızlık puanına benzer. Normalde olası tüm q ağaç yapılarını saymak imkansızdır. Bunun yerine, tek bir yapraktan başlayan ve yinelemeli olarak ağaca dallar ekleyen açgözlü bir algoritma kullanılır. I_L ve I_R'nin bölünmeden sonra sol ve sağ düğümlerin örnek kümeleri olduğunu varsayıldığında, $I = I_L \cup I_R$ iken, bölmeden sonraki kayıp azalması Eşitlik 14'teki gibidir. Bu formül genellikle uygulamada bölünmüş adayları değerlendirmek için kullanılır.

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{14}$$

III. UYGULAMA



Şekil 1. Önerilen Jar kötü amaçlı yazılım tespit modeli

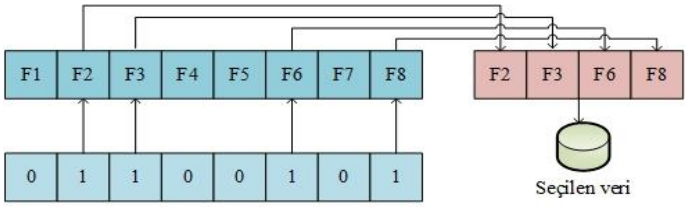
Şekil 1'de görselleştirilen bu çalışmada, modelleme kalitesini iyileştirmek amacıyla öznitelik seçimi kullanılmıştır. Çalışmadaki problem bir sınıflandırma problemidir. Hedefler, model performansının maksimize edilmesi ve kullanılan özniteliklerin sayısının minimize edilmesidir. İPSO algoritması, problemleri en az bilgi ile çözmek için kullanılan metasezgisel algoritmalardan biri olarak kabul edilmektedir [32,33]. Parçacıkların uygunluk değerlerinin hesaplanmasında Eşitlik 15'teki fonksiyon kullanılmıştır [34]. Önerilen yöntemde hedeflenen, uygunluk fonksiyonunun minimizasyonudur. Burada P sınıflandırıcının performansı, N_f öznitelik alt kümesi yani seçilen özniteliklerin sayısı, Nt veri setindeki toplam öznitelik sayısıdır. α, [0, 1] arasında değer alan, sınıflandırma performansının etkisini kontrol eden ve toplam öznitelik sayısına göre öznitelik alt kümesinin boyutu arasındaki oransal değere karar veren parametredir [34]. Sınıflandırma performansı bu çerçevede en önemli ölçü olduğundan α parametresi 0.99 olarak belirlenmiştir [35,36].

Uygunluk fonksiyonunda sınıflandırıcı olarak, geleneksel olarak kullanılan SVM, En Yakın Komşu (K-Nearest Neighbours) algoritmaları yerine RF algoritması kullanılmıştır [16,32,34]. Karar ağacı modelleri algoritmadaki sadeleştirmelerinden ve farklı verilere ait öznitelik türlerini işlemede ki esnekliklerinden dolayı oldukça popülerdirler. Bununla birlikte, tek ağaç (single-tree) modeli belirli eğitim verilerine duyarlıdır ve aşırı

öğrenme problemi kolay oluşmaktadır. Topluluk yöntemleri, bireysel ağaçları birleştirerek bu sorunları çözebilmekte ve tek sınıflandırıcılardan daha doğru sonuçlar üretebilmektedir. Topluluk yöntemlerinden biri olan rastgele orman algoritması, her ağacın rastgele bağımsız bir veri kümesine bağlı olduğu ve ormandaki tüm ağaçların aynı dağılıma sahip olduğu çoklu ağaç tahmincilerinin birleşimi olarak tanımlanabilmektedir. Rastgele ormanın kapasitesi, yalnızca tek tek ağacın gücüne değil, aynı zamanda farklı ağaçlar arasındaki korelasyona da bağlıdır. Tek ağacın gücü ne kadar güçlüyse ve farklı ağaçların korelasyonu ne kadar az olursa, rastgele ormanın performansı o kadar iyi olmaktadır [37]. Eşitlik 15'te kullanılan P parametresi sınıflandırıcı performansı; doğruluk, F-puanı, kesinlik (precision) gibi performans değerlendirme metriklerinden biri olabilmektedir [38]. Rastgele orman algoritmasının performans değerlendirilmesinde P parametresi için doğruluk (accuracy) değeri kullanılmıştır.

$$f(x) = \alpha(1-P) + (1-\alpha)\left(1 - \frac{N_f}{Nt}\right) \tag{15}$$

Önerilen algoritmadaki x_i^tparçacığı, t yinelemesinde sürüde i aday çözümünü temsil eder. Sürüdeki i. Parçacık d-boyutlu bir vektörle temsil edilir ve $x_i^t = [x_{i1}^t, x_{i2}^t .. x_{id}^t]$ şeklinde tanımlanır. Burada, x_{id}^t, t yinelemesi için i. parçacığın d. boyuta göre konumudur. Öznitelik seçme problemlerinde arama uzayı d-boyutlu durum uzayıdır ve ikilidir. Burada her boyutun ikili değeri, karşılık gelen özelliğin kararını belirtir. Bir öznitelik seçilecekse, karşılık gelen bit değeri bir olarak ayarlanır, öznitelik istenmiyorsa karşılık gelen bit değeri sıfır olarak ayarlanır. Örnek olarak Şekil 2'deki gibi 8 özniteliğin olduğu varsayılsın ve her bir parçacık ikili olarak [0 1 1 0 0 1 0 1] temsil edilsin. Bu parçacıkta birinci bit 0 olduğu için F1 özniteliğinin seçilmeyeceği, ikinci bit 1 olduğu için F2 özniteliğinin seçileceği anlamına gelmektedir. Sonuçta 1 olan bitler F2, F3, F6, F8 öznitelikleri seçilecektir.



Şekil 2. İkili öznitelik seçimi mekanizması

Bu çalışmada öznitelik seçim yöntemi olarak kullanılan İPSO algoritmasının parametreleri; parçacık sayısı (N), maksimum yineleme sayısı (T), bilişsel faktör (C1), sosyal faktör (C2), eylemsizlik ağırlığı (W), mesafe türü (P: {1: Minkowski p-norm, 2: Öklid mesafesi}), dikkate alınacak komşu sayısı (K)'dir [38]. N, T, C1, C2, W, P, K parametrelerinin değerleri sırasıyla 10, 50, 2, 2, 0.6, 2, 9 olarak seçilmiş ve Python PySwarms kütüphanesi kullanılarak algoritma koşturulmuştur [38]. XGBoost sınıflandırıcı için Python xgboost kütüphanesi ve "default" parametreler kullanılmıştır. AdaBoost, Gradient Boosting, NB, SVM algoritmaları için Python sklearn kütüphanesi kullanılmıştır. ANN içinse Python Keras kütüphanesi kullanılmıştır. Kurulan ANN modeli 1 giriş katmanı, 8 düğümden oluşan 1 gizli katman ve çıkış katmanından oluşmaktadır. Aktivasyon fonksiyonu olarak Rectifier Lienar Units (RELU) kullanılmıştır. Çıkış katmanında kullanılan aktivasyon fonksiyonu ise sigmoid'dir. Loss parametresi binary_crossentropy, optimizer parametresi adam, batch_size 32, epoch parametresi ise 100 olarak ayarlanmıştır. Diğer dosya yükleme, veri çerçevesinin ayarlanması vb. işlemlerde Python numpy, pandas, statistics kütüphaneleri kullanılmıştır.

Kurulan modelin eğitim ve test başarımının değerlendirilebilmesi için kullanılan; doğru olarak sınıflandırılan örneklerin oranı olan doğruluk değeri Eşitlik 16'da, pozitif olarak tahmin edilen örneklerin gerçekte ne kadarının pozitif olduğunu ifade eden kesinlik değeri Eşitlik 17'de, gerçek pozitif değerlerin ne kadarının doğru olduğunu ifade eden duyarlılık (recall) değeri Eşitlik 18'de, duyarlılık ve kesinlik değerlerinin harmonik ortalaması olan F1-Skoru (F1-Score) Eşitlik 19'da gösterilmiştir.

$$Doğruluk = \frac{TP + TN}{TP + FN + TN + FP} \tag{16}$$

$$Kesinlik = \frac{TP}{TP + FP} \tag{17}$$

$$Duyarlılık = \frac{TP}{TP + FN} \tag{18}$$

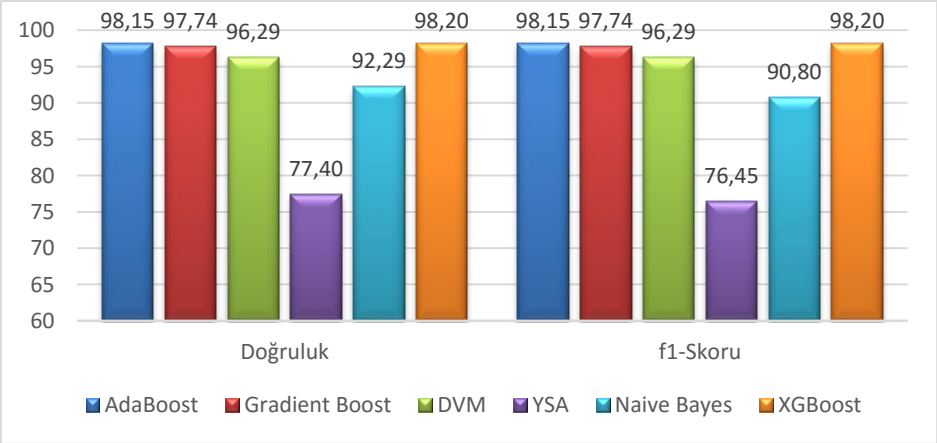
$$F1 - Skor = \frac{2xDuyarlılıkxKesinlik}{Duyarlılık + Kesinlik} \quad (19)$$

IV. DENEYSEL SONUÇLAR

Önerilen modelin eğitilmesi ve test edilmesi işlemleri için Google Colaboratory ya da kısaca Colab bulut sistemi kullanılmıştır. Bu sistem içerisinde birçok Python kütüphanesini hazır olarak bulundurmakta ayrıca ücretsiz olarak hizmet vermektedir [39]. Önerilen modelin performans metrikleri elde edilirken; tüm algoritmaların koşumunda 10 kat çapraz doğrulama kullanılmıştır. Kullanılan veri setinde 6284 öznitelik bulunmaktadır. Veri seti ilk önce öznitelik seçme işlemi yapılmadan XGBoost algoritması ve diğer ML yöntemleri ile eğitilip test edilmiştir. Elde edilen performans metrikleri Tablo 1'de gösterilmiştir. AdaBoost, Gradient Boosting, SVM, ANN, NB, XGBoost ile elde edilen eğitim doğrulukları sırasıyla 0.9952, 1.00, 1.00, 0.9525, 0.7893, 0.9954 olarak elde edilmiş, test doğruluk değerleri ise sırasıyla 0.9815, 0.9774, 0.9629, 0.9229, 0.774, 0.9820 olarak elde edilmiştir. Ayrıca eğitilen veri setinin test sonuçları olan doğruluk ve F1-Skoru Şekil 3'te görselleştirilmiştir.

Tablo 1. Orijinal veri seti ile performans metrikleri

Sınıflandırıcı	Eğitim				
	Eğitim Süresi (sec)	Doğruluk (Accuracy)	Kesinlik (Precision)	F1- Skoru (F1-Score)	Duyarlılık (Recall)
AdaBoost	93.06	0.9952	0.9952	0.9952	0.9952
Gradient Boosting	221.79	1.0000	1.0000	1.0000	1.0000
SVM	83.74	1.0000	1.0000	1.0000	1.0000
ANN	402.14	0.9525	0.9756	0.9388	0.9526
Naive Bayes	2.60	0.7893	0.8345	0.7818	0.7893
XGBoost	226.70	0.9954	0.9955	0.9954	0.9954
	Test				
	Test Süresi (sec)	Doğruluk (Accuracy)	Kesinlik (Precision)	F1-Skoru (F1-Score)	Duyarlılık (Recall)
AdaBoost	1.59	0.9815	0.9816	0.9815	0.9814
Gradient Boosting	0.12	0.9774	0.9781	0.9774	0.9774
SVM	6.13	0.9629	0.9634	0.9629	0.9629
ANN	1.15	0.9229	0.9470	0.9080	0.9229
Naive Bayes	0.19	0.7740	0.8237	0.7645	0.7741
XGBoost	0.20	0.9820	0.9824	0.9820	0.9820

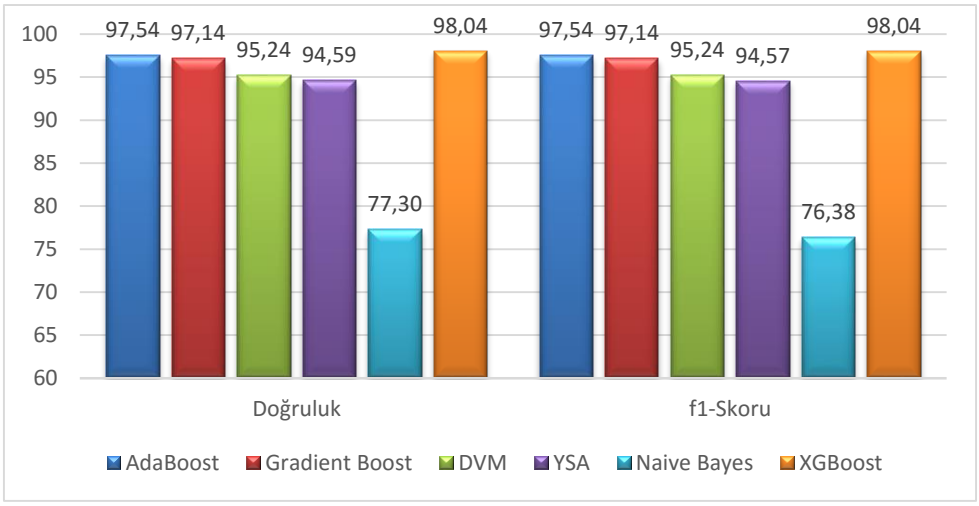


Şekil 3. Orijinal veri seti ile doğruluk ve F1-Skor değerleri

Veri seti daha sonra önerilen RF uygunluk fonksiyonlu İPSO algoritması ile yapılan öznitelik seçme işlemine tabi tutulmuş ve öznitelik sayısı 3378 olarak elde edilmiştir. Elde edilen öznitelikler XGBoost algoritması ve diğer ML yöntemleri ile eğitilip test edilmiştir. Elde edilen performans metrikleri Tablo 2'de gösterilmiştir. AdaBoost, Gradient Boosting, SVM, ANN, NB, XGBoost ile elde edilen eğitim doğrulukları sırasıyla 0.9931, 0.9932, 0.9997, 0.9925, 0.7840, 0.9925 olarak elde edilmiş, test doğruluk değerleri ise sırasıyla 0.9754, 0.9714, 0.9524, 0.9459, 0.773, 0.9804 olarak elde edilmiştir. Ayrıca eğitilen veri setinin test sonuçları olan doğruluk ve F1-Skoru Şekil 4'te görselleştirilmiştir.

Tablo 2. Öznitelik seçme işlemi ile performans metrikleri

Sınıflandırıcı	**Eğitim**				
	Eğitim Süresi (sec)	**Doğruluk (Accuracy)**	**Kesinlik (Precision)**	**F1-Skoru (F1-Score)**	**Duyarlılık (Recall)**
AdaBoost	47.52	0.9931	0.9932	0.9931	0.9932
Gradient Boosting	50.21	0.9932	0.9932	0.9932	0.9932
SVM	34.97	0.9997	0.9997	0.9997	0.9997
ANN	188.11	0.9925	0.9927	0.9925	0.9925
Naive Bayes	1.45	0.7840	0.8294	0.7763	0.7841
XGBoost	128.54	0.9925	0.9926	0.9925	0.9925
	Test				
	Test Süresi (sec)	**Doğruluk (Accuracy)**	**Kesinlik (Precision)**	**F1-Skoru (F1-Score)**	**Duyarlılık (Recall)**
AdaBoost	0.96	0.9754	0.9756	0.9754	0.9754
Gradient Boosting	0.08	0.9714	0.9718	0.9714	0.9714
SVM	0.63	0.9524	0.9528	0.9524	0.9524
ANN	0.93	0.9459	0.9491	0.9457	0.9459
Naive Bayes	0.11	0.7730	0.8211	0.7638	0.7731
XGBoost	0.13	0.9804	0.9808	0.9804	0.9805



Şekil 4.Öznitelik seçme işlemi ile doğruluk ve F1-Skor değerleri

Tablo 3'te literatürde kullanılmış aynı veri seti ve farklı veri setleri ile gerçekleştirilen çalışmalar ile bu çalışmalarda kullanılan yöntemler gösterilmiş ve bu çalışmalarla önerilen çalışmanın başarım oranlarının karşılaştırması verilmiştir.

Tablo 3. Farklı çalışmalardaki Jar kötü amaçlı yazılım tespit başarı oranının karşılaştırması

Çalışmalar	Yöntem	Veri Seti	Doğruluk
Balan ve Popescu [1]	Hiyerarşik kümeleme	Bitdefender Cyber Threat Intelligence Laboratory	97.54
Jha vd. [40]	LSTM, GRU	Contagio malware dump	93.33
R. Pinheiro vd. [13]	En İyi Eşlenik Algoritması	REJAFADA	95.61
R. Pinheiro vd. [2]	Aşırı Öğrenme Makineleri	REJAFADA	91.58
Obaidat vd. [3]	Image Tabanlı ESA	Jarhead, VirusTotal, VirusShare, Archive.org VXStream	98.4
Önerilen Çalışma	İPSO tabanlı XGBoost	REJAFADA	98.04

V. SONUÇLAR

Her yıl binlerce kötü amaçlı yazılım geliştirilmektedir ve sürekli olarak artmaktadır. Bu nedenle, kötü amaçlı yazılım tespit mekanizmaları hayati önem taşımaktadır. Kötü amaçlı yazılım tanımlamasında zafiyet meydana geldiği durumlarda, hassas verilerin yetkisiz kişilerce kullanıma sunulması ihtimali vardır [15]. Java kötü amaçlı yazılımları, Java'nın yaygın kullanımı nedeniyle önemli bir güvenlik tehdit kaynağı olmaya devam etmektedir. Bu tehditlerin önlenmesine katkı sunmak adına bu çalışmada İPSO öznitelik seçme tabanlı XGBoost sınıflandırıcı bir model önerilmiştir. Kötü amaçlı jar uzantılı dosyaların davranışsal özniteliklerinin çıkarıldığı REJAFADA adlı veri setinin kullanıldığı çalışmada RF sınıflandırma algoritmasının kullanıldığı uygunluk fonksiyonu ile hem doğruluk hem hız anlamında fonksiyonun minimizasyonuna katkı sağlanmıştır. İPSO algoritması ile seçilen özniteliklere sayısal olarak bakıldığında 3378 öznitelik seçilmiş ve toplam öznitelik sayısına göre %46.24 oranında öznitelik sayısı azaltılmıştır. Yine eğitim ve test sürelerine bakıldığında orijinal veri seti ile önerilen çalışmada elde edilen sonuçlara göre, eğitim için geçen süre %43.4 oranında azaltılmış, test için geçen süre %35 oranında azaltılmıştır.

Seçilen veri seti XGBoost algoritması ile eğitilip teste tabi tutulduğunda elde edilen test sonuçlarına göre; Doğruluk, Kesinlik, F1-Skoru, Duyarlılık değerlerinin sırasıyla 0.9804, 0.9808, 0.9804, 0.9805 olduğu ve diğer algoritmalara göre daha yüksek performans gösterdiği sonucuna varılmıştır. Aynı zamanda XGBoost algoritmasının performansının karşılaştırmasını yapabilmek amacıyla ile literatürde görülen diğer Boosting algoritmalarından olan AdaBoost, Gradient Boosting ve ML tekniklerinden olan SVM, NB, ANN ile seçilen öznitelikleri içeren veri seti eğitilip teste tabi tutulmuştur. Performans metriklerine bakıldığında Boosting algoritmalarının birbirine yakın sonuçlar aldığı, SVM, ANN yöntemlerinin eğitim doğruluklarının yüksek olmasına rağmen test doğruluk oranlarının Boosting algoritmalarına göre nispi olarak geride kaldığı görülmüştür. NB algoritmasında ise eğitim ve test doğruluklarında diğer algoritmalara göre başarı oranının düşük kaldığı gözlemlenmiştir.

Tablo 3'e bakıldığında; REJAFADA veri seti üzerinde gerçekleştirilen çalışmalara göre başarım oranının önerilen yöntem ile geliştirildiği gözlemlenmiştir [3, 15]. Obaidat vd'nin yaptığı image tabanlı ESA mimarisinde elde edilen sonuçlarda ise her ne kadar veri seti farklı olsa da yakın sonuçlar elde edilmiştir [5].

Sonuç olarak, kötü amaçlı yazılımların tespiti kullanıcılar ve kurumlar açısından çok önemlidir. Oransal bakıldığında çalışmaların yüksek başarıma ulaşması elbette önemlidir. Ancak tespit edilemeyen kötü amaçlı yazılımın verebileceği zararlar göz ardı edilmemelidir. Bu bağlamda bu çalışmada; İPSO öznitelik seçimli XGBoost algoritması ile bir model kurularak tespit mekanizması oluşturulmuştur. %98.04 oranında doğruluk değeri elde edilmiştir. Ancak uygulanacak yeni öznitelik çıkarma yaklaşımları ve kurulacak yeni modellerle başarım oranının yükseltilmesi faydalı olacaktır.

KAYNAKLAR

[1] Balan, G., & Popescu, A. S. (2018). Detecting Java Compiled Malware using Machine Learning Techniques. *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. 20-23 September, Timisoara, Romania, 435-439.

[2] Pinheiro, R. P., Lima, S. M., Souza, D. M., Silva, S. H., Lopes, P. G., de Lima, R. D., de Oliveira, J. R., Monteiro, T. de A., Fernandes, S. M., & Albuquerque, E. de Q. (2022). Antivirus applied to JAR malware detection based on runtime behaviors. *Scientific Reports, 12*(1). 1-17.

[3] Obaidat, I., Sridhar, M., Pham, K. M., & Phung, P. H. (2022). Jadeite: A novel image-behavior-based approach for Java malware detection using deep learning. *Computers & Security, 113*. 102547.

[4] Kumar, R., & Vaishakh, A. R. E. (2016). Detection of obfuscation in java malware. *Procedia Computer Science*, *78*. 521-529.
[5] Krebs on Security. (2020). *Krebs on Security* https://krebsonsecurity.com/2020/03/live-coronavirus-map-used-to-spread-malware/, (16.05.2022).
[6] Ye, Y., Li, T., Adjeroh, D., & Iyengar, S. S. (2017). A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, *50*(3). 1-40.
[7] Özgür, A., & Erdem, H. (2018). Feature selection and multiple classifier fusion using genetic algorithms in intrusion detection systems. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *33*(1). 75-87.
[8] Anıl, U. (2022). Using network traffic analysis deep learning based Android malware detection. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *37*(4). 1823-1838.
[9] Bhilvare, A., & Manik, T. (2015). An Overview of Different Malware Analysis Techniques in Android. *IJSRD-International Journal for Scientific Research & Development*, *3*(1). 368-372.
[10] Yerima, S. Y., Sezer, S., & McWilliams, G. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security*, *8*(1). 25-36.
[11] Kulkarni, K. (2018). *Android Malware Detection through Permission and App Component Analysis using Machine Learning Algorithms.* Master's thesis, University of Toledo. Toledo.
[12] Tong, F., & Yan, Z. (2017). A hybrid approach of mobile malware detection in Android. *Journal of Parallel and Distributed computing*, *103*. 22-31.
[13] Pinheiro, R., Lima, S., Fernandes, S., Albuquerque, E., Medeiros, S., Souza, D., Monteiro, T., Lopes, P., Lima, R., & Oliveira, J. (2019). Next generation antivirus applied to Jar malware detection based on runtime behaviors using neural networks. *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 06-08 May, Porto, Portugal, 28-32.
[14] Herrera, A., & Cheney, B. (2015). JMD: a hybrid approach for detecting Java malware. *Proceedings of the 13th Australasian Information Security Conference (AISC 2015)*, *27*. Sydney, Australia, 30.
[15] Jha, P. K., Shankar, P., Sujadevi, V. G., & Prabhaharan, P. (2018). Deepmal4j: Java malware detection employing deep learning. *International Symposium on Security in Computing and Communication*. 389-402.
[16] Gunasundari, S., Janakiraman, S., & Meenambal, S. (2018). Multiswarm heterogeneous binary PSO using win-win approach for improved feature selection in liver and kidney disease diagnosis. *Computerized Medical Imaging and Graphics*, *70*. 135-154.
[17] Brezočnik, L., Fister, I., & Podgorelec, V. (2018). Swarm intelligence algorithms for feature selection: a review. *Applied Sciences*, *8*(9). 1521.
[18] Ji, B., Lu, X., Sun, G., Zhang, W., Li, J., & Xiao, Y. (2020). Bio-inspired feature selection: An improved binary particle swarm optimization approach. *IEEE Access*, *8*. 85989-86002.
[19] Abbasi, M. S., Al-Sahaf, H., Mansoori, M., & Welch, I. (2022). Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection. *Applied Soft Computing*, *121*. 108744.
[20] Ali, Z., & Soomro, T. R. (2018). An efficient mining based approach using PSO selection technique for analysis and detection of obfuscated malware. *Journal of Information Assurance & Cyber security*, *2018*. 1-13.
[21] Dong, D., Ye, Z., Su, J., Xie, S., Cao, Y., & Kochan, R. (2020). A malware detection method based on improved fireworks algorithm and support vector machine. *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*. 846-851.
[22] Song, K., Yan, F., Ding, T., Gao, L., & Lu, S. (2020). A steel property optimization model based on the XGBoost algorithm and improved PSO. *Computational Materials Science*, *174*. 109472.
[23] Mo, H., Sun, H., Liu, J., & Wei, S. (2019). Developing window behavior models for residential buildings using XGBoost algorithm. *Energy and Buildings*, *205*. 109564.
[24] Dua, Dheeru, & Graff, Casey. (2017). UCI Machine Learning Repository. *University of California, Irvine, School of Information and Computer Sciences* https://archive.ics.uci.edu/ml/datasets/cardiotocography, (08.03.2022).
[25] Cimen, M. E., & Boz, A. F. (2019). Parameter identification of a non-minimum phase second order system with time delay using relay test and PSO, CS, FA algorithms. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *34*(1). 461-477. https://doi.org/10.17341/gazimmfd.416507
[26] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. San Francisco, California, USA, 785-794.
[27] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, *29*(5). 1189-1232.

[28] Zhou, J., Qiu, Y., Khandelwal, M., Zhu, S., & Zhang, X. (2021). Developing a hybrid model of Jaya algorithm-based extreme gradient boosting machine to estimate blast-induced ground vibrations. *International Journal of Rock Mechanics and Mining Sciences*, *145*. 104856. https://doi.org/10.1016/j.ijrmms.2021.104856
[29] Jabeur, S. B., Mefteh-Wali, S., & Viviani, J.-L. (2021). Forecasting gold price with the XGBoost algorithm and SHAP interaction values. *Annals of Operations Research*. 1-21.
[30] Chen, Y., Guo, A., Chen, Q., Quan, B., Liu, G., Li, L., Hong, J., Wei, H., & Hao, Z. (2021). Intelligent classification of antepartum cardiotocography model based on deep forest. *Biomedical Signal Processing and Control*, *67*. 102555. https://doi.org/10.1016/j.bspc.2021.102555
[31] Wang, W., Shi, Y., Lyu, G., & Deng, W. (2017). Electricity consumption prediction using xgboost based on discrete wavelet transform. *DEStech Trans. Comput. Sci. Eng*. 716-729.
[32] Akbari, H., Sadiq, M. T., Payan, M., Esmaili, S. S., Baghri, H., & Bagheri, H. (2021). Depression Detection Based on Geometrical Features Extracted from SODP Shape of EEG Signals and Binary PSO. *Traitement du Signal*, *38*(1)
[33] Too, J., Abdullah, A. R., Mohd Saad, N., & Tee, W. (2019). EMG feature selection and classification using a Pbest-guide binary particle swarm optimization. *Computation*, *7*(1). 12.
[34] Vieira, S. M., Mendonça, L. F., Farinha, G. J., & Sousa, J. M. (2013). Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Applied Soft Computing*, *13*(8). 3494-3504.
[35] Faris, H., Mafarja, M. M., Heidari, A. A., Aljarah, I., Al-Zoubi, A. M., Mirjalili, S., & Fujita, H. (2018). An efficient binary Salp Swarm Algorithm with crossover scheme for feature selection problems. *Knowledge-Based Systems*, *154*. 43-67. https://doi.org/10.1016/j.knosys.2018.05.009
[36] Too, J., Abdullah, A. R., & Mohd Saad, N. (2019). Binary competitive swarm optimizer approaches for feature selection. *Computation*, *7*(2). 31.
[37] Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018). Random forest for credit card fraud detection. *2018 IEEE 15th international conference on networking, sensing and control (ICNSC)*. Zhuhai, China, 1-6.
[38] Miranda, L. J. (2018). PySwarms: a research toolkit for Particle Swarm Optimization in Python. *Journal of Open Source Software*, *3*(21). 433. https://doi.org/10.21105/joss.00433
[39] Google Colaboratory. (2022). *Colaboratory*, https://colab.research. google.com/, (10.08.2022).
[40] Jha, P. K., Shankar, P., Sujadevi, V. G., & Prabhaharan, P. (2018). Deepmal4j: Java malware detection employing deep learning. *International Symposium on Security in Computing and Communication*. 19-22 September, Bangalore, India, 389-402.