T.C.

YALOVA ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

FİDYE YAZILIMLARININ MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TESPİT EDİLMESİ

YÜKSEK LİSANS TEZİ

Volkan OKUR

195113003

Adli Bilişim Anabilim Dalı

Adli Bilişim (Tezli) Programı

Tez Danışmanı: Prof. Dr. Murat GÖK

HAZİRAN 2021

T.C.

YALOVA ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

FİDYE YAZILIMLARININ MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TESPİT EDİLMESİ

YÜKSEK LİSANS TEZİ

Volkan OKUR

195113003

Adli Bilişim Anabilim Dalı

Adli Bilişim (Tezli) Programı

Tez Danışmanı: Prof. Dr. Murat GÖK

HAZİRAN 2021

ÖNSÖZ

Değerli bilgi ve deneyimlerinden yararlandığım tez danışmanım Prof. Dr. Murat GÖK'e teşekkür ederim. Maddi ve manevi desteklerini hiçbir zaman esirgemeyen eşim Melike GÜLER OKUR ve aileme şükranlarımı sunarım.

Haziran 2021

Volkan OKUR

Elektronik & Bilgisayar

Teknolojileri Öğretmeni

İÇİNDEKİLER Sayfa No

KISALTMALAR

ML :Makine Öğrenmesi
API :Uygulama Programlama Arayüzü
N/A : Uygulanabilir değil
k-EYK: K-En Yakın Komşuluğu
ÇKAA: Çok Katmanlı Algılayıcı Ağ
DVM: Destek Vektör Makineleri

ÇİZELGE LİSTESİ Sayfa No

ŞEKİL LİSTESİ Sayfa No

FİDYE YAZILIMLARININ MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TESPİT EDİLMESİ

ÖZET

Fidye yazılımları, bilgisayar dosyalarına şifreleme yaparak kullanıcının erişimini engelleyen kötü amaçlı yazılımlardır. Bulaştığı sistemlerin açılabilmesi için kişilerden maddi bir bedel (fidye) talep eder. Bu talep gerçekleşmezse kurbanın şifrelenmiş dosyalara erişimi mümkün değildir. Kimi zaman bilgisayar biçimlendirilse dahi zararlı yazılım aktif olarak kalabilmektedir. Kurban, dosyalarına erişebilmek için mecburen talebi yerine getirerek fidyeyi ödemek zorunda kalır. Fidye ödemesi, finansal takibinin zor olmasından dolayı çoğunlukla kripto paralar üzerinden istenmektedir. İstenen fidyenin miktarı verilen zarara göre değişmektedir. Zarar büyüdükçe, istenen fidye o oranda artmaktadır. Bu tez çalışmasının amacı, makine öğrenmesi yöntemleri ile fidye yazılımlarını tespit etmektedir. Kullandığımız fidye yazılımı veri seti, 1.524 örnek ve 30.967 öznitelikten oluşmaktadır. Bu veri seti üzerinde çoklu sınıflandırma çalışmaları yapılabilmektedir. Bu çalışmada, ilk olarak, veri setimiz üzerinde Naif Bayes, Bayes Ağı, k-En Yakın Komşuluğu (k-EYK), Rastgele Orman, Çok Katmanlı Algılayıcı Ağ (ÇKAA), Destek Vektör Makineleri (DVM) ve Torbalama yöntemleri ile fidye yazılımı sınıflandırma çalışmaları gerçekleştirilmiştir. İkinci aşamada, fidye yazılımı tahmini başarımını artırabilmek adına, öznitelik seçim yöntemleri (Doğrusal İleri Seçim Yöntemi, Genetik Algoritma ve Parçacık Sürü Eniyileme) ile belirtilen sınıflandırma algoritmaları test edilmiştir. Üçüncü aşamada, C-XOR adını verdiğimiz yeni bir öznitelik seçimi yöntemi geliştirilmiştir. Çoklu sınıflandırmada C-XOR yöntemi ile birlikte Rastgele Orman Algoritması, % 59,15 doğruluk değeri ile en iyi sonucu vermiştir.

Anahtar Kelimeler: Siber güvenlik, Zararlı yazılımlar, Fidye yazılımı, Öznitelik seçimi, Çoklu sınıflandırma, C-XOR algoritması.

DETECTION OF RANSOMWARE USING MACHINE LEARNING ALGORITHMS

ABSTRACT

Ransomware is malware that encrypts computer files, preventing user access. It demands a financial fee (ransom) from people in order to open the systems it has infected. If this request fails, the victim will not be able to access encrypted files. Sometimes, even if the computer is formatted, malicious software can remain active. The victim must fulfill the request and pay the ransom in order to gain access to his files. The ransom payment is mostly requested over cryptocurrencies due to the difficulty of financial tracking. The amount of ransom required varies according to the damage inflicted. The bigger the damage, the higher the ransom demanded. The aim of this thesis is to identify ransomware with machine learning methods. The ransomware data set we use consists of 1,524 samples and 30,967 features. Multiple classification studies can be performed on this data set. In this study, firstly, ransomware classification on our dataset using Naive Bayes, Bayesian Network, k-Nearest Neighborhood (k-EYK), Random Forest, Multi-Layer Sensor Network (MCCA), Support Vector Machines (SVM) and Bagging methods. works have been carried out. In the second stage, the classification algorithms specified by the feature selection methods (Linear Forward Selection Method, Genetic Algorithm and Particle Swarm Optimization) were tested in order to increase the ransomware prediction performance. In the third stage, a new feature selection method called C-XOR has been developed. Random Forest Algorithm together with C-XOR method in multi classification gave the best result with an accuracy value of % 59.15.

Keywords: Cyber security, Malware, Ransomware, Feature selection, Multiclassification, C-XOR algorithm.

1. GİRİŞ

Fidye yazılım (Ransomware), kurbanlarından fidye talep eden zararlı bir yazılımdır (malware). İşletim sistemindeki açıklardan yararlanarak, kimi zaman e-posta yoluyla, kimi zaman zararlı web sitesi içerikleriyle bulaşıp aktif hale geçme şansını bulurlar. Bu kötü yazılımlar bilgisayar kullanıcılarının dosyalarını şifreleyerek büyük maddi zararlara sebep olurlar. Bu sorun, her ne kadar anti-virüs programlarıyla engellenmeye çalışılsa da anti-virüs yazılımın güncel olmaması, zararlı yazılımı tanımaması vb. kısıtlardan dolayı bulaş riski tam olarak ortadan kaldırılamamaktadır. Ayrıca, anti-virüslerin güncellense dahi ilk saldırıların zararını düzeltmeleri mümkün değildir.

Bu konu toplum içinde bazen yetkili devlet kurumları, bazen de çeşitli firmalar tarafından eğitim ile kullanıcıların bilinçlendirilmesiyle çözülmeye çalışılmıştır. Tüm dünyada böyle bir zararın önüne geçilmesi ciddi maddi desteklerle olabilecekken, her bilgisayar kullanıcısı kendi önlemini kendisi almak durumunda kalacaktır. Devletler basit bir çözüm olarak halkını ulusal kanallarından ve kısa mesajlar ile uyarılarda bulunarak bilinçlendirmeye çalışmaktadır. Kullanıcıların bilmedikleri bir web sitesine girmemeleri, bilinmeyen e-postaları açmamaları konusunda uyarıldığını, web sitelerinden görebiliyoruz. Örneğin, Avusturalya hükümeti[1] vb.

Fidye yazılımlarından korunabilmek için çeşitli antivirüs yazılımları, Windows işletim sistemi güncellemeleri, güvenlik duvarı vb. gibi yöntemler kullanılmaktadır. Araştırmacılar bazıları statik, bazıları da dinamik araştırma ile fidye yazılımın davranışlarını anlamaya ve buna göre çözüm üretmeye çalışmaktadırlar. Son yıllarda makine öğrenmesi yöntemleri ile zararlı yazılımların tespit edilmesine çalışılmıştır. Makine öğrenmesindeki algoritmalar ile bir yazılımın kötü niyetli olduğunu yüksek oranda anlamak mümkün olmaktadır. Aynı şekilde fidye yazılımlarındaki gelişmeler yine sistem açıklarını kullanıp, sisteme sızarak dosya kullanımını engellemeyi başardığını görmekteyiz. Sonuç olarak; kuracağımız yazılımların iyi veya kötü yazılım olduğunu tanımlayabilirsek, onun makinede oluşturabileceği zararları da minimize etmiş oluruz. Bu bağlamda, makine öğrenmesi ile kurulmak istenen yazılım incelenerek yazılım hakkında bir sonuç çıkartabiliriz. Verilecek bu kararda doğru tespit oranını artırabilmek için yeni metotlar ve yaklaşımlar geliştirmek mecbur hale gelmiştir.

[1] https://www.cyber.gov.au/ransomware, 01.04.2021

1.1 Tezin Amacı

Bu tez çalışmasının amacı, zararlı yazılımlardan biri olan fidye yazılımlarını makine öğrenmesi yöntemleri ile tespit etmektedir. Kullandığımız veri seti üzerinde çoklu sınıflandırma çalışmaları yapılabilmektedir. Bu çalışmada, ilk olarak, veri setimiz üzerinde Naif Bayes, Bayes Ağı, k-En Yakın Komşuğu (k-EYK), Rastgele Orman, Çok Katmanlı Algılayıcı Ağ (ÇKAA), Destek Vektör Makineleri (DVM) yöntemleri ile fidye yazılımı sınıflandırma çalışmaları gerçekleştirdik. İkinci aşamada, fidye yazılımı tahmini başarımını artırabilmek adına, öznitelik seçim yöntemleri (Doğrusal İleri Seçim Yöntemi, Genetik Algoritma ve Parçacık Sürü Eniyileme) ile birlikte belirtilen sınıflandırma algoritmaları test edilmiştir. Üçüncü aşamada, C-XOR adını verdiğimiz yeni bir öznitelik seçimi yöntemi geliştirildi.

1.2 Literatür Araştırması

Sgandurra ve arkadaşları (2016), veri kümesinin 11 farklı sınıfa ait 582 çalışan fidye yazılımı örneğinden ve 942 iyi uygulama örneğinden oluştuğuna dikkati çekmiştir. Ancak, ilk veri kümesinin 1.450 fidye yazılımı ve 1.131 iyi yazılım örneği ile daha büyük olduğunu ifade etmişlerdir. Bir fidye yazılımını (veya iyi yazılımı) yalnızca API çağrıları kümesi boş değilse, yani fidye yazılımının düzgün bir şekilde çalıştırılabileceği anlamına gelirse analiz ettiğini söylemiştir. Son istatistiklere göre 2015 yılında 140 milyondan fazla yeni kötü amaçlı yazılım örnekleri bulunmuştur. Bunların arasında büyük bir kısmı kötü amaçlı yazılım sınıfı olan fidye yazılımlarından kaynaklanmaktadır. Özel amacı kurbanın sistemini kullanılamaz hale getirmek olan, özellikle önemli dosyaları şifreleyerek ve ardından kullanıcıdan hasarı geri almak için fidye ödemesini istenmiştir. Çeşitli fidye yazılımları, gelişmiş paketleme tekniklerini içerir ve bu nedenle statik olarak analiz edilmesi zordur. Sgandurra ve arkadaşları, dinamik analiz için bir makine öğrenimi yaklaşımı olan EldeRan ve fidye yazılımlarının sınıflandırılması sunmaktadır. EldeRan, fidye yazılımının özellik belirtilerini kontrol ederek, ilk kurulum aşamasında 11sınıfa ait 582 fidye yazılımından oluşan bir veri kümesi üzerinde, bizim testlerimiz tarafından gerçekleştirilen bir dizi eylemi izler. 942 iyi yazılım uygulamasıyla EldeRan'ın 0,995'lik ROC eğrisinin altında bir alana ulaştığını gösterir. Ayrıca EldeRan, önceden tüm bir fidye yazılımı ailesinin mevcut olmasını gerektirmeden çalışır. Bu sonuçlar ile dinamik analizin fidye yazılımı tespitini destekleyebileceğini gösterdiğini, çünkü fidye

yazılımı örnekleri çalışma zamanında aileler arasında ortak olan ve yeni varyantların erken tespitine yardımcı olan bir dizi karakteristik özellik sergilediğini ifade etmiştir. [1].

Khan ve arkadaşları (2020), tezlerinde "Dijital DNA Dizileme Motoru, DNAact-Ran" yöntemini önermiştir. DNAact-Ran, Dijital DNA dizileme tasarım kısıtlamalarını ve k-mer frekans vektörünü kullanmıştır. Önerilen yaklaşımın etkinliğini ölçmek için, doğruluk, geri çağırma, f ölçümü ve doğruluk başarımını ölçmek için 582 fidye yazılımı ve 942 iyi yazılım örneğinde DNAact-Run'ı değerlendirmiş ve diğer yöntemlerle karşılaştırıldığında DNAact-Run'ın fidye yazılımını doğru ve etkili bir şekilde tahmin edip tespit edebildiğini iddia etmişlerdir [2].

DNAact-Ran, fidye yazılımının tespit edilmesinden önce ön işleme ve öznitelik seçme yöntemini uygulamışlardır. Öznitelik seçiminin amacı, sayıyı azaltmak, ilgisiz, gürültülü ve gereksiz öznitelikleri ortadan kaldırmak ve en iyi temsili öznitelikleri seçmektir. MOGWO ve BCS teknikleri, önemli ilgili temsili öznitelikleri seçmek için kullanmışlardır. Başlangıçta, veri kümesi 16383 öznitelik içerdiğini, önişleme aşamasını uyguladıktan sonra 426 özniteliğe düşürdüklerini ve sonrasında MOGWO ve BCS öznitelik seçim algoritması kullanılarak 26 önemli öznitelik seçildiğini ifade etmişlerdir [2].

Yazarlar, [3]'de ortaya koydukları model ile ortalama performansı, karşılaştırma, verilerini sınıflandırma, bir değişken seçme, bir makine öğrenimi modeline uyma, çapraz doğrulama kullanarak bir model seçme işlemi yapmıştır. Doğru sınıflandırma oranı açısından, sadece p değeri 0,01'den küçük olan değişkenleri kullanan rastgele orman modeli, ki-kare testi sonucunda sınıflandırma için % 98,11'de ve tespit oranı açısından en iyi performansı göstermiştir. Fidye yazılımı programlarını doğru bir şekilde fidye yazılımı olarak sınıflandırma oranı, % 97,49 en iyi performansı göstermiştir. Normal programları fidye yazılımı olarak yanlış sınıflandırma oranı olan yanlış pozitif oranı açısından en iyi model, yalnızca % 1,43'te 0,05'ten düşük p değerine sahip değişkenleri kullanan XGBoost modelidir. Değişken seçiminin etkisi incelerken, lojistik regresyon modeli ve destek vektör makinesi, değişken seçiminden önce performansı oldukça azalmış, ancak Rastgele Orman Algoritmasının ve XGBoost'un performansı artmıştır. Lasso düzenlileştirme yöntemini kullanan lojistik regresyon modelinde, seçilen değişken en az ve doğru sınıflandırma oranı ve % 96,84

algılama oranı ile tatmin edici performans göstermiştir. Değişken seçim yöntemine tabi tutulan Rastgele Orman modeli, ki-kare testi göreceli olarak daha yüksek performans gösterdiğini ifade etmiştir. Bir gariplik olarak, karşılıklı bilgi kullanılarak değişken seçimi uygulandıktan sonra, uygun modellerin performansı önceki çalışmalara göre göreceli olarak daha düşük olma eğiliminde olmuş ve bunun daha fazla bilginin kullanılmasından kaynaklandığı doğruladığını ifade etmiştir [3].

Yalnızca binom değişkenlerinden oluşan yüksek boyutlu verileri kullanan özel sınıflandırma durumunda, ki-kare testini kullanan değişken seçim yönteminin, daha az olan bağımsız değişkenleri kaldırarak bazı modellerin performansını artırmaya yardımcı olabileceği doğrulandığını söylemiştir. Bu çalışmada önerilen değişken seçim yöntemi ve modeli, dinamik analiz yoluyla değişkenleri çıkaran ve fidye yazılımını bilgisayara yeni programlar yüklenirken sınıflandıran bir aşı programında kullanılıyorsa, tespit performansında çok fazla gelişme olması beklenir. Ayrıca çalışmasında kullanılan dinamik analiz verisi sayısı 1.524 olduğundan ve fidye yazılımı türleri 11 tür kadar çeşitli olmadığından, daha fazla sayıda ve çeşitli türlerde örneklerin kullanıldığı bir çalışmanın gerekli olduğu düşünülmektedir. Çeşitli bilgileri kullanarak değişkenleri seçmek ve fidye yazılımını tespit etmek için bir yöntem üzerinde araştırma ihtiyacıdır. Ayrıca bu çalışmada kullanılan verilere 8 gizli katmana sahip derin bir sinir ağı (DNN) uygulayarak algılama performansının doğrulanması sonucunda, % 93 ile % 97 arasında doğru sınıflandırma oranı göstermiştir. Gelecekte, derin öğrenme modellerinin performansını tamamlamak için daha fazla araştırma ve fidye yazılımındaki çeşitli gelişen değişikliklerle aktif bir şekilde başa çıkabilen bir hiyerarşik tespit yöntemi yürütüleceğini ifade etmişlerdir [3].

Adamu & Awan (2019), kullanılan veri seti, fidye yazılımını tahmin etmek için bağımsız değişkenler olarak kullanılacak 30.000 öznitelik içerdiğini, ancak, tüm özniteliği analize dâhil etmek zor olduğundan, öznitelik seçimi için bir kavram kanıtı sunmak için beş özniteliği kullandığını, öznitelik seçiminden sonra altı makine öğrenme algoritması üzerinde uygulandığını ifade etmiştir.

Bu yaklaşımların performansını göstermek için, numuneyi eğitim ve test için altı sınıflandırma algoritması üzerinde test etmişlerdir. Her eğitim seti için deney altı kez tekrarlamıştır. Destek vektör makinesi, seçilen diğer makine öğrenme algoritmalarına kıyasla % 88,2'lik en yüksek doğruluğu, 0,179'luk en düşük kök ortalama kare hatası

(RMSE) elde etmiştir. Bu, yaklaşımının, makine öğrenimi tekniklerini uygularken kapsamlı bilgiler sağlayan daha iyi doğruluk ve düşük hata oranıyla fidye yazılımının sınıflandırılmasına yardımcı olduğunu göstermiştir [4].

Adamu'ya göre kötü amaçlı yazılım endüstrileri, binlerce kötü amaçlı yazılım oluşturmak ve değiştirmek için Zeus gibi otomatik bir kötü amaçlı yazılım geliştirme araç seti icat ettiğinden, fidye yazılımı yazarları ekonomik faydalarla yönlendirildiğini ifade etmiştir. Yeni kötü amaçlı yazılım varyantını, yalnızca statik analize dayanan sınıflandırma tekniklerinin sınırlaması nedeniyle eskisinden ayırt etmek çoğu kez zordur. Bu nedenle imza, içerik tabanlı, kalıp eşleştirme gibi teknikler algılama-sınıflandırma için daha az etkili hale gelmiş ve fidye yazılımının tehdit, hedef ve davranışları hakkında iç görü bilgileri sağlamıştır. Güvenlik alanında kötü amaçlı yazılım tespiti için makine öğrenme algoritmaları, kripto fidye yazılıma özgü yüksek performans tespit oranını kanıtlamıştır. Çalışmasında kötü amaçlı fidye yazılımlarının tespiti için denetlenen makine öğrenimi çerçevesi önermiştir. Destek vektörü Makine sınıflandırması, 0,179'luk daha az ortalama kare hatası (RMSE) ve diğer algoritmalardan daha iyi performans gösteren % 88,2'lik doğrulukla daha iyi bir performans göstermiştir. Gelecekte, bulut platformunda makine öğrenimi ve aracı tabanlı mikro-simülasyon kullanarak ağ davranışı tahminine odaklanacağını söylemiştir [4].

Takeuchi vd. (2018), fidye yazılımı nispeten yeni bir kötü amaçlı yazılım türü olduğundan, bilinen kadarıyla, şimdiye kadar sadece birkaç çalışma yapıldığı ifade etmiştir. Nolen vd.'e göre(akt. Takeuchi vd. ,2018) , bir dosya türünün değiştirilmesi, dosyaların benzerliği ve Shannon'ın dosya entropisi dâhil olmak üzere fidye yazılımı algılaması için bir dizi gösterge tasarlamıştır. Açıklık getirmek gerekirse, bir dosya türü yalnızca başlık bilgisi değil, aynı zamanda bir dosyaya özgü baytlardır. Hedeflenen dosyanın dosya türü şifrelemeden sonra değişecektir. Fidye yazılımının oluşturduğu şifrelenmiş dosyaların ortak bir yönü vardır ve bu benzerlik ile ölçülebilir. Bu göstergeler Shannon'ın entropisiyle ölçüldüğünü, bir dosyanın fidye yazılımı tarafından mı yoksa bir kullanıcı tarafından mı şifrelendiğini ayırt etmek için, kötü amaçlı yazılım tespiti için bahsedilen göstergeler birleştirildiğini ifade etmişlerdir [5].

Rieck vd.(2011) (akt. Takeuchi vd.,2018), çalışmalarında ise API çağrılarını inceleyerek Microsoft Windows sistemlerindeki fidye yazılımının kötü amaçlı

etkinliklerini tespit etmişlerdir. Fidye yazılımının yürüttüğü API çağrıları, özniteliği olarak çıkarılır ve ardından bu API çağrılarının öznitellikleri, kötü amaçlı etkinliklerin göstergesi olarak kullanıldığını ifade etmişlerdir.

Takeuchi vd. SVM'leri kullanan bir fidye yazılımı tespit planı önermiştir. Form fidye yazılımı yürütmeleriyle sonuçlanan API çağrı günlüklerinin vektör temsilleri, bir SVM için eğitim örnekleri olarak kullanır. Mevcut çözümden temel fark, önerilen SVM tabanlı şemanın API çağrılarının sıralarını derinlemesine incelemesi ve vektör temsillerimizin yürütme günlüklerindeki q-gram sayısını içermesidir. Bunu yaparak, bilinmeyen fidye yazılımları, daha düşük kötü amaçlı programların tespit edilememe olasılığı ile etkili bir şekilde tespit edilir. Yetkilerin kısıtlandığı sanal bir ortam programı kullanarak, gerçek fidye yazılımları ile dinamik analiz deneyleri gerçekleştirilmiştir. Test sonuçları, önerilen planın tahminlerin doğruluğunu artırdığını ve eksik fidye yazılımı oranını azalttığını gösterdiğini ifade etmiştir [5].

Takeuchi vd.'nin önerisiyle API özniteliklerin önemi değerlendirilmiş ve bahsettiği şekilde tez çalışmamızda SVM makine öğrenmesine de yer verilmiştir.

Bu bölüme kadar, ortak kullanılan veri setiyle çalışma yapan yazarların çalışmalarından bahsedilmiştir. Statik, dinamik ve tersine mühendislik ile ilgili de farklı veri setlerinde çeşitli çalışmalar bulunmaktadır.

Bunlardan birisi 302 virüs ve dll dosya tabanlı dinamik analiz üzerinde çalışan Poudyal vd.(2019)'nin çalışmasında, tersine mühendislik, statik analiz ve makine öğrenimi tekniklerinden yararlanan bir fidye yazılımı algılama çerçevesi çalışması önermiştir. Özellik veri tabanını derleme ve dll düzeyinde ikili dosyaların analizinden oluşturmuş ve makine öğrenmesi modeli geliştirmiştir. Deneysel sonuçları, sekiz denetimli makine öğrenimi sınıflandırıcısı arasında çerçevesinin, birleşik özellik veri kümesi için bunların yedisinde % 90'dan fazla (ortalama % 96,5) bir doğruluk elde edebileceğini göstermektedir. Fidye yazılımı tespit oranı, Lojistik regresyon için minimum % 89,18 ve Rastgele Orman algoritması için maksimum % 97,95 ile birleşik özellik veri setine sahip veri seti için önemli ölçüde daha yüksektir. Çalışmasında, makine öğrenimini kullanarak fidye yazılımı tespiti için ayırt edici özellikler oluşturmak için montaj ve dll düzeyinde ikili dosyaların statik analizinin çok önemli olduğunu iddia etmiştir [6].

Poudyal vd.'nın iddiaları göre;

- Fidye yazılımının tespit doğruluğunu iyileştirmek için dll'leri kullanan otomatik bir statik analiz çerçevesi tasarlamışlardır.
- Farklı seviyelerdeki kodu elde etmek için fidye yazılımı ve normal ikili dosyaların tersine mühendislik uygulamışlardır. Bu kodlar, statik analiz kullanılarak analiz edildi ve çıktıları, makine öğrenimini kullanarak modeli oluşturmak için kullanılmıştır.
- Verilen tüm denetimli makine öğrenimi sınıflayıcıları için hem bireysel hem de birleşik özellikli veri kümesi dikkate alınırken, ortalama fidye yazılımı algılama doğruluğu % 92,11 idi. Daha fazla sayıda fidye yazılımı örneğine sahip olmak ve deneyi derin öğrenme dahil diğer makine öğrenmesi algoritmalarıyla gerçekleştirilmiştir.

Poudral vd. kullandığı tersine mühendislik tekniği, fidye yazılımı algılamada ki başarı oranı ile dll dosya yapısı ve saldırısının önemli olduğu fikrini uyandırmaktadır.

Ahmed vd.(2020), örnek veri setiyle izole bir ortamda otomatik bir dinamik analiz kullanmıştır. Toplam orijinal veri kümesi 2.562 olmasına rağmen, doğru şekilde çalışmayan örnekleri çıkarmıştır. Cuckoo, örnekleri çalıştırmak için belirlediği maksimum zaman aşımı nedeniyle veya farklı fidye yazılımı aile adları atadığı için analizi sonlandırmıştır. 14 farklı dosyadan 673 fidye yazılımı aileler ve 742 iyi huylu örnek analiz etmiştir. Kötü niyetli davranışlarını görmek ve fidye yazılım ana bilgisayarda çalışırken örneklerin yürütme izlerini yakalamak için sanal makine kullanmış, 4 ila 9 dakika aralığına kadar olan her örnek analiz etmiştir. Cuckoo, API çağrıları, ağ trafiği, dosya ve klasör değişiklikleri, işlemler ve bellek dökümleri açısından bilgileri izlemiş ve kaydetmiştir. Bazı fidye yazılımı örnekleri, kötü amaçlı etkinliklerini gerçekleştirmeden önce fare veya klavye olayı gibi insan etkileşimini bekler. Bu nedenle, web sitelerine göz atmak, masaüstündeki belgeleri ve klasörleri tıklamak ve silmek gibi temel kullanıcı etkinliklerini gerçekleştiren bir Python komut dosyası kullanılmıştır. Örneklerin analizi sırasında, kullanıcının verilerini şifrelemek için hem simetrik hem de asimetrik algoritmalar kullanan fidye yazılımı varyantlarını gözlemlemişlerdir. Kripto fidye yazılımı, kurbanın makinesinde AES (Gelişmiş Şifreleme Standardı) algoritması ile rastgele simetrik bir anahtar oluşturur ve ardından bu oluşturulan anahtarla birlikte dosyaları şifrelemiştir. Verileri şifreledikten sonra gizli anahtarı asimetrik şifreleme ile şifrelemiştir. Yürütme sonunda, dosyaları

şifrelemek için fidye yazılımı örnekleri alma süresi Petya gibi 27 saniyeden CryptoWall için 2 dakikaya kadar değişmiştir.

Ahmed vd. bir makine öğrenimi yaklaşımı kullanarak yüksek düzeyde hayatta kalabilen fidye yazılımı (HSR) için davranışsal bir kötü amaçlı yazılım algılama çerçevesi önermiştir. Windows platformlarına bulaşan 673 gerçek dünya fidye yazılımı örneği için otomatik bir dinamik davranış analizi gerçekleştirmiştir. Yeni ortaya çıkan 14 fidye yazılımı ailesinin kötü niyetli davranışlarına odaklanmıştır. Kötü amaçlı program tarafından gerçekleştirilen faaliyetler, kontrollü bir sanal ortamda kaydetmiş ve JSON formatında örneklerin oluşturulmuş raporu almıştır. Dahası, bu araştırmanın temel gözlemi, fidye yazılımı türünün sistemde gerçekte ne yaptığını gösterebilecek entegre bir dizi özelliği araştırmaktır. TF-IDF algoritmasının davranışsal özelliklerin sıralanması ve ağırlıklandırılması için etkili bir yaklaşım olduğu gösterilmiştir. Makine öğrenimi algoritmalarını eğitmek için zaman maliyetini azaltabilecek yedi entegre bilgilendirici özellik sınıfı önermiştir. Destek Vektör Makinesi ve Yapay Sinir Ağı algoritmalarını kullanarak entegre değerli özellikleri kullanarak HSR için bir algılama modeli geliştirmişlerdir [7].

1.3 Hipotez

Fidye yazılımlarının makine öğrenmesi yöntemleri ile tespitinde başarımı artırmak için öznitelik seçimi yöntemleri ile öznitelik sayısı düşürülerek tespit başarımı artırılabilir. Bu bağlamda, mevcut öznitelik seçimi yöntemleri yanı sıra yeni öznitelik seçimi yöntemleri de geliştirilebilir. Bu hipotez çerçevesinde kovaryans matris ve XOR kapısı temelli yeni bir öznitelik yöntemi geliştirildi. Geliştirilen bu yöntem fidye yazılımı veri seti üzerinde test edilerek literatürde yer alan yöntemler ile kıyaslandı. Elde edilen deneysel sonuçlar hipotezin doğruluğunu teyit etmiştir.

1.4 Tez Organizasyonu

Tez metni altı bölümden oluşmaktadır:

Bölüm 1'de, tez çalışmasının amacı, literatür taraması ve hipotezi açıklanmıştır.

Bölüm 2'de, fidye yazılım ve zararlı yazılımlar anlatılmıştır.

Bölüm 3’de, deneysel çalışmalarda kullanılan makine öğrenmesi yöntemleri açıklanmıştır.

Bölüm 4'de, geliştirdiğimiz C-XOR yöntemi tanıtılmıştır.

Bölüm 5'de, veri seti, bulgular ve deneysel sonuçları anlatılmıştır.

Bölüm 6’da, tezin sonuçları ve gelecek vizyonu değerlendirilmiştir.

2. FİDYE YAZILIMLAR

2.1 Zararlı Yazılımlar

Internet, teknik olarak, birçok bilgisayarın ve bilgisayar sistemlerinin birbirine bağlı olduğu, dünya çapında yaygın olan ve sürekli büyüyen bir iletişim ağıdır [14].

Kullanan kişinin izni olmadan kötü amaçla oluşturulmuş yazılımlara zararlı yazılım diyoruz. Bu tür kötücül yazılımlar kişisel verileri çalma, kopyalama, silme veya değiştirme yapabilirler. Ayrıca bu verileri izinsiz olarak internette yer alan her türlü kaynakta istediği gibi paylaşabilir ve bununla ilgili kullanan kişiye maddi veya manevi baskı oluşturabilir. Ayrıca, bu yazılımlar internet ortamında mağdur kişinin bilgilerini kullanarak bazı yasadışı eylemler yapabilir. Bu tür kötücül yazılımları tanımak ve bunlara karşı gerekli önlemleri almak zamanımız dünyasında çok hayati bir konu haline gelmiştir. Zararlı yazılımlar, milenyum çağına girmemizle beraber siber saldırılarda aktif rol oynamış ve saldırılarda yeni yöntemler geliştirilerek birçok devletin, kuruluşun ve şirketin büyük zararlar etmesine neden olmuştur.

Bu tehlikeli yazılımlar reklamlar, bilgilendirme e-postalaarı ve sosyal mühendislik saldırıları gibi çeşitli yöntemlerle kullanıcıların bilgisayarlarına kolayca bulaşabilmektedir. Zararlı yazılımlar, bulaştığı sistemlerdeki dosyaları çözülmesi zor olan şifreleme algoritmaları ile şifreleyerek özel anahtarlar üretmekte ve bu anahtarlar özel bir sunucuda ayrı olarak tutulmaktadır. Ayrıca şifreleme işleminin yanı sıra, bu yazılımın yerleştiği bilgisayarın kayıt defterine girilen kod ile bilgisayar her açıldığında zararlı yazılımın çalışması sağlanmaktadır [16].

Bilgisayar sistemlerindeki dosyaları silerek, isimlerini değiştirerek ya da bilgisayar sistemlerinin işlemci, bellek gibi kaynaklarını tüketen bu zararlı virüslerin tarihçesi, bilgisayar tarihçesi kadar eskidir. 1971 yılında ortaya çıkan Creeper bilinen ilk bilgisayar virüsüdür. Virüslerin bugüne kadar pek çok çeşidi ortaya çıkmıştır ve verdiği zararlar artarak devam etmektedir [18].

Solucan(worm), kurbanın ilgisini ilk olarak vermiş olduğu bağlantıya veya virüs içeren dosyaya çekip, ardından bu dosya veya ilgili bağlantıya tıklatmasını sağlayabilmeyi hedefler. Bu yolla verilen adresi tıklayan kullanıcının bilgisayarına kendini yüklemiş olur. Kullanıcıların e-posta adreslerine iliştirilmiş tüm adresleri kopyalayarak bunların

hepsine veya istediğine mail atabilir, bu maili açan kişilere de kendini bulaştırabilir. Böylece bu e-posta aracılığıyla bütün dünyaya internet ağı üzerinden hızla yayılabilir.

Virüslerin aksine solucanların yayılması ve bulaşması için insan yardımı gerekmez. Bir kez virüs bulaştırdıktan sonra diğer makinelere yayılmak için kullanıcının yardımı olmaksızın bilgisayar ağlarını kullanırlar. E-posta programlarındaki zayıf taraflar gibi ağ güvenlik açıklarından faydalanan solucanlar, sürecin yeniden başlandığı yeni sistemlere virüs bulaştırma umuduyla kendisinin binlerce kopyasını gönderebilir. Çoğu solucan sistem kaynaklarını "yese" ve dolayısıyla performansı düşürse de çoğu artık dosyaları çalmak veya silmek için tasarlanan kötü amaçlı "veri yükleri" içerir [19].

Casus yazılımlar (spyware), bilgi ve bilgisayar güvenliğinde en önemli ve gittikçe yaygınlaşan tehdit ve saldırıların başında gelmektedir. Bu yazılımların kötü niyetli kullanımı sonucunda, bilişim teknolojilerinden yararlanmak isteyen her türlü kullanıcı ve kurum oldukça ciddi zararlara maruz kalmaktadır [20].

Casus yazılım, kullanıcılara ait önemli bilgilerin ve kullanıcının yaptığı işlemlerin, kullanıcının bilgisi olmadan toplanmasını ve bu bilgilerin kötü niyetli kişilere gönderilmesini sağlayan yazılım olarak tanımlanır. Bazı kaynaklarda dar manada “snoopware” (burun sokan yazılım) olarak da adlandırılan casus yazılımlar, diğer kötücül yazılımlara göre özellikle İnternet kullanıcıları tarafından sistemlere farkında olmadan bulaştırılmaktadırlar. Casus yazılımlar, virüs ve solucanlardan farklı olarak hedef sisteme bir kez bulaştıktan sonra kendi kopyasını oluşturarak daha fazla yayılmaya ihtiyaç duymazlar. Casus yazılımın amacı kurban olarak seçilen sistem üzerinde gizli kalarak istenen bilgileri toplamaktır. Bu bilgi kimi zaman bir kredi kartı numarası gibi önemli bir bilgi bile olabilir. Bunun dışında, ticari firmalar İnternet üzerindeki kullanıcı alışkanlıklarını saptamak amacıyla casus yazılımları İnternet üzerinde yayabilmektedirler [15].

Reklam malzemelerini görüntüleyen programlar reklam yazılımı kategorisine girer. Bu uygulamalarda genellikle otomatik olarak internet tarayıcısında reklam içeren yeni bir pencere açar veya tarayıcının giriş sayfasını değiştirir. Reklam yazılımları çoğunlukla ücretsiz sağlanan programlarla birlikte gelir ve (genellikle yararlı olan) bu uygulamaları hazırlayanların geliştirme maliyetlerini karşılamalarına olanak sağlar.

Reklam yazılımı tek başına tehlikeli değildir; kullanıcılar yalnızca reklamlardan rahatsız olur. Gerçek tehlikesi reklam yazılımlarının izleme işlevleri de gerçekleştirebilmesidir.

Ücretsiz sağlanan bir ürün kullanmaya karar verdiğinizde, yükleyici büyük bir olasılıkla ek bir reklam yazılımı programının yüklendiğini size bildirir. Çoğunlukla bu yüklemeyi iptal etmenize ve programı reklam yazılımı olmadan yüklemenize izin verilir [21].

Bilgisayar yazılımı literatüründe Truva atı (trojan) zararlı bir program içeren veya kullanıcının bilgisayarına bu zararlı programı yükleyen yazılımlar anlamına gelmektedir. Bu kavram bilindik Truva Atı hikâyesinden türetilerek bu alanda kullanılmaya başlanmıştır. Truva atları bilgisayar kullanıcısına kullanışlı veya farklı programlarmış gibi görünür, çalıştırıldığında ise kullanıcı açısından zararlı bir hale gelirler.

Truva atı legal bir program içindeki illegal yazılım olup, talimat alır ve bir iş yapıyormuş gibi görünür. Oysaki kullanıcıdan habersiz ve kullanıcının iradesi dışında işlemler yapan bir kötü niyetli yazılımdır. Çeşitli programların içine gizlenen Truva atı çalıştırıldıklarında sisteme zarar verir. Truva atları diğer kötü niyetli yazılımlardan farklı olarak kendi başlarına çalışamazlar. Truva atlarının zararlılığı kullanıcının hareketlerine bağlıdır. Sistem internete bağlandığında dışarıya veri aktarımı yaparlar. Truva atı mitolojideki Truva savaşlarında tahta atın hediye olarak gönderilmesi ile kale içine alınması, ardından atın içine gizlenen düşman askerlerinin kaleyi ele geçirmeleri mantığıyla çalışmaktadır. Truva atları virüs ve solucanlardan farklı olarak yayılmayıp, sadece girdikleri sistem içerisinde faaliyet gösterirler. Bu zararlı yazılımlar genellikle bilgisayarlara indirilen müzik, dosya ve programlara iliştirilmek suretiyle sisteme giriş yapmaktadır. Bu yazılımlar e-postaların araç olarak kullanılması suretiyle de gönderilebilmektedir. Bu zararlı yazılımı kötü niyetli kişiler uzaktaki kişilerin bilgisayarlarından veri hırsızlığı amacıyla kullanmaktadırlar [22].

Botnet, robotun kısaltmış hali olan “bot”, ve network kelimesinin kısaltılması olan “net” birleşmesinden türetilmiştir. Siber suçlular genellikle virüslü bir e-posta eki veya bir bağlantıyla kötü amaçlı yazılımların cihazınıza yerleşmesini sağlar. Bu şekilde bilgisayarlara yerleşmesi sağlanmış uzaktan kumanda edilebilen bir yazılım, uygulama veya kod komut dosyası anlamına gelir. Bot bulaşmış virüslü bilgisayara zombi

denmesinin sebebi, bilgisayarın uzaktan kontrolle yönetilebilecek hale getirilmesinden kaynaklanır. Botlar genellikle veri hırsızlığı, sunucu çökmesi ve kötü amaçlı yazılım dağıtımı gibi toplu saldırıları otomatikleştirmek için kullanılır. Bazı botnetler tamamen yasaldır ve kullanıcı deneyimini iyileştirmek için kullanılır. Arama motorları da tarama işlemi için bot kullandığı için arama sıralamalarını belirleyebilir [19].

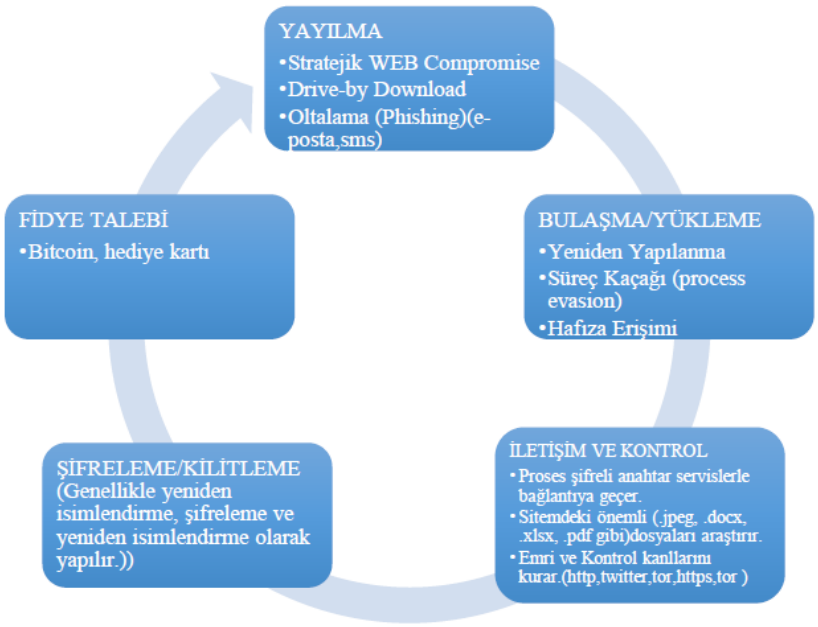
Rootkit, bilgisayara bulaşan, çalışan işlemler arasında kendini gizleyen, kötü niyetli kişilere, uzaktan bilgisayarınızın tam hâkimiyetini sağlayan tespit edilmesi oldukça zor olan bilgisayar programıdır. Virüsler gibi amacı sisteminizi yavaşlatmak ve yayılmak değildir. Bilgisayarınızın kontrolünü ele geçirmek ve bulunduğu sistemde varlığını gizlemektir. Önceleri çok kullanıcılı sistemlerde sıradan kullanıcıların yönetim programlarına ve sistem bilgilerine erişimini gizlemek için geliştirilmiş ve kullanılmış olmasına rağmen, kötü niyetli kullanımına da rastlamak mümkündür [17].

Siber Tehditler ise DDos, Gelişmiş Sürekli Tehdit (Advanced Persistent Threats-APT), Oltalama (Phishing), Sosyal Mühendislik olarak da ifade edebiliriz.

2.2 Fidye Yazılım

Şekil 2.1'de [8] fidye yazılımın tipik işleyiş süreci hakkında bilgi verilmiştir. Fidye yazılımda yayılma e-posta, oltalama, kısa mesaj, dosya indirme gibi işlemlerle gerçekleştiğini görüyoruz. Bu süreci yeniden yapılanma, hafıza erişimi ve süreç kaçağı takip etmektedir. İşlem şifreli anahtar servislerle iletişime geçer. Sistemdeki önemli dosyaları araştırmaya başlar. Bu dosyalar, jpeg, pdf, docx, xlsx dosyaları olabilir. Http, twitter vb. kontrol kanallarını kurar. Yeniden şifreleme ve kilitleme işlemini gerçekleştirir. Son olarak fidye talebinde bulunulur.

Bu bulaşma işlemi sonrasında, sistemde bulaştırdığı kodu çalıştıracak, bulunduğu makinenin "sanal mı, gerçek mi?" olduğunu test edecektir. Windows'un geri dönülmesini önlemek için ilgili ayarlarını (sistem geri yükleme kurtarma vb.) kapatacaktır. API çağrılarıyla gerekli iletişim kanalını sağlayacaktır.



Şekil 2.1 Fidye yazılımı saldırılarının tipik işleyiş süreci.

Dosyalama şifrelemesini yapmaya başlayacak, bitirince erişime kapatacak, sınırlı erişim sunacak ve en sonunda fidye talebinde bulunacaktır.

Kötü amaçlı yazılımların tespiti için üç tür yaklaşım yoğun olarak kullanılmakta ve öne çıkmaktadır. Bunlar; imza tabanlı tespit tekniği, davranış tabanlı tespit tekniği ve sezgisel tespit tekniğidir. İmza tabanlı tespit tekniği, birçok antivirüs yazılımının kullandığı bir tekniktir. İmza ise tek olan ve kısa bir byte dizisinden oluşmaktadır. Bu teknik; bilinen ve daha önceden tespit edilmiş bir kötü amaçlı yazılımın imzası çıkarılıp veri tabanına kaydedildikten sonra test edilecek yazılımın imzasıyla karşılaştırma esasına göre çalışmaktadır. Ancak bilinmeyen, karşılaşılmamış yazılımlarda başarısız olmaktadır. Kötü amaçlı yazılımın zararlarının ortaya çıkması, tespit edilmesi ve veritabanı güncellemesi sürecinde geçen zamanda kötü amaçlı yazılım zarar vermeye devam etmektedir. Polimorfik ve metaformik kötü amaçlı yazılımları tespit edememesi bu yöntemin bir dezavantajı olarak karşımıza çıkmaktadır. Davranış tabanlı tespit tekniği, yazılımın çalışma anındaki davranışlarını izleyerek tespit etmeye çalışmaktadır. Kötü amaçlı yazılımın, mevcut sistemden izole edilmiş kum havuzu (sandbox) yada bir sanal makine (virtual machine) üzerinde çalıştırılarak API çağrıları, sistem çağrıları, internet trafiği gibi özelliklerin toplanıp izlenmesi ilkesine dayalı bir tespit tekniğidir [9].

Üçüncü yöntem ise sezgisel tespitte makine öğrenmesi ön plana çıkmaktadır. Burada API çağrıları ve opcode ile dosyaların davranışları öğrenilmeye buna göre yazılımın iyi veya kötümü olduğu, kötü yazılım ise hangi sınıfa ait olabileceği tahmin edilmeye çalışılmaktadır.

Hedef sistemde aktif hale gelen zararlı yazılım ilk olarak kendini rastgele isimlerle C:/Windows/ dizininin içine kopyalamaktadır. Sistem her açıldığında otomatik olarak aktif hale gelebilmek için sistemin kayıt defterinde bir anahtar girdisi oluşturmaktadır. Bu aşamalardan sonra zararlı yazılım hedefteki tüm dosyaları şifrelemektedir. "IMAGE.E01/Partition2/NONAME [VSC]/ [Current]/ [root]/Users/user/ NTUSER.DAT»Software»Microsoft»Windows»CurrentVersion»Run»" altında bulunan registry kaydından "amdqkacroic.exe" (TeslaCrypt) isimli zararlı yazılım dosyaların şifrelendiği tespit edilmiştir. "amdqkacroic.exe" isimli zararlı yazılım araştırıldığında söz konusu zararlı yazılımın dosyaları şifreledikten sonra kendisini sildiği görülmüştür [10].

AppVeri% dizininden çalıştırma ve yaygın Windows çalıştırılabilir dosyalarıyla aynı adı verilen çalıştırılabilir dosyaları kullandığı da görülmüştür.

Dünyada iki tip fidye yazılımı olduğu kabul görmektedir. Birincisi, "Locker ransomware" adını verdiğimiz kullanıcının sisteme erişimini engellemeyi amaçlayan ve sistemin kilidini açmak için fidye talep edilen türüdür. İkincisi, "Crypto ransomware" ile kurban makinesindeki dosyalarının bir kısmının veya tamamının şifrelendiği kripto türünün olduğu zararlı yazılımdır. Bunların alt kategorisinde olan diğer çeşitlerinden birisi "scaware" olarak bilinen türüdür. Kurban bunu güvenlik yazılımı olarak görür ve kötü yazılım olduğunu belirten pop-up bildirimler alabilir. "Screen Locker" ise bir amblem ile ekranı kilitleyip sizleri resmi bir soruşturmaya uğradığınıza inandırabilir. "Doxware" adını verdiğimiz bir türüde kurban fidye ödemezse dosyaların çevrimiçi yayınlanacağı tehdidinde bulunabilir. Mobil fidye yazılım türüde, kurbaın mobil telefonundaki bilgileri veya cihazı kilitleyerek fidye talebinde bulunabilir. Dünyada 500 milyondan fazla kullanıcıyı koruyan antivirüs yazılım şirketi Bitdefender'ın son Android istatistikleri, Android SLocker fidye yazılımının, 2016 yılının ilk yarısında Danimarka'da kötü amaçlı yazılım bulaştığı raporlanan mobil cihazların neredeyse yarısına, Almanya'nın ise % 25'ine bulaştığını gösteriyor. Avustralya % 21,54 ile 3. sıradayken, İngiltere % 16,48 ile 4. sırada yer

alıyor. Amerika'da ise fidye yazılımın % 16 oranında olduğu görülüyor. Global antivirüs yazılım firması Bitdefender'ın Kasım 2015'te fidye yazılım mağdurlarıyla gerçekleştirdiği araştırmada, kullanıcıların yarısının verilerini geri alabilmek için 500$ kadar ücret ödemeye razı olduğu ve mağdurların yarısının da böyle bir ödeme yaptığı ortaya çıktı. İngilizler kişisel dokümanları, fotoğrafları ve işle ilgili belgeleri için 400 pounda kadar fidye ödemeye razı olurken, Almanlar 210€, Amerikalılar ise 350$ 'ı gözden çıkarabiliyor. Bu durum da, fidye yazılım sektörünü şaşırtıcı miktarda kazançlı hale getirerek, siber suç aktivitelerinin ilerlemesine neden oluyor.

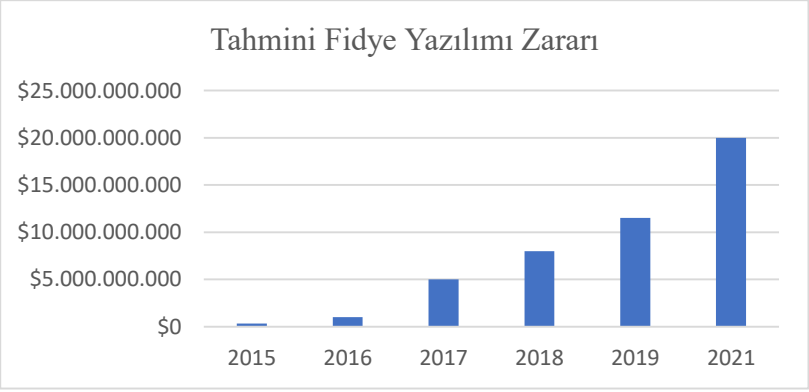
Fidye yazılımların davranış özelliklerini maddeler halinde şu şekilde açıklayabiliriz;

- Yük kalıcılığı: Bir saldırının tamamlanana kadar yürütülmesini sağlamak için, yeniden başlatmalar boyunca devam etmesi ve başlatıldığında devam edebilmesi gerekir. Fidye yazılım, yürütülebilir dosyanın bir kopyasını Windows başlangıç dizinine yerleştirir, bir kayıt defteri çalıştırma anahtarı girişi ekler veya zamanlanmış bir görev ayarlaması yapar.
- Sistem geri yükleme: Kötü amaçlı eylemlerin geri alınamamasını sağlamak için kötü amaçlı yazılım, sistem geri yükleme işlevini devre dışı bırakmayı deneyebilir.
- Gizli teknikler: Kötü amaçlı yazılımlar, kullanıcı tarafından fark edilmekten veya virüs tarayıcıları tarafından tespit edilmekten kaçınmak için gizli bir şekilde çalışmaya çalışır.
- Konumu test etme: Kurulum adımlarını başlatmadan önce sistem ortamını eşleyebilir. Bu genellikle gerçek bir bilgisayarda mı yoksa onu analiz etmeye çalışan bir korumalı alan ortamında mı çalıştığını belirlemek için yapılır.
- Ağ trafiği: İnternet bağlantısı gerektiren fidye yazılımı bunu iki olası görev için yapar: yük ile ilgili dosyaların indirilmesi ve / veya şifreleme anahtarının iletişimi.
- Ayrıcalık yükseltme: Sistemle ilgili kötü amaçlı etkinlikleri yürütmek, kurbanın kullanıcı hesabına verilenlerin ötesinde erişim hakları gerektirebilir. Örneğin, fidye yazılımı, yalnızca Yönetici olarak yapılabilen Ana Önyükleme Kaydının üzerine yazmak isteyebilir. Yalnızca yönetici erişimini istemek işe yarayabilir veya diğer ayrıcalık yükseltme teknikleri kullanılabilir [11].

Fidye yazılımın oluşturduğu maddi zararlar ve örnekleri inceleyelim. Mağdurlar, alenen itiraf etmek istememelerine rağmen, çoğu zaman fidye olarak önemli miktarlarda ödeme yaparlar. Örneğin, tanınmış bir finans firması olan Travelex, 2019'un Aralık ayı sonlarında bir anlaşmanın ardından 2,3Milyon$ ödedi. Sadece kısa bir zaman önce, bir tıbbi merkez operatörü olan DCH Sağlık Sistemi, saldırganlara açıklanmayan bir miktar ödedi. Aslında, şirketlerin bu tür fidyeleri Bitcoin gibi kripto para birimlerinde ödemesi oldukça yaygındır, bu da işlemlerin ve alıcılarının takibini oldukça zorlaştırır. Dahası, çoğu zaman, bu tür suçların failleri, mağdurun ait olduğu hukuki menfaatlerin dışında kalan yer belirleme faaliyetlerinde bulunurlar [12].

Mağdurların listesi sadece özel işletmelerden oluşmuyor; hatta şehir, kasaba ve ilçe hükümetleri gibi devlet kurumları bile bu tür fidye ödemeye mecbur bırakıldı. Örneğin Florida'da iki şehir, Lake City ve Riviera Beach sırasıyla 500.000$ ve 600.000$ ödemek zorunda kalırken, Indiana'daki Le PorteCounty 130.000$ ödedi. Ağustos 2019 itibarıyla, 70'ten fazla eyalet ve yerel hükümet fidye yazılımı saldırılarına maruz kaldı ve bunların birçoğu sonunda saldırganların taleplerine boyun eğdi. Reddettikleri bazı durumlarda, çok daha büyük bir kayıp ve kurtarma maliyeti ile vuruldular; örneğin, Atlanta Şehri Mart 2018'deki bir ihlalin ardından 51.000 $ 'lık bir fidye ödemeyi reddettiğinde, sistemlerini yeniden inşa etmek için büyük bir 17Milyon$ harcamak zorunda kaldı. Kısacası, Baltimore Şehri bir fidye yazılımı talebine uymayı reddettiğinde benzer bir kaderi yaşadı [12].

Fidye yazılımın oluşturduğu zararın yıllık tahmini rakamları şekil 2.2'de görülmektedir [13]. Buna göre, 2021 yıl sonu tahmini zararın 20Milyar$ ulaşması beklenmektedir.



Şekil 2.2 Fidye yazılımlarının yıllık neden olduğu tahmini zararlar.

3. MAKİNE ÖĞRENME MODELİ

Günümüzde makine öğrenmesi (machine learning), mühendislik ve finansın başını çektiği pek çok alanda yaygın olarak kullanılmaktadır. Bu alanlardan biri olan sosyal medya kullanımı beraberinde verilerin depolanması fikrini ortaya koymuş ve veriler gün geçtikçe çığ gibi büyümeye başlamıştır. Verilerin büyümesi sadece insanlar tarafından değil aynı zamanda kullanılan sistemler tarafından da artırmaya devam etmiştir. Büyük veri yığınları kullanarak bir karar destek sistemi (decision support system) modeli kurmak için, temel koşul ifadeleri (if-else) ile oluşturulacak yazılımların kural sayısı yetersiz kalacaktır. Örneğin, milyonlarca müşterisi olan bir alışveriş sitesinin ürün tavsiye motoru, her müşterisi için özel ürünler önermektedir. Böyle bir sistemi geleneksel programlama teknikleri ile oluşturmak neredeyse imkânsızdır. Çünkü geleneksel yöntemlerle geliştirilen yazılımlar, en başta tanımlanan algoritmaya göre çalışırlar. Bu yazılımlar sadece algoritmada tanımlanan koşullara göre işlemler gerçekleştirirler. Makine öğrenmesi algoritmaları, geleneksel programlama algoritmalarından farklıdırlar. Çünkü makine öğrenmesi algoritmaları öğrenen algoritmalar oldukları için, artan veri miktarı ile birlikte kural sayılarını da dinamik olarak artırırlar. Bir yüz tanıma sistemi geliştirmek istendiğinde, sadece insan yüzünün matematiksel modelinin bilinmesi veya görüntü işleme (image processing) algoritmalarının çalıştırılması, performans açısından yetersiz kalabilir. Bilgisayarlar, milyonlarca insan yüzünü sadece pikseller topluluğu olarak değerlendirmektedir. Bu piksel yığınları içinden, çok boyutlu dizi teknikleri kullanılarak birbirinin aynı olan piksellerin bulunması gerekir. Çok fazla işlem yükü gerektiren bu zorlu süreci, öğrenebilen makine öğrenmesi algoritmaları ile yönetmek, daha doğru bir yaklaşım olacaktır.

Google'ın otonom aracının her geçen gün daha başarılı performans göstermesinde, makine öğrenmesi algoritmaları önemli bir yer tutmaktadır. Sürücülerin trafikte karşılaştıkları anlık gelişmelere verdikleri tepkiler için aldıkları trafik eğitiminin yanı sıra, her sürüşte elde ettikleri tecrübelerin de önemi büyüktür. Bir sürücünün trafikteki araçları, trafik işaretlerini, trafik lambalarını, yayaları, hayvanları, hava koşullarını ve daha birçok etkeni çok kısa sürede değerlendirmesi gerekir. Bir otonom aracın seyir halindeyken bu kadar çok etkeni değerlendirebilmesi, sadece çok gelişmiş algılayıcılar veya kameralar kullanılması ile gerçekleştirilemez. Araç kameralarından anlık görüntülerin alındıktan sonra doğru analiz edilmesi, iyi bir görüntü işleme süreci

gerektirir. Sonraki aşamada ise elde edilen görüntülerin, makine öğrenmesi algoritmaları ile doğru şekilde sınıflandırılması gerekir. Örneğin, bir insan görüntüsü makine öğrenmesi tarafından hatalı bir obje olarak sınıflandırılırsa, bu hata ölümle sonuçlanabilecek bir kazaya sebep olabilir. Google'ın otonom aracı, makine öğrenmesi algoritmaları sayesinde kararlar alarak binlerce kilometre yol yapmaktadır. Ayrıca' algılayıcılar ve kameralar ile sürekli olarak yeni veriler toplanması, başarılı olarak sınıflandırılan nesne sayısının artmasına yardımcı olmaktadır.

Bir diğer Google hizmeti olan Google Çeviri, bir çeviri gerçekleştirdikten sonra kullanıcıdan nasıl daha iyi bir çeviri yapılabileceği ile ilgili öneriler almaktadır. Böylece çeviri hizmeti, kullanıcıların önerilerinden öğrenerek gün geçtikçe daha iyi çeviriler yapabilmektedir.

Finans kuruluşları ve şirketler, borsadaki hisse senetlerinin seyri hakkında makine öğrenmesi algoritmalarına dayanan öngörü sistemleri kullanmaktadırlar. Bankalar müşterilerine kredi verirken ve kredi kartı dolandırıcılıklarını tespit ederken, makine öğrenmesine dayalı algoritmalardan faydalanarak kararlar alırlar. Şirketlerin insan kaynakları birimlerinde şirket çalışanları seçilirken makine öğrenmesine dayanan yazılımlar kullanılabilmektedir. Bugün çok büyük bir endüstri haline gelen futbol müsabakalarının sonuçlarının tahmin edilmesinde, istatistiki bilgilerin yanı sıra makine öğrenmesi algoritmalarının da önemli bir yer tuttuğu görülmektedir. Son yıllarda medikal görüntü işleme alanı için geliştirilen makine öğrenmesi algoritmaları, özellikle bir tümörün kanser olup olmamasının belirlenmesi gibi sınıflandırma problemlerinde kullanılmaktadır [23].

Çalıştığımız veri kümelerinde verilerin büyük kısmı belli bir sınıfta yığılmış olabilmektedir. Bu verileri makine öğrenmesine verdiğiniz zaman makine yüksek doğruluk oranını yakalayabilmek için kendine en kolay yöntemi, olasılığı deneyerek yüksek veriye sahip tek bir sınıfa daha çok veri sonucu atayıp doğruluk (accuracy)/çapraz doğrulama (cross-validation) verebilmektedir. Biz bu değerlerde hata matrisi (confusion matrix) incelediğimiz zaman yüksek doğruluk oranının sadece bir sınıftan dolayı kaynaklandığını gerçekte sonuçların güvenilir olmadığını farkedeceğiz.

Literatürde dengeli olmayan bu veri setleri (Imbalanced datasets) için yapabileceğimiz yöntemlerden birisi sınıf dengesi kurmaktır. Bu denge üç şekilde kurulabilir.

- Birincisi, az örnekleme (undersampling) yöntemidir. Dengesizlik oranını azaltmak için örnekleri veri kümesinden kaldırır. Bu etkiyi elde etmenin en basit yolu, çoğunluk sınıfından bazı öğeleri rastgele atmaktır. Çoğu yetersiz örnekleme yöntemi, azınlık sınıfını dokunulmadan bırakır ve yalnızca çoğunluk sınıfı unsurlarını ortadan kaldırır, ancak ilkinin (küçük) bir azaltılmasına izin vermenin kendi yararları olduğu gösterilmiştir [24].
- İkincisi yüksek hızda (oversampling) örneklemedir. Bu teknikler zıt yaklaşımı benimser. Çoğunluk sınıfını küçültmek yerine, azınlık sınıfının boyutu büyütülür. Yeni azınlık unsurları, orijinal örneklerin kopyaları veya karışık varyantları veya enterpolasyon sonuçları olarak eğitim setine eklenir. En basit yaklaşım olarak, rastgele alt örnekleme tekniğine benzer şekilde, rastgele yüksek hızda örnekleme, çoğaltma için azınlık sınıfı öğelerini rastgele seçer. Muhtemelen en popüler yüksek hızda örnekleme yöntemi, mevcut azınlık unsurları ile en yakın azınlık sınıfı komşularından biri arasında doğrusal enterpolasyon yoluyla yeni azınlık örnekleri inşa eden 'smote' tekniğidir. 'Smote' uygulamasının azınlık sınıfının aşırı genelleşmesine neden olabileceği defalarca belirtilmiş ve bu konuyu ele almak için birkaç uzantı önerilmiştir. Dahası, bulut sunucusu oluşturma prosedürü her zaman haklı değildir. Bu uzantılar üç gruba ayrılabilir. 'Smote' ile birlikte bir ön veya son işleme tekniği uygulayan yöntemler, küçüklerin oluşumunu sınırlayan yöntemler yada özellik uzayının belirli bölgelerine aitlik elemanları ve eleman oluşturma sürecini değiştiren yöntemler.
- Üçüncü ise Karma yöntemlerdir. Karma veri seviyesi çözümleri, veri dengeleme yaklaşımları içinde düşük örnekleme ve yüksek hızda örneklemeyi birleştirir. Ya aşırı ve az örneklemeyi birleştirirler (her iki prosedürü aynı anda gerçekleştirirler) veya bir veri temizleme adımı olarak yüksek hızda örneklemeden sonra düşük örnekleme gerçekleştirirler. Aslında, 'Smote' ile son işlemeyi birleştiren yüksek hızda örnekleme yöntemleri ikinci gruba uygundur [24].

3.1 Öznitelik Seçimi

Veri seti içinde makine modellememizi sağlayacak her bir sütun öznitelik olarak adlandırılmaktadır. Bu öznitelikler yaş, cinsiyet, meslek, kişisel özellikler, hedeflenen verinin ayırt edici özellikleri, alışveriş siteleri için ürün özellikleri vb. olabilir.

Öznitelik seçimi ise bu özniteliklerin içinde bize en faydalı olabilecek olanları seçme işlemidir. Bazı öznitelikler gereksiz olabilir. Performansa hiçbir katkısı olmaz. Sadece veri olarak yer işgal eder. Bu öznitelikler ayıklanmadığı zaman makine öğrenmesi modeline girer ve modelin çalışma yükünü artırır. Veri setinin öğrenme aşamasında aşırı uyum gösterme (overfitting) adını verdiğimiz durum gerçekleşir. Test aşamasında bu performansı gösteremediği zaman hata oranlarının artmasına neden olur. Modellemenin çalışma süresi artar. Çok büyük verilerde çalıştığınız zaman bu süre günlerce uzayabilir.

Yukardaki sebeplerden dolayı bir modelleme yapıyorsanız muhakkak öznitelik seçimi yapmak zorundasınız. Öznitelik yöntemlerini üç ana kısıma ayırabiliriz.

- Filtre yöntemi: Burada hedeflediğimiz değişken ile öznitelikler arasındaki ilişki dikkate alarak, altküme oluşturulur. Bu kısımda tercih edilen yöntemlerden birisi pearson korelasyonudur. Doğrusal olarak ilişkili değişkenler arasında ilişki derecesini ölçerek çalışır. Diğer yöntem Ki-kare yöntemidir. Cinsiyet gibi kategorileştirilen özniteliklerde çalışmaktadır. Boy kilo vb gibi sayısal değerlerde çalışmamaktadır. Hedef değişkenle, özniteliğin bağımlı olup olmadığını test sonucunda atadığı bir p değeriyle ölçmektedir.
- Sarmalayıcı (wrapper) yöntem: Bir alt gruba ait modeller oluşturur ve performans ölçer. Bu işlem iki farklı yöntem olan ileri (forward) arama veya geri (backward) arama ile yapılmaktadır. İleri aramada, boş bir öznitelik kümesiyle başlayıp, birer birer yeni öznitelikler eklenerek devam eder. Her seferinde bu özniteliklerin kalitesi ölçülür. Geri aramada, en iyi öznitelikleri bulabilmek için tüm kümeden başlayarak her adımda en kötü performansı sergileyen öznitelikler elenerek işlemi gerçekleştirir.
- Gömülü (embedded) yöntem: Makine öğrenimi modellerinin sağladığı iç görülerle elde edilen yöntemdir. Kendi içinde Lasso lineer regresyon, RF algoritması gibi öz nitelik seçimleriyle işlemi yapar.

  Gömülü yöntemlerin avantajları birisi, en etkili girdi değişkenini dikkate almasıdır. Bir başka avantajı, Hızlı çalışabilen bir yapıya sahip olmasıdır. Doğruluk oranı ise filtre yöntemlere göre daha fazla ve aşırı uyuma göre daha duyarlı olması da diğer avantajlarıdır.

3.1.1 Genetik Algoritma

Genetik Algoritma fikri J. Holland'a aittir (1975). Holland, arkadaşları ve öğrencileri ile birlikte bu algoritmayı geliştirmiş ve ilk çalışmalarının sonucunda "Adaptation in Natural and Artificial Systems" isimli kitabını çıkartmıştır.

Genetik Algoritma, yönlendirilmiş rasgele araştırma algoritmalarının bir türüdür. Tabii seçme (selection) ile canlılarda bulunan genetik gelişimin benzetişimini gerçekleştirmektedir. Algoritma diğer evrimsel algoritmalar gibi araştırma uzayında bulunan çözümlerin bazılarının oluşturduğu bir başlangıç popülasyonunu (initial population) kullanmaktadır. Başlangıç popülasyonu her jenerasyonda (generation), tabii seçme (natural selection) ve tekrar üreme (reproduction) işlemleri vasıtası ile art arda geliştirilir. En son kuşağın en uygun yani en kaliteli (fittest) bireyi, problem için optimal çözüm olmaktadır. Bu çözüm her zaman optimum olmayabilir ama kesinlikle optimuma yakın bir optimal çözümdür.

Holland , basit bit dizileri (bit strings) kullanarak karmaşık yapıların kodlanabileceğini göstermiştir. Yapılar, çözülecek problem için çözümleri temsil etmektedir. Bunlar, muhtemel tüm çözümleri içine alan araştırma uzayından alınır ve bu dizilerin veya çözümlerin belirli bir miktarı Genetik Algoritmanın kullanacağı popülasyonu oluşturur. Daha sonra temel genetik operatörlerin belirli bir seti, ard arda gelen kullanılır. Şayet bu işlem uygun şekilde kontrol edilirse, çözüm popülasyonunun ortalama kalitesi çok hızlı olarak gelişme gösterir. Yani, çözülecek probleme çok iyi uyarlanmış yapıları içeren çözüm kümesinin ortaya çıkması sağlanır.

Önce başlangıç popülasyonu oluşturmakta ve sonra tabii seçme işlemi ile birlikte genetik operatörler çaprazlama (crossover) ve mutasyon gelecek jenerasyondaki çözümleri üretmek amacıyla kullanılmaktadır. Kalite veya uygunluk değerlendirme (fıtness evaluation) mi tekrar üreme olayında uygulanan seçme işlemini gerçekleştirebilmek için her bir bireye uygulanmaktadır. Birbirini takip eden jenerasyonların ve değerlendirilmesi çevrimi, optimal bir çözüm bulununcaya kadar devam etmektedir. Basit bir Genetik Algoritma beş temel elemandan oluşmakta ve bunların her biri algoritmanın performansını önemli derecede etkilemektedir. Bu parametreler: çözümlerin temsil (representation) şekli, başlangıç popülasyonu oluşturma yöntemi, uygunluk veya kalite değerlendirme (fitness evaluation) fonksiyonu, kullanılan genetik operatörler ve kontrol parametreleridir [25].

3.1.2 Doğrusal İleri Seçim Yöntemi

Doğrusal İleri Seçme yönteminde aşağıda verilen adım sıralaması izlenir:

- Modelde hiç değişken olmadan başlanır.
- Bağımsız değişkenler modele eklendiğinde anlamlılık düzeyleri kontrol edilir. Eşik değerinden daha düşük olan en düşük anlamlılık düzeyine sahip bağımsız değişken seçilir.
- Yeni bağımsız değişkenler ilave edilene kadar devam edilir [23].

3.1.3 Parçacık Sürü Optimizasyonu Yöntemi

Basit kurallarla hareket eden çok sayıda ajanın (parçacığın) kompleks davranışları oluşturmak amacıyla kullanılma fikri, ilk olarak doğal olayları taklit eden bilgisayar animasyonlarında görüntülerin birleştirilmesi problemini çözmek amacıyla kullanılmıştır. Reeves (1983), yılında bulanık bir nesnenin gerçekleştirilmesi için birlikte çalışan bir parçacık sistemini oluşturmuştur. Bu parçacık sistemi, stokastik olarak hareket eden bir noktalar dizisini üretir ve bunlara tipik olarak daha önceden belirlenmiş noktalar başlangıç değerleri olarak atanır. Aynı zamanda her parçacığa bir başlangıç hız vektörü de atanmaktadır. Grafiksel benzetimlerde, bu noktalar sınırlı doku, ömür ve renk gibi ilave karakteristiklere sahip olabilir. Hız vektörleri tekrarlamalı olarak bazı rasgele değer alan faktörlerle ayarlanır. Her parçacık, hız vektörünü kullanarak mevcut pozisyonundan hareketi daha gerçekçi yapan sınırlı bir açı içerisinde ayarlama yapmak suretiyle hareket ettirilir. Bazı animasyonlarda, basit parçacıklardan çok daha yüksek seviyeli dinamiklerle grup davranışını bir araya getirmek gerekmektedir. Reynolds (1987), modellenen cisim açısından çok daha yüksek dereceli sürü algoritmasına temel olarak Reeves'in tanımladığı parçacık sistemini kullanmıştır. Reynolds, parçacık hareketini alıp buna oryantasyon ve iç-cisim iletişimini dahil etmiştir. Bu ilave davranışlar, bireysel kuş benzeri cisimlerin (boids) bazı basit sürü kurallarını takip etmesini mümkün kılmıştır: Cisimler, yakındaki cisimlere çarpışmaktan kaçınmalı; birbirlerinin hız vektörlerini uygunlaştırmak için çaba sarfetmeli ve birbirlerine yakın kalmaya çalışmalıdırlar. Bireylerin zekâsını artıran bu modelin geliştirilmesi, bireysel yörüngelerin oluşturulma ihtiyacını ortadan kaldırmıştır. Kennedy (1995), sosyal davranışları tanımlamak amacıyla Reynolds'ın modelini genişletmek istemişlerdir. Daha önemlisi, alternatif bir sürü algoritması tanımlayan Hepper ve Grenander (1990) tarafından önerilen basit

hedefi çok daha gerçekçi bir hedefle yani "yiyecek araştırma" hedefiyle yer değiştirmişlerdir. Böylece parçacık sürü optimizasyon algoritması ortaya çıkmıştır [25].

3.2 Sınıflandırma

Makine öğrenmesinde sınıflandırma, bilgisayar programının verilen veri girişinden öğrendiği ve sonrasında yeni gözlemleri sınıflandırmak için bu öğrenmeyi kullandığı denetimli öğrenme yaklaşımıdır. Bu veri setleri iki farklı sınıf olabileceği gibi, ikiden daha fazla sayıda sınıf da olabilir. Çoklu sınıflandırma yaptığımız bu çalışmamızda Naif bayes, k-En Yakın Komşuluğu, Çok Katmanlı Algılayıcı Ağ, Derin öğrenme, Rastgele Orman, Destek Vektör Makineleri başlıca kullandığımız algoritmalardır.

3.2.1 Rastgele Orman Sınıflandırması

Sınıflandırma doğruluğundaki önemli gelişmeler, bir ağaç topluluğunun büyümesinden ve en popüler sınıf için oy vermelerine izin verilmesinden kaynaklanmıştır. Bu toplulukları büyütmek için, topluluktaki her ağacın büyümesini yöneten genellikle rastgele vektörler oluşturulur. Erken bir örnek torbalamadır. Her ağacın yetiştirileceği eğitim setindeki örneklerden rastgele bir seçim (değiştirmeden) yapılır. Diğer bir örnek ise rastgele bölünmüş seçimdir. Burada her düğümde bölünme en iyi K bölünmeleri arasından rastgele seçilir. Breiman, orijinal eğitim setindeki çıktıları rastgele hale getirerek yeni eğitim setleri oluşturur. Diğer bir yaklaşım, eğitim setindeki örnekler üzerinde rastgele seçilen ağırlıklardan eğitim setini seçmektir. Her bir ağacı büyütmek için kullanmak üzere bir özellik alt kümesinin rastgele bir seçimini yapan "rastgele alt uzay" yöntemi üzerine bir dizi makale yazmıştır. Yazılı karakter tanıma üzerine önemli bir makalede, Amit ve Geman çok sayıda geometrik özelliği tanımlar ve her düğümde en iyi bölünme için bunların rastgele bir seçimini araştırır [30]

Tüm bu prosedürlerdeki ortak unsur, k'inci ağaç için rasgele bir k vektörünün, geçmiş rasgele vektörlerden bağımsız olarak, Θ_1, ..., Θ_{k-1}, ancak aynı dağılımla üretilmesidir; ve bir ağaç, eğitim seti ve Θ_k kullanılarak büyütülür ve x'in bir giriş vektörü olduğu bir sınıflandırıcı $h(x,\Theta_k)$ ile sonuçlanır. Örneğin, torbalamada rastgele vektör Θ, kutulara rastgele atılan N darttan kaynaklanan N kutu içindeki sayılar olarak üretilir; burada N, eğitim setindeki örneklerin sayısıdır. Rastgele bölünmüş seçimde Θ, 1 ile K arasında bir dizi bağımsız rastgele tamsayıdan oluşur. Θ 'nin doğası ve boyutluluğu,

ağaç yapımında kullanımına bağlıdır. Çok sayıda ağaç oluşturulduktan sonra, en popüler sınıfa oy verirler. Bu prosedürlere rastgele ormanlar diyoruz.

Rastgele bir orman, ağaç yapılı sınıflandırıcıların bir koleksiyonundan oluşan bir sınıflandırıcıdır {h (x, Θ_k), k = 1, ...} burada {Θ_k} bağımsız, özdeş olarak dağıtılmış rasgele vektörlerdir ve x girişindeki en popüler sınıf için her ağaç bir birim oyu verir.

H1 (x), h_2 (x), ..., h_K (x) sınıflandırıcılarından oluşan bir topluluk verildiğinde ve rastgele Y, X vektörünün dağılımından rasgele çekilen eğitim setiyle, margin fonksiyonu (Denk.1)'de tanımlanmıştır.

$$mg(X,Y)=av_kI(h_k(X)=Y)- \max_{j\neq Y} av_kI(h_k(X)=j) \quad (1)$$

Burada I (•) gösterge işlevidir. Teminat, sağ sınıf için X, Y'deki ortalama oy sayısının başka herhangi bir sınıf için ortalama oyu aşma kapsamını ölçer. Marj ne kadar büyükse, sınıflandırmaya o kadar güvenir. Genelleme hatası (Denk.2)'de verilmiştir.

$$PE^*=P_{X,Y}(mg(X,Y) < 0) \quad (2)$$

Burada X, Y alt simgeleri olasılığın X, Y uzayının üzerinde olduğunu gösterir. Rastgele ormanlarda, h_k (X) = h (X, Θ_k). Çok sayıda ağaç için, Büyük Sayıların Güçlü Yasası ve aşağıdaki ağaç yapısından izler:

Ağaç sayısı arttıkça, neredeyse kesin olarak tüm Θ_1, ... PE * dizileri (Denk.3)'de görülmektedir.

$$P_{X,Y}(P_\Theta(h(X,\Theta)=Y)-\max_{j\neq Y} P_\Theta(h(X,\Theta)=j)<0) \quad (3)$$

Bu sonuç, rastgele ormanların neden daha fazla ağaç eklendikçe fazla sığmadığını, ancak genelleme hatasının sınırlayıcı bir değerini ürettiğini açıklar [30].

3.2.2 Çok Katmanlı Algılayıcı Ağ

Derin öğrenme, birden çok işleme katmanından oluşan hesaplama modellerinin, birden çok soyutlama düzeyiyle verilerin temsillerini öğrenmesine olanak tanır. Bu yöntemler, konuşma tanıma, görsel nesne tanıma, nesne algılama ve ilaç keşfi ve genomik gibi diğer birçok alanda son teknolojiyi önemli ölçüde geliştirdi. Derin öğrenme, bir makinenin önceki katmandaki temsilden her katmandaki gösterimi hesaplamak için kullanılan dâhili parametrelerini nasıl değiştirmesi gerektiğini belirtmek için geri yayılım algoritmasını kullanarak büyük veri kümelerindeki

karmaşık yapıyı keşfeder. Derin evrişimli ağlar görüntülerin, videoların, konuşmaların ve seslerin işlenmesinde çığır açmıştır; yinelenen ağlar ise metin ve konuşma gibi sıralı verilere ışık tutmuştur.

Derin olsun ya da olmasın en yaygın makine öğrenimi biçimi denetimli öğrenmedir. Görüntüleri örneğin bir ev, bir araba, bir kişi veya bir evcil hayvan içerecek şekilde sınıflandırabilen bir sistem inşa etmek istediğimizi hayal edin. Önce, her biri kendi kategorisiyle etiketlenmiş büyük bir ev, araba, insan ve evcil hayvan görsel veri seti topluyoruz. Eğitim sırasında, makineye bir resim gösterilir ve her kategori için bir puan vektörü şeklinde bir çıktı üretir. İstenen kategorinin tüm kategoriler arasında en yüksek puana sahip olmasını istiyoruz, ancak bunun eğitimden önce gerçekleşmesi pek olası değil. Çıktı puanları ile istenen puan örüntüsü arasındaki hatayı (veya mesafeyi) ölçen objektif bir işlevi hesaplıyoruz. Makine daha sonra bu hatayı azaltmak için dahili ayarlanabilir parametrelerini değiştirir. Genellikle ağırlıklar olarak adlandırılan bu ayarlanabilir parametreler, makinenin giriş-çıkış fonksiyonunu tanımlayan "topuzlar" olarak görülebilen gerçek sayılardır. Tipik bir derin öğrenme sisteminde, makineyi eğitmek için bu ayarlanabilir ağırlıklardan yüz milyonlarca ve yüz milyonlarca etiketli örnek olabilir. Ağırlık vektörünü doğru bir şekilde ayarlamak için, öğrenme algoritması, her ağırlık için, ağırlık küçük bir miktar artırılırsa hatanın ne kadar artacağını veya azalacağını gösteren bir gradyan vektörü hesaplar. Ağırlık vektörü daha sonra gradyan vektörünün tersi yönde ayarlanır.

Tüm eğitim örneklerinde ortalaması alınan amaç işlevi, ağırlık değerlerinin yüksek boyutlu uzayında bir tür engebeli manzara olarak görülebilir. Negatif gradyan vektörü, bu manzaradaki en dik iniş yönünü gösterir ve onu minimuma yaklaştırır, burada çıktı hatası ortalama olarak düşüktür.

Uygulamada, çoğu uygulayıcı stokastik gradyan iniş (SGD) adı verilen bir prosedür kullanır. Bu, birkaç örnek için girdi vektörünü göstermeyi, çıktıları ve hataları hesaplamayı, bu örnekler için ortalama gradyanı hesaplamayı ve ağırlıkları buna göre ayarlamayı içerir. İşlem, eğitim setinden amaç fonksiyonunun ortalaması düşmeyi bırakana kadar birçok küçük örnek grubu için tekrarlanır. Stokastik olarak adlandırılır çünkü her küçük örnek grubu, tüm örnekler üzerinde ortalama gradyanın gürültülü bir tahminini verir. Bu basit prosedür, çok daha ayrıntılı optimizasyon teknikleriyle karşılaştırıldığında genellikle şaşırtıcı derecede hızlı bir şekilde iyi bir ağırlık seti

bulur. Eğitimden sonra, sistemin performansı test seti adı verilen farklı bir dizi örnek üzerinde ölçülür. Bu, makinenin genelleme yeteneğini test etmeye hizmet eder.

Makine öğreniminin mevcut pratik uygulamalarının çoğu, elle tasarlanmış özelliklerin yanı sıra doğrusal sınıflandırıcılar kullanır. İki sınıflı bir doğrusal sınıflandırıcı, özellik vektörü bileşenlerinin ağırlıklı toplamını hesaplar. Ağırlıklı toplam bir eşiğin üzerindeyse, girdi belirli bir kategoriye ait olarak sınıflandırılır.

3.2.3 k-En Yakın Komşuluğu

En yakın komşu veya örnek tabanlı öğrenme olan tembel öğrenmeyi kullanır. Tembel öğrenme, hesaplamanın sınıflandırma zamanına kadar geciktirilmesiyle ayırt edilir. Eğitim sırasında karar ağaçları veya kurallar gibi açık teoriler oluşturulmaz. Bir test örneğiyle karşı karşıya kaldığında, tembel öğrenme algoritmaları bir tahmin yapmak için eğitim örneklerine erişir. Böyle bir algoritma LAZYDT'dir. Diğer ismiyle Tembel karar ağacı öğrenme algoritması olarak adlandırılır. Sınıflandırılacak her örnek için, LAZYDT bir entropi ölçüsü kullanarak örneğe en uygun olan bir kural oluşturur. Kuralın öncülü, öznitelik-değer çiftleri biçimindeki koşulların birleşimidir. Kuralın sonucu, kuralın öncülünü karşılayan eğitim örneklerinin çoğunluk sınıfı olan tahmin edilecek sınıftır. Bu, karar ağacı öğreniminin küçük ayrık problemini hafifletmek için başka bir yaklaşım sağlar [26].

Makine öğrenmesi KNN Bir x örneği ve bunun ilişkili etiket kümesi Y $\subseteq \mathcal{y}$ verildiğinde, KNN'lerin ML-KNN yönteminde dikkate alındığını varsayalım. $\overrightarrow{y_x}$, x için kategori vektörü olsun, burada onun l 'inci bileşeni $\overrightarrow{y_x}(l)$ ($l \in \mathcal{y}$) ise 1 değerini ve aksi takdirde 0 değerini alır. Ek olarak, N (x), eğitim setinde tanımlanan x'in KNN setini göstersin. Böylece, bu komşuların etiket setlerine dayalı olarak, bir üyelik sayma vektörü (Denk.4)'de tanımlanabilir:

$$\overrightarrow{C_x}(l) = \sum_{a \in \mathrm{N(x)}} \overrightarrow{y_a}(l),\ l \in \mathcal{y} \qquad (4)$$

$\overrightarrow{C_x}(l)$ l'inci sınıf ait olan x'in komşularının sayısını sayar.

Her test örneği t için, ML-KNN ilk olarak eğitim setinde KNN'lerini N (t) tanımlar. H_1^l, t'nin etiketi l olan olay, H_0^l,ise t'nin etiketi l olmayan olay olsun. Ayrıca, E_j^l, (j $\in$ {0, 1,.., K}), t'nin KNN'leri arasında, tam olarak l etiketine sahip j örnekleri olduğu

olayı göstersin. Bu nedenle, üyelik sayma vektörü $\overrightarrow{C_t}$ dayalı olarak, kategori vektörü $\overrightarrow{y_t}$ (Denk.5)'de MAP ilkesi kullanılarak belirlenir:

$$\overrightarrow{y_t}(l) = \underset{b\in\{0,1\}}{\arg\max} \; P\left(H_b^l \mid E^l_{\overrightarrow{C_t}(l)}\right), (l \in \mathcal{y}) \quad (5)$$

Bayesian kurallarını kullanarak (Denk.6) yazılabilir:

$$\overrightarrow{y_t}(l) = \underset{b\in\{0,1\}}{\arg\max} \; P\left(H_b^l\right) P\left(E^l_{\overrightarrow{C_t}(l)} \mid H_b^l\right) / P\left(E^l_{\overrightarrow{C_t}(l)}\right)$$

$$= \underset{b\in\{0,1\}}{\arg\max} \; P\left(H_b^l\right) P\left(E^l_{\overrightarrow{C_t}(l)} \mid H_b^l\right) \quad (6)$$

Denklemde gösterildiği gibi, kategori vektörü $\overrightarrow{y_t}$ 'yi belirlemek için gereken tüm bilgiler, önceki olasılıklar $P(H_b^l)$ $(l \in \mathcal{y}, b \in \{0,1\})$ $(l \in Y, b \in \{0, 1\})$ ve son olasılıklar $P(E_j^l \mid H_b^l)$ $(j \in \{0, 1,..., K\})$. Aslında, bu önceki ve sonraki olasılıkların tümü, frekans sayımına dayalı eğitim setinden doğrudan tahmin edilebilir [27].

3.2.4 Destek Vektör Makineleri

Popüler sınıflandırma tekniklerinden birisi de destek vektör makineleridir (support vector machines). Destek vektör makineleri, kalkülüs, vektör geometrisi ve kısıtlı en iyileme gibi matematik konularına dayanan; doğrusal ve doğrusal olmayan sınıflandırma problemlerini gerçekleştirmeye olanak tanıyan bir makine öğrenmesi tekniğidir. İlk olarak AT&T laboratuvarlarında Vapnik ve arkadaşları tarafından geliştirilmiştir. Daha öncesinde ise 1963 yılında yine Vapnik tarafından geliştirilen bir algoritmaya dayanmaktadır. Destek vektör makineleri ile sınıflandırmanın temel amacı, iki boyutlu bir uzayda sınıflandırma yapmanın yanı sıra yüksek boyutlu öznitelikler uzayında da hiper düzlemlerin iyi bir şekilde ayrılarak en uygun sınıflandırmanın sağlanmasıdır [23].

Büyük ölçekli sınıflandırma problemlerini çözmek, metin sınıflandırması gibi birçok uygulamada çok önemlidir. Doğrusal sınıflandırma, çok sayıda örnek ve özelliğe sahip büyük seyrek veriler için en umut verici öğrenme tekniklerinden biri haline gelmiştir. Bu tür verilerle başa çıkmak için kullanımı kolay bir araç olarak Liblinear'ı geliştirilmiştir. L2-düzenlenmiş lojistik regresyon (LR), L2-kaybı ve L1-kaybı doğrusal destek vektör makinelerini (DVM'ler) destekler.

Destek Vektör Makinelerinin doğrusal, polinom, sigmoid ve radyal gibi çeşitleri vardır. Biz bu çeşitlerden doğrusal, polinom ve sigmoid DVM'leri kullandık.

Linear, iki popüler ikili doğrusal sınıflandırıcıyı destekler: LR ve doğrusal SVM.

Bir dizi örnek-etiket çifti verildiğinde (xi, yi), i = 1, . . ., ℓ; $x_i \in R^n$, $y_i \in \{-1, +1\}$, her iki yöntem de (Denk.7)'de kısıtlanmamış optimizasyon problemini farklı kayıp fonksiyonları ile çözer $\xi(w; x_i, y_i)$:

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w; xi, yi) \quad (7)$$

burada C> 0 bir ceza parametresidir.

SVM için, iki yaygın kayıp fonksiyonu max $(1- y_i w^T x_i, 0)$ ve $\max(1- y_i w^T x_i, 0)^2$. İlki L1-SVM olarak anılırken, ikincisi L2-SVM'dir. LR için kayıp fonksiyonu, olasılıksal bir modelden türetilen $\log (1 + e^{y^i w^T x_i})$ 'dir. Bazı durumlarda, sınıflandırıcının ayırt edici işlevi bir önyargı terimi içerir, b. Liblinear bu terimi, w vektörünü ve her x_i örneğini ek bir boyutla artırarak ele alır: $wT \leftarrow [w^T, b]$, $x_i^T \leftarrow [x_i^T, B]$, burada B, kullanıcı tarafından belirtilen bir sabittir. L1-SVM ve L2-SVM için yaklaşım, bir koordinat alçalma yöntemidir. LR ve ayrıca L2-SVM için, Liblinear bir güven bölgesi Newton yöntemi uygular. Test aşamasında, x veri noktasını $w^T x > 0$ ise pozitif, aksi halde negatif olarak tahmin ederiz [28].

3.2.5 Naif Bayes

Naif Bayes, Bayes'in kuralını ve sınıfa göre özniteliklerin koşullu olarak bağımsız olduğuna dair güçlü bir varsayımla birlikte kullanan basit bir öğrenme algoritmasıdır. Bu bağımsızlık varsayımı uygulamada sıklıkla ihlal edilse de, Naif Bayes yine de çoğu kez rekabetçi sınıflandırma doğruluğu sağlar. Hesaplama verimliliği ve diğer birçok arzu edilen özelliğiyle birleştiğinde bu, Naif Bayes'in pratikte geniş çapta uygulanmasına yol açar [23].

Naif Bayes, bir x nesnesi verilen her bir y sınıfının son olasılığını P (y | x) tahmin etmek için örnek verilerdeki bilgileri kullanmak için bir mekanizma sağlar. Bu tür tahminlere sahip olduğumuzda, bunları sınıflandırma veya diğer karar destek uygulamaları için kullanabiliriz.

Naif Bayes'in özellikleri aşağıdakileri içerir.

- İşlemsel verimlilik: Eğitim süresi, hem eğitim örneklerinin sayısı hem de özniteliklerin sayısı açısından doğrusaldır ve sınıflandırma süresi, özniteliklerin sayısına bağlı olarak doğrusaldır ve eğitim örneklerinin sayısından etkilenmez.
- Düşük varyans: Naif Bayes aramayı kullanmadığından, yüksek önyargı maliyetine rağmen düşük varyansa sahiptir.
- Artan öğrenme: Naif Bayes, eğitim verilerinden türetilen düşük dereceli olasılık tahminlerine göre çalışır. Bunlar, yeni eğitim verileri alındıkça kolayca güncellenebilir.
- Posterior olasılığının (meydana gelen bir olayın gözden geçirilmiş, güncellenmiş olasılığı) doğrudan tahmini.
- Gürültü karşısında sağlamlık: Naif Bayes her zaman tüm tahminler için tüm özellikleri kullanır ve bu nedenle sınıflandırılacak örneklerde gürültüye nispeten duyarsızdır. Olasılıkları kullandığı için eğitim verilerindeki gürültüye duyarsızlığını görmekteyiz.
- Kayıp değerler karşısında sağlamlık: Naif Bayes her zaman tüm tahminler için tüm öznitelikleri kullandığından, bir öznitelik değeri eksikse, diğer özniteliklerden gelen bilgiler yine de kullanılır ve bu da performansta zarif bir düşüşe neden olur. Olasılıklı çerçevesi nedeniyle eğitim verilerindeki eksik öznitelik değerlerine de oransal olarak duyarsızdır.

Naif Bayes, Bayes'in kuralına dayalı bir Bayes Ağ Sınıflandırıcısı biçimidir.

$$P(y \mid x) = P(y)\, P(x \mid y) / P(x) \quad (8)$$

Sınıfa verilen niteliklerin koşullu olarak bağımsız olduğu varsayımı ile birlikte. Öznitelik değeri verileri için bu varsayım (Denk.9) ve (Denk.10)'da,

$$P(x \mid y) = \prod_{i=1}^{n} P(x_i \mid y) \quad (9)$$

x_i: x'teki i.'inci özniteliğinin değeri

N: özniteliklerin sayısı

$$P(x) = \prod_{i=1}^{k} P(c_i)\, P(x \mid c_i) \quad (10)$$

k: sınıf sayısı

c_i, i.'inci sınıf

Böylece (Denk.8) denklemin sağ tarafındaki payları normalleştirerek hesaplanabilir.

Ortaya çıkan sınıflandırıcı, yalnızca parametrelerin seçilme biçiminde farklılık gösteren, lojistik regresyon tarafından kullanılana eşdeğer bir doğrusal model kullanır.

Kategorik özellikler için, gerekli olasılıklar P (y) ve P (x_i | y) normal olarak değerleri eğitim zamanında eğitim verilerinden tek bir geçişle hesaplanan dizilerde depolanan frekans sayımlarından türetilir. Bu diziler, yeni veriler elde edildikçe güncellenebilir ve artımlı öğrenmeyi destekler. Olasılık tahminleri genellikle Laplace tahmini veya en düşük tahmin gibi yumuşatma işlevleri kullanılarak sıklık sayımlarından türetilir.
Sayısal öznitelikler için, veriler ayrıklaştırılır (ayrıklaştırmaya bakın) veya olasılık yoğunluğu tahmini kullanılır.
Metin madenciliğinde, Naif Bayes'in iki çeşidi sıklıkla kullanılmaktadır. (McCallum vd. ,1998) (akt. Webb, 2017). Çok değişkenli Bernoulli modeli, yukarıda açıklandığı gibi Naif Bayes'i kullanır; her belge, her biri belirli bir kelimenin varlığını veya yokluğunu temsil eden ikili değişkenlerin bir vektörü olarak temsil edilir. Ancak, o belge için olasılıklar hesaplanırken yalnızca bir belgede bulunan sözcükler dikkate alınır.
Bunun aksine, çok terimli model, bir kelimenin bir belgede görünme sayısı hakkında bilgi kullanır. Bir belgedeki bir kelimenin her geçtiğini ayrı bir olay olarak ele alır. Bu olayların birbirinden bağımsız olduğu varsayılır. Dolayısıyla, bir sınıf verilen bir belgenin olasılığı, sınıfa verilen her kelime olayının olasılıklarının çarpımıdır [29].

3.2.6 Bayes Ağları

Bayes ağları, olasılıksal grafik modeller ailesine aittir. Grafik yapılar belirsiz bir alan hakkındaki bilgiyi temsil etmek için kullanılır. Özellikle, grafikteki her düğüm rastgele bir değişkeni temsil ederken, düğümler arasındaki kenarlar karşılık gelen rastgele değişkenler arasındaki olasılıksal bağımlılıkları temsil eder. Grafikteki bu koşullu bağımlılıklar genellikle bilinen istatistiksel ve hesaplama yöntemleri kullanılarak tahmin edilir. Bu nedenle, Bayes ağları grafik teorisi, olasılık teorisi, bilgisayar bilimi ve istatistikten ilkeleri birleştirir [32].

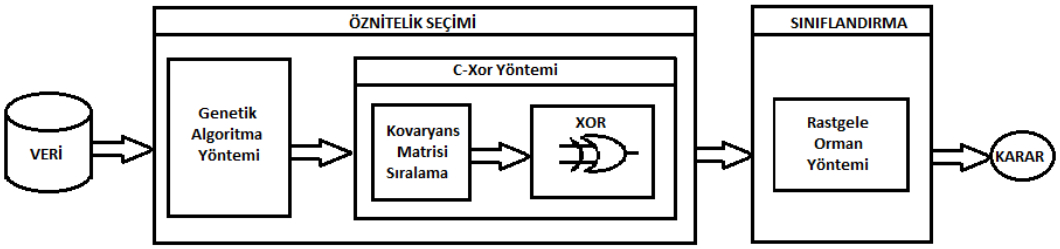
3.2.7 Torbalama

Tahmin edicileri torbalamak, bir tahmincinin birden çok sürümünü oluşturmak ve bunları toplu bir tahminciyi elde etmek için kullanmak için bir yöntemdir. Birleştirme, sayısal bir sonucu tahmin ederken versiyonların ortalamasını alır ve bir sınıfı tahmin ederken çoğul bir oylama yapar. Çoklu sürümler, öğrenme setinin önyükleme kopyaları yapılarak ve bunları yeni öğrenme setleri olarak kullanılarak oluşturulur. Sınıflandırma ve regresyon ağaçları kullanılarak gerçek ve simüle edilmiş veri

kümeleri üzerinde yapılan testler ve doğrusal regresyonda alt küme seçimi, torbalamanın doğrulukta önemli kazançlar sağlayabileceğini göstermektedir. Hayati unsur, tahmin yönteminin istikrarsızlığıdır. Öğrenme setini bozmak, oluşturulan tahmin aracında önemli değişikliklere neden olabilirse, torbalama doğruluğu artırabilir [33].

4. GELİŞTİRİLEN C-XOR ÖZNİTELİK SEÇİMİ YÖNTEMİ

Fidye yazılımlarının tespiti için geliştirmiş olduğumuz C-XOR makine öğrenmesi modeli Şekil 4.1.'de görülmektedir. Birinci aşamada, Genetik Algoritma öznitelik seçimi yöntemi ile düşürülmektedir. İkinci aşamada, C-XOR geliştirilmiş öznitelik seçim yönteminde kovaryans matrisi kullanarak en yüksek korelasyona sahip öznitelikler belirlendi. Üçüncü aşamada, bu yöntem içinde ikili en yüksek korelasyon sahip öznitelikler gruplandırılarak Xor kapısını uygulandı.



Şekil 4.1 Geliştirilen Makine Modeli.

Kovaryans, iki değişken arasındaki ilişkiyi ortaya koymak için kullanılır. Kovaryansın büyüklüğünden çok işareti önemlidir. Pozitif olması değişkenlerin doğru orantılı olduğunu, negatif olması değişkenlerin ters orantılı olduğunu, sıfır olması (veya sıfıra yakın olması) ise birbirleri arasında doğrusal bir bağ olmadığı anlamına gelmektedir. Değişkenler arasındaki kovaryanslar, kovaryans matris ile ifade edilir. Kovaryans formülü (Denk.11)'de görülmektedir.

$$COV_{(x,y)} = \frac{(\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}))}{n-1} \quad (11)$$

$COV_{(x,y)}$: x bağımsız değişkeni ile y bağımlı değişkeninin kovaryansı

x_i: x değişkeni

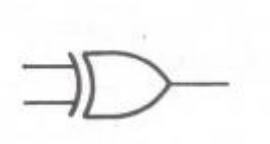
y_i: y değişkeni

$\bar{x}$: x değişkeninin ortalaması

$\bar{y}$: y değişkeninin ortalaması

n: toplam veri sayısı

Xor (özel-veya) mantık kapısının, Şekil 4.2'de sembolü ve doğruluk tablosu görülmektedir. Bu tabloyu incelediğimizde, sistemde girişler aynı olduğu zaman çıkışlar "0" değerini alır. Girişler farklı olduğunda ise çıkış "1" değerini almaktadır.



A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

(a) (b)

Şekil 4.2 Xor kapısının gösterimi(a) ve Doğruluk tablosu(b).

Xor kapısı formülü, (Denk.12)'de gösterilmiştir.

$$X = \overline{A}B + A\overline{B} \tag{12}$$

Bu özel kapılar giriş değerlerinin farklı olması esasına göre çalışır. Aynı olması durumda çıkışı sıfır lojiğe çeker. Bu durumu diğer mantık kapılarıyla tek başına yapmak mümkün değildir. Bu kapı bu çerçevede özel bir durum karşısında kullanılan kapılardandır. Bu özel durum girişlerin aynı olması mantığına dayanmıştır.

Bu formülü Python dilinde yazdığımız kod ile ikişerli gruplar halinde kovaryansı en yüksek olanları gruplayarak Xor uyguladık. Dolayısıyla her iki giriş için bir çıkış elemanı oluşturduk. Bu sayede 7977 Genetik Algoritmanın seçtiği özniteliği yarı yarıya 3988 sayısına düşürmüş olduk. Düşürdüğümüz bu yeni özellikleri yeniden makine öğrenmesine verdik. Çıkan sonuçlar Genetik Algoritma da bulduğumuz sonuçların belli bir puan üstünde olduğunu gördük.

Sınıflandırma kısmında ise fidye yazılımı veri seti üzerinde en yüksek başarımı veren Rastgele Orman algoritmasını kullandık.

5. BULGULAR

5.1 Veri seti

30.967 öznitelik ve 1.524 örnek Fidye yazılım verisi[2] üzerinde çalışma yapmış bulunmaktayız. Çizelge 5.1'de verdiğimiz bu veri kümesi, 582 fidye yazılımı örneğinin ve 942 iyi uygulamanın analizini içermektedir.

Çizelge 5.1'de Fidye yazılım örneklerinin bulunduğu veri setimizin, tüm öznitelikleri ikili veri (binary)'dir. Veri setinde yer alan "Family", fidye yazılımın hangi sınıfa (0-11) ait olduğunu gösteren bir etikettir.

Çizelge 5.1: Veri setinde kullanılan öznitelikler.

Adı	Açıklama	Sayısı
API	API çağrıları	232
DROP	Bırakılan dosyaların uzantıları	346
REG	Kayıt defteri anahtarı işlemleri	6622
FILES	Dosya işlemleri	4141
FILES_EXT	Dosya işlemlerinde yer alan dosyaların uzantısı	935
DIR	Dosya dizini işlemleri	2424
STR	Gömülü dizeler (strings)	16267

Veri setinde 1 adet iyi yazılım ve 11 çeşit fidye yazılım türü bulunmaktadır. Bunlar Goodware (0), Critroni (1), CryptLocker (2), CryptoWall (3), Kollah (4), Kovter (5), Locker (6), Matsnu (7), Pgpcoder (8), Reveton (9), TeslaCrypt (10), Trojan-Ransom (11) olarak ifade edilmiştir.

Veri setinde bulunan örneklerin sayısı toplamda 1524 ve dağılımı ise Goodware 942, Critroni 50, CryptLocker 107, CryptoWall 46, Kollah 25, Kovter 64, Locker 97, Matsnu 59, Pgpcoder 4, Reveton 90, TeslaCrypt 6, Trojan-Ransom 34 olarak görülmektedir.

[2] https://github.com/PSJoshi/Notes/wiki/datasets

5.2 Başarım Metrikleri

Fidye yazılımların tespiti için geliştirdiğimiz C-XOR öznitelik yöntemini sınıflandırıcılarla test ettik. Test başarım metriklerini Çizelge 5.2'de görülen çoklu sınıflandırıcı karmaşıklık matrisini kullanarak hesapladık.

Çizelge 5.2: Karmaşıklık matrisi.

		TAHMİN											
		CL	CW	K	M	L	C	H	R	TR	T	P	G
GERÇEK	Crypt Lock(CL)	T_{CLCL}	T_{CLCW}	T_{CLK}	T_{CLM}	T_{CLL}	T_{CLC}	T_{CLH}	T_{CLR}	T_{CLTR}	T_{CLT}	T_{CLP}	T_{CLG}
	Crypto Wall(CW)	T_{CWCL}	T_{CWCW}	T_{CWK}	T_{CWM}	T_{CWL}	T_{CWC}	T_{CWH}	T_{CWR}	T_{CWTR}	T_{CWT}	T_{CWP}	T_{CWG}
	Kovter(K)	T_{KCL}	T_{KCW}	T_{KK}	T_{KM}	T_{KL}	T_{KC}	T_{KH}	T_{KR}	T_{KTR}	T_{KT}	T_{KP}	T_{KG}
	Matsnu(M)	T_{MCL}	T_{MCW}	T_{MK}	T_{MM}	T_{ML}	T_{MC}	T_{MH}	T_{MR}	T_{MTR}	T_{MT}	T_{MP}	T_{MG}
	Locker(L)	T_{LCL}	T_{LCW}	T_{LK}	T_{LM}	T_{LL}	T_{LC}	T_{LH}	T_{LR}	T_{LTR}	T_{LT}	T_{LP}	T_{LG}
	Critroni(C)	T_{CCL}	T_{CCW}	T_{CK}	T_{CM}	T_{CL}	T_{CC}	T_{CH}	T_{CR}	T_{CTR}	T_{CT}	T_{CP}	T_{CG}
	Kollah(H)	T_{HCL}	T_{HCW}	T_{HK}	T_{HM}	T_{HL}	T_{HC}	T_{HH}	T_{HR}	T_{HTR}	T_{HT}	T_{HP}	T_{HG}
	Reveton(R)	T_{RCL}	T_{RCW}	T_{RK}	T_{RM}	T_{RL}	T_{RC}	T_{RH}	T_{RR}	T_{RTR}	T_{RT}	T_{RP}	T_{RG}
	Trojan-Ransom(TR)	T_{TRCL}	T_{TRCW}	T_{TRK}	T_{TRM}	T_{TRL}	T_{TRC}	T_{TRH}	T_{TRR}	T_{TRTR}	T_{TRT}	T_{TRP}	T_{TRG}
	TeslaCrypt(T)	T_{TCL}	T_{TCW}	T_{TK}	T_{TM}	T_{TL}	T_{TC}	T_{TH}	T_{TR}	T_{TTR}	T_{TT}	T_{TP}	T_{TG}
	Pgpcoder(P)	T_{PCL}	T_{PCW}	T_{PK}	T_{PM}	T_{PL}	T_{PC}	T_{PH}	T_{PR}	T_{PTR}	T_{PT}	T_{PP}	T_{PG}
	Goodware(G)	T_{GCL}	T_{GCW}	T_{GK}	T_{GM}	T_{GL}	T_{GC}	T_{GH}	T_{GR}	T_{GTR}	T_{GT}	T_{GP}	T_{GG}

Hata matrisi, bir sınıflandırıcının farklı sınıf etiketlerini ne ölçüde sınıflandırabildiğini gösteren bir analiz aracıdır. Bir veri setindeki sınıf etiketi sayısı n adet ise hata matrisi n x n boyutundan oluşan bir matris olarak düşünülebilir. Bizim bu çalışmamızın veri setine baktığımızda, 11 adet fidye yazılım ve 1 adet iyi yazılım ile toplamda 12 adet sınıf etiketi mevcuttur. Makine öğrenmesinde yaptığımız işlem çoklu sınıflandırmadır. Dolayısıyla bu veri seti için oluşturulacak hata matrisi 12 x 12 boyutunda bir matris olacaktır.

Başarım metrikleri hakkında kısa tanımlar yaparsak;

- Doğru Pozitif (DP): Sınıflandırıcı tarafından doğru olarak etiketlendirilmiş (tahmin edilmiş) pozitif sınıf etiketleridir. Eğer yazılım gerçekten iyi bir yazılım ve kurulan sınıflandırma modelinin tahmin ettiği sonuçta iyi bir yazılım ise bu durum doğru pozitif ile ifade edilmektedir.

- Doğru Negatif (DN): Sınıflandırıcı tarafından doğru olarak etiketlendirilmiş negatif sınıf etiketleridir. Eğer yazılım gerçekten iyi bir yazılım değil ve kurulan sınıflandırma modelinin tahmin ettiği sonuçta iyi bir yazılım olmadığını gösteriyor ise bu durum doğru negatif ile ifade edilmektedir.
- Yanlış Negatif (YN): Sınıflandırıcı tarafından yanlış olarak etiketlendirilmiş negatif sınıf etiketleridir. Eğer bir yazılım gerçekten iyi bir yazılım ve kurulan sınıflandırma modelinin tahmin ettiği sonuçta iyi bir yazılım değil ise bu durum yanlış negatif ile ifade edilmektedir.
- Yanlış Pozitif (YP): Sınıflandırıcı tarafından yanlış olarak etiketlendirilmiş pozitif sınıf etiketleridir. Eğer bir yazılım gerçekten iyi bir yazılım değil ve kurulan sınıflandırma modelinin tahmin ettiği sonuçta iyi bir yazılım ise bu durum yanlış pozitif ile ifade edilmektedir.

Doğruluk (accuracy) ölçütü (Denk.13)'de görüleceği gibi, doğru olarak sınıflandırılmış örnek sayısının tüm örnek sayısına oranı olarak ifade edilebilir.

$$\text{Doğruluk} = (DP + DN) / (DP + DN + YP + YN) \quad (13)$$

Doğruluk ölçütü, performans ölçülürken veri setinin dengeli dağılması önemli bir kriterdir. Bu veri setimizde bazı örnekler çok yüksek olurken bazı örneklerin az olduğunu görüyoruz. Örneğin, 8 numaralı sınıftan 4 ve 10 numaralı sınıftan 6 örnek varken, 0 nolu sınıftan 942 örnek bulunmaktadır. Az örnek olmasından dolayı bu örnekleri sınıflandıramamaktadır. Buda doğruluk oranı düşürmektedir.

Fleiss'in kappa katsayısı, ikiden fazla sabit sayıda değerleyici arasındaki karşılaştırmalı uyuşmanın güvenirliğini ölçen bir istatistik yöntemidir. Fleiss'in kappa ölçüsü sabit sayıda (n tane) değerleyicinin her birinin, (N tane) maddeyi veya kişiyi (C tane) birbirinden karşılıklı hariç olan kategoriye göre ayırmaları süreci sonunda ortaya çıkan değerleyiciler arasındaki uyuşmayı ölçer. Fleiss'in kappa ölçüsü bu uyuşmanın bir şans eseri olabileceğini de ele aldığı için basit yüzde orantı olarak bulunan uyuşmadan daha güçlü bir sonuç verdiği kabul edilir. Ortaya çıkan kategorik değişken olduğu için Fleiss'in kappa katsayısı bir parametrik olmayan istatistik türüdür [31].

Kesinlik (Precision) ölçütü (Denk.14)'de görüldüğü gibi, pozitif olarak sınıflandırılmış tüm örnekler arasında gerçekten kaç tanesinin doğru sınıflandırıldığını göstermektedir.

$$\text{Kesinlik} = DP / (DP + YP) \quad (14)$$

Doğru olarak sınıflandırılan fidye yazılımların sayısını bulmak için kesinlik ölçütünü kullanmak daha doğru bir yaklaşımdır.

Yüksek duyarlığı (recall, sensivity) olan sınıflandırıcılarda yanlış olarak sınıflandırılmış, negatif örnek sayısının az olması beklenir. Bu veri setinde oranı yükseldiğinde en büyük sorun teşkil edecek grup, şüphesiz yanlış negatif grubudur. Çünkü bir yazılım fidye yazılım olduğu halde o yazılıma iyi bir yazılım olduğunu işaretlemek çok kötü sonuçlar ortaya çıkarabilir. Hata matrisi kullanarak hesaplanan duyarlılığın oranı (Denk.15)'de ifade edilmiştir.

Duyarlık = DP / (DP + YN) (15)

Veri setinin eğitim seti ve test seti olarak ayrılması esnasında, veri setinin dağılımında oluşabilecek düzensizlikler, makine öğrenmesi modelinin performansını olumsuz etkileyebilir. Bu problem k-katlı çapraz doğrulama (k-fold cross validation) yöntemi ile çözüme kavuşturulabilir. Çapraz doğrulamada, veri seti k olarak ifade edilen sayı kadar parçaya ayrılır. Daha sonra her adımda k-1 adet parça makine öğrenmesi algoritmasında eğitilir ve kalan parça üzerinde test edilir. Burada önemli olan nokta, her adımda daha önce denenmemiş parçanın test seti olarak kullanılmasıdır. Son olarak da her adım neticesinde elde edilen hata değerlerinin ortalaması alınarak, modelin toplam hatası elde edilmiş olur [23].

Biz bu çalışmamızda çapraz doğrulamada k=10 sayısını aldık. Her adımda farklı bir parçanın test seti olarak kullanıldığı ve (Denk.16)'de görüldüğü üzere, elde edilen hataların (H_1,H_2...H_{10}) ortalaması toplam hatayı (H) görüldüğü üzere performansı verilmiştir.

$$H = {}^{1}/_{n} \sum_{i=1}^{n} H_i \quad (16)$$

5.3 Deneysel Sonuçlar

Kurduğumuz hipotezimize göre Genetik Algoritma da elde ettiğimiz öznitelikleri kendi yöntemimizle azaltmaya, bu yolla hem çalışma hızını artırmaya hemde performans – doğruluk oranını artırmaya çalıştık.

- Birinci adım; veri setimizin öznitelik sayısı (f) genetik algoritma yöntemiyle 7.977 özniteliğe (k) düşürülmüştür.

- İkinci adım; elde edilen k adet özniteliklerin kovaryansı hesaplanarak, k x k boyutunda kovaryans matrisi hesaplanmıştır.
- Üçüncü adım; Kovaryans matris kullanılarak birbirine en çok benzeyen ikili değişken grupları belirlendi.
- Dördüncü adım; üçüncü adımda belirlenen özniteliklere Xor kapısı uygulandı. Böylece k öznitelik sayısı k/2'ye düşürüldü.
- Beşinci adım; Elde edilen k/2 adet öznitelik içeren veri seti sınıflandırma algoritmalarıyla başarım testlerine tabi tutulmuştur.

Fidye yazılım veri seti üzerinde Naif Bayes, BayesNet, k-EYK, ÇKAA, Rastgele Orman, DVM, Torbalama sınıflandırıcı algoritmaları test ettik. Testler sonucunda sınıf doğruluğu, Kappa, ağırlıklandırılmış kesinlik ve ağırlıklandırılmış duyarlık başarım metriklerine göre sonuçlar elde ettik. Çizelge 5.3'de veri setine sadece dengeleme uyguladıktan sonra elde ettiğimiz deneysel sonuçlar görülmektedir. Buna göre k-EYK % 55,06 ile en yüksek sonucu vermektedir. Kappa değeri 0,509, kesinlik 0,5321, duyarlık ise 0,5309 olarak görülmektedir.

Çizelge 5.3: Tüm veri setinde çoklu sınıflandırma başarımları.

Algoritmalar	Doğruluk (%)	Kappa	Kesinlik	Duyarlık
Naif Bayes	42,87	0,376	0,4826	0,4115
BayesNet	N/A	N/A	N/A	N/A
k-EYK	55,06	0,509	0,5321	0,5309
ÇKAA	36,85	0,309	0,4038	0,3535
Doğrusal DVM	36,63	0,308	0,4255	0,364
Polinom DVM	19,85	0,125	0,3143	0,1974
Sigmoid DVM	36,76	0,31	0,423	0,3658
Torbalama	54,95	0,507	0,54	0,50
Rastgele Orman	9,11	0,008	0,0571	0,0901

Çizelge 5.3 incelendiğinde en yüksek başarı k-EYK yöntemi % 55,06 sınıf doğruluğu, 0,509 Kappa, 0,5321 ağırlıklı kesinlik ve 0,5309 ağırlıklı duyarlık sonucu elde edilmiştir.

Bu veri seti öznitelik seçimi için Genetik Algoritma, parçacık sürü optimizasyonu ve Doğrusal İleri Seçim yöntemi uygulanmıştır. Parçacık Sürü Optimizasyonu çizelge 5.4'de, Genetik Algoritma çizelge 5.5'da, Doğrusal İleri Seçim yöntemi

çizelge 5.6'de gösterilmiştir. Bu çizelgeler incelendiği zaman en iyi performansı % 57,79 k-EYK Algoritması ile Genetik Algoritma yönteminin verdiğini görmekteyiz. Kappa değeri 0,54, ağırlıklandırılmış kesinlik 0,54 ve ağırlıklandırılmış duyarlık 0,58 olarak görülmektedir. Bu değerlerden dolayı bir sonraki aşamaya Genetik Algoritmayla devam edilmiştir.

Çizelge 5.4: Parçacık Sürü Eniyileme 3370 öznitelikle çoklu sınıflandırma.

Algoritmalar	Doğruluk (%)	Kappa	Kesinlik	Duyarlık
Naif Bayes	36,29	0,31	0,41	0,36
BayesNet	38,51	0,33	0,41	0,38
k-EYK	48,68	0,44	0,49	0,49
ÇKAA	8,41	0,00	0,00	0,08
Doğrusal DVM	N/A	N/A	N/A	N/A
Polinom DVM	N/A	N/A	N/A	N/A
Sigmoid DVM	N/A	N/A	N/A	N/A
Torbalama	47,70	0,43	0,47	0,48
Rastgele Orman	49,75	0,45	0,49	0,50

Çizelge 5.5: Genetik Algoritma 7977 öznitelikle çoklu sınıflandırma

Algoritmalar	Doğruluk (%)	Kappa	Kesinlik	Duyarlık
Naif Bayes	N/A	N/A	N/A	N/A
BayesNet	43,03	0,38	0,47	0,43
k-EYK	57,79	0,54	0,54	0,58
ÇKAA	22,71	0,16	25,9	22,19
Doğrusal DVM	36,72	0,31	40,93	36,52
Polinom DVM	25,39	0,19	37,24	25,21
Sigmoid DVM	36,72	0,31	40,92	36,52
Torbalama	58,86	0,55	0,56	0,59
Rastgele Orman	58,31	0,54	N/A	0,58

Çizelge 5.6: Doğrusal İleri Seçimi Yöntemi 57 öznitelikle çoklu sınıflandırma.

Algoritmalar	Doğruluk (%)	Kappa	Kesinlik	Duyarlık
Naif Bayes	46,01	0,41	0,46	0,46
BayesNet	46,23	0,41	0,46	0,46
k-EYK	47,36	0,42	0,48	0,47
ÇKAA	10,42	0,02	0,04	0,10
Doğrusal DVM	N/A	N/A	N/A	N/A
Polinom DVM	N/A	N/A	N/A	N/A
Sigmoid DVM	N/A	N/A	N/A	N/A
Torbalama	48,39	0,44	0,49	0,48
Rastgele Orman	48,24	0,44	0,49	0,48

Çizelge 5.7'de geliştirilmiş C-XOR öznitelik seçimi yöntemi ile elde ettiğimiz yeni öznitelikleri makine öğrenmesine verdiğimizde elde ettiğimiz sonuçları görmektesiniz.

Çizelge 5.7: C-XOR öznitelik seçimi yöntemi 3988 öznitelikle çoklu sınıflandırma.

Algoritmalar	Doğruluk (%)	Kappa	Kesinlik	Duyarlık
Naif Bayes	42,96	0,38	0,45	0,43
BayesNet	44,05	0,39	0,46	0,44
k-EYK	58,38	0,55	0,62	0,58
ÇKAA	32,25	0,26	34,93	31,98
Doğrusal DVM	36,72	0,31	36,87	36,46
Polinom DVM	31,41	0,25	43,69	31,27
Sigmoid DVM	36,72	0,31	36,87	36,46
Torbalama	59,14	0,55	0,62	0,59
Rastgele Orman	59,15	0,55	0,63	0,59

Çizelge 5.7'e göre en yüksek doğruluğu % 59,15, Kappa 0,55, ağırlıklandırılmış kesinlik 0,63 ve ağırlıklandırılmış duyarlık 0,59 ile Rastgele Orman algoritması vermiştir.

6. SONUÇLAR

Tez çalışmasında, fidye yazılımları tespit edebilmek için C-XOR adını verdiğimiz bir öznitelik seçimi yöntemi geliştirdik. C-XOR yöntemi Fidye yazılımı veri seti üzerinde çeşitli makine öğrenmesi sınıflandırıcı algoritmalarıyla test ettik. C-XOR yöntemi üç aşamadan oluşmaktadır. İlk aşamada, Genetik Algoritma yöntemi ile veri setindeki öznitelikler seçilmektedir. İkinci aşamada ise, ilk aşamada elde edilen veri setine göre kovaryans matris hesaplanmaktadır. Bu matris ile her bir özniteliğin en yüksek korelasyona sahip olduğu öznitelik belirlenmektedir. Bu işlem için kovaryans matrisde her bir öznitelik birbirleriyle ikili olarak kıyaslanmakta ve en yüksek korelasyon değeri elde edilmektedir. Üçüncü aşamada, en yüksek korelasyona sahip özniteliklere XOR(özel veya) mantığı uygulanmaktadır. Böylece özniteliklerin sayısı yarıya düşürülür. Fidye yazılım veri seti üzerinde C-XOR öznitelik seçimi yöntemimizi Naif Bayes, BayesNet, k-EYK, ÇKAA, DVM, Torbalama ve Rastgele Orman algoritmalarında test ettik. En yüksek başarımı Rastgele Orman algoritması yöntemi verdi.

Gelecek de yapacağımız çalışmalarda; ilk olarak, C-Xor öznitelik seçimi yöntemini birleştirilmiş sınıflandırıcılarla başarımı daha da artırmak için test edeceğiz. İkinci olarak, C-XOR yöntemini başka ikili veri setleri üzerinde test edeceğiz. Üçüncü olarak, C-XOR yöntemini geliştireceğimiz bir çevrimiçi sunucu ile yapay zeka eko-sisteminde kullanıcıların kullanımına açmayı planlamaktayız. Son olarak, C-XOR yöntemini yaygın olarak kullanılan veri madenciliği uygulamaları (weka, orange vb.) için eklentiler geliştirmeyi planlamaktayız.

KAYNAKÇA

[1] Sgandurra, D., Muñoz-González, L., Mohsen, R., & Lupu, E. C. (2016). Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020.

[2] Khan, F., Ncube, C., Ramasamy, L. K., Kadry, S., & Nam, Y. (2020). A digital DNA sequencing engine for ransomware detection using machine learning. IEEE Access, 8, 119710-119719.

[3] Lee, S., & Hwang, J. (2018). A study on variable selection and classification in dynamic analysis veri for ransomware detection. The Korean Journal of Applied Statistics, 31(4), 497-505.

[4] Adamu, U., & Awan, I. (2019, August). Ransomware prediction using supervised learning algorithms. In 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 57-63). IEEE.

[5] Takeuchi, Y., Sakai, K., & Fukumoto, S. (2018, August). Detecting ransomware using support vector machines. In Proceedings of the 47th International Conference on Parallel Processing Companion (pp. 1-6).

[6] Poudyal, S., Subedi, K. P., & Dasgupta, D. (2018, November). A framework for analyzing ransomware using machine learning. In 2018 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1692-1699). IEEE.

[7] Ahmed, Y. A., Kocer, B., & Al-rimy, B. A. S. (2020). Automated Analysis Approach for the Detection of High Survivable Ransomware. KSII Transactions on Internet and Information Systems (TIIS), 14(5), 2236-2257

[8] Kılıç, Ç., Dünden bugüne fidye yazılımların (Ransomware) gelişimi ve geleceği. 2019, İstanbul Bilgi Üniversitesi.

[9] Tokmak, M. and E.U. Küçüksille, Kötü Amaçlı Windows Çalıştırılabilir Dosyalarının Derin Öğrenme İle Tespiti. Bilge International Journal of Science and Technology Research. 3(1): p. 67-76.

[10] İlker, K., Teslacrypt fidye yazilim virüsünün tespiti, teknik analizi ve çözümü. Uluslararası Yönetim Bilişim Sistemleri ve Bilgisayar Bilimleri Dergisi. 2(2): p. 87-94.

[11] Nieuwenhuizen, D., A behavioural-based approach to ransomware detection. Whitepaper. MWR Labs Whitepaper, 2017.

[12] Dey, D. and A. Lahiri. Should We Outlaw Ransomware Payments? in Proceedings of the 54th Hawaii International Conference on System Sciences. 2021.

[13] Mohammad, A.H., Ransomware evolution, growth and recommendation for detection. Modern Appl. Sci, 2020. 14(3).

[14] Eken, H. (2013).Mobil ve Web Uygulamalarının Yazılım Güvenliği.

[15] Canbek, G., & Sağıroğlu, Ş. (2007). Kötücül ve Casus Yazılımlar: Kapsamlı Bir Araştırma. Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 22(1), 121-136.

[16] Çelik, S., & Çeliktaş, B. (2017). Güncel Siber Güvenlik Tehditleri: Fidye Yazılımlar. Cyberpolitik Journal, 2(4), 296-323.

[17] Siber Güvenliğe İlişkin Temel Bilgiler Ulusal Siber Olaylara Müdahale Merkezi (USOM-TRCERT)Bilgi Teknolojileri ve İletişim Kurumu Telekomünikasyon İletişim Başkanlığı.

[18] Bakır, E. (2011). İnternet Güvenliğinin Tarihçesi. TÜBİTAK Bilgem.

[19] Bilgisayar Virüsü, https://www.kaspersky.com.tr/resource-center/threats.(01.04.2021).

[20] Canbek, G., & Sağiroğlu, Ş. (2008). Casus Yazılımlar: Bulaşma Yöntemleri Ve Önlemler. Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 23(1).

[21] ESET Sözlüğü, https://help.eset.com/glossary/tr-TR/adware.html.(01.04.2021).

[22] Kılıç, N. (2018). İnternet ortamında kişilik haklarına saldırıdan doğan hukuki sorumluluk (Master's thesis, Çankaya Üniversitesi).

[23] Uğuz, S. (2019). Makine Öğrenmesi – Teorik yönleri ve Python uygulamaları ile bir yapay zeka ekolü, Nobel Akademi Yayıncılık.

[24] He, H., & Garcia, E. A. (2009). Learning from imbalanced veri. IEEE Transactions on knowledge and veri engineering, 21(9), 1263-1284.

[25] Karaboğa, D. (2017). Yapay zeka optimizasyon algoritmalari. Nobel Akademi Yayıncılık.

[26] Zheng, Z., & Webb, G. I. (2000). Lazy learning of Bayesian rules. Machine Learning, 41(1), 53-84.

[27] Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. Pattern recognition, 40(7), 2038-2048.

[28] Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). Liblinear: A library for large linear classification. the Journal of machine Learning research, 9, 1871-1874.

[29] Sammut, C., & Webb, G. I. (2017). Encyclopedia of machine learning and veri mining. Springer.

[30] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[31] Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. Psychological bulletin, 76(5), 378.

[32] Ruggeri, F., Faltin, F., & Kenett, R. (2007). Bayesian networks. Encyclopedia of Statistics in quality and reliability.

[33] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.