T.C.
HALİÇ ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE WEB İSTEKLERİNDE ANOMALİ TESPİTİ

YÜKSEK LİSANS TEZİ

Çağlar ABABAY

Tez Danışmanı: Dr. Öğr. Üye. Figen ÖZEN

Ocak 2022

T.C.
HALİÇ ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE WEB İSTEKLERİNDE ANOMALİ TESPİTİ

YÜKSEK LİSANS TEZİ

Hazırlayan
Çağlar ABABAY

Danışman
Dr. Öğr. Üyesi Figen ÖZEN

Ocak 2022

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ'NE

Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Programı Öğrencisi Çağlar Ababay tarafından hazırlanan ***"Makine Öğrenmesi Yöntemleriyle Web İsteklerinde Anomali Tespiti"*** konulu çalışması jürimizce Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 11/01/2022

Jüri Üyesinin Ünvanı, Adı, Soyadı ve Kurumu: İmzası

Jüri Üyesi : Dr. Öğr. Üye. Figen Özen
Haliç Üniversitesi

Jüri Üyesi : Prof. Dr. B. Koray Tunçalp
Haliç Üniversitesi

Jüri Üyesi : Dr. Öğr. Üye. Zeynep Turgut
İstanbul Medeniyet Üniversitesi

Bu tez yukarıdaki jüri üyeleri tarafından uygun görülmüş ve Enstitü Yönetim Kurulu'nun kararıyla kabul edilmiştir.

(Enstitü Müdürünün Ünvanı, Adı, Soyadı)

Çağlar Ababay

ORİJİNALLİK RAPORU

%7	%7	%0	%2
BENZERLİK ENDEKSİ	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCİ ÖDEVLERİ

BİRİNCİL KAYNAKLAR

1	harun-demir.com İnternet Kaynağı	%3
2	medium.com İnternet Kaynağı	%1
3	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Öğrenci Ödevi	%1
4	Submitted to Kastamonu University Öğrenci Ödevi	<%1
5	ichi.pro İnternet Kaynağı	<%1
6	dspace.balikesir.edu.tr İnternet Kaynağı	<%1
7	globalnotlar.com İnternet Kaynağı	<%1
8	www.openaccess.hacettepe.edu.tr:8080 İnternet Kaynağı	<%1
9	www.researchsquare.com İnternet Kaynağı	<%1

…./…./2022

TEZ ETİK BEYANI

Yüksek Lisans Tezi olarak sunduğum "Makine Öğrenmesi Yöntemleri İle Web İsteklerinde Anomali Tespiti" başlıklı bu çalışmayı baştan sona kadar danışmanım Dr. Öğr. Üy. Figen Özen'in sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Çağlar ABABAY

ÖNSÖZ

İnsan hayatında internet her geçen gün önemini artırmaktadır. Birçok konuda hayatı kolaylaştırdığı yadsınamaz bir gerçektir. Bankacılık işlemleri, kamu kuruluşlarının dijital hizmetleri, internet üzerinden yapılan alışveriş, sosyal medya kullanımı gibi birçok farklı mecra üzerinden hayatımızın vazgeçilmez bir ihtiyacı haline gelmiştir.

İnternet kullanımım bu denli büyümesiyle beraber kişileri ve kuruluşları zarara uğratabilecek riskler de artmıştır. Bu risklerin temelinde veri sızıntıları, maddi kayıplar, marka itibarlarının zedelenmesi yer almaktadır.

Bu çalışmada ise web sitelerine yapılan anormal isteklerin tespiti, farklı makine öğrenmesi metodlarıyla uygulanmıştır. Anormal isteklerin tespiti ile web sitelerine yapılan saldırılar, veri hırsızlığı gibi durumların hızlıca önüne geçilmesi mümkün olacaktır.

Yüksek lisans öğrenimim boyunca bana birçok konuda destek olan çok değerli eşime, hayattaki en büyük motivasyon kaynağım olan biricik kızıma, tezimde benzersiz bir veri seti kullanmam konusunda izin ve desteklerini esirgemeyen Enes GÜREN'e, derslerime giren değerli hocalarıma ve tezim konusunda desteğini esirgemeyen çok değerli hocam Figen ÖZEN'e sonsuz teşekkür ediyorum.

Ocak 2022 Çağlar ABABAY

İÇİNDEKİLER

Sayfa No

KISALTMALAR

YSA : Yapay sinir ağı
DVM : Destek vektör makinesi
RBF : Radial Basis Function
TP : True positive
FP : False positive
TN : True negative
FN : False negative
DoS : Denial of service
VANET : Vehicular ad-hoc network
ITS : Intelligent transportation system

ÇİZELGE LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

ÖZET

MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE WEB İSTEKLERİNDE ANOMALİ TESPİTİ

Bir websitesinin en önemli önceliklerinden biri kesintisiz olarak hizmet sağlamasıdır. Yoğun kullanıcı trafiğinin yanı sıra kötü niyetli saldırılar da sistem kaynaklarının yetersiz kalmasına ve hizmetin kesilmesine sebep olabilmektedir. Hizmetin kesilmesi, web sitesi için hem maddi kayıp hem de itibar kaybına neden olacaktır. Bunun yanı sıra insanlar üzerinde psikolojik etkileri dahi olduğu düşünülmektedir. Günümüzde, ülkemizde ve dünyada bu tarz kesintiler ve etkileriyle ilgili haberlere bolca örnek verilebilir. Bundan dolayı anomali tespiti ile ilgili geliştirmelere büyük bir ilgi vardır. Bu tarz hizmet kesintileri büyük şirketler için milyarlarca dolar kaybı, küçük şirketler için belki de piyasadan silinme tehlikesi içerir. Ek olarak; çok yoğun trafik alan herhangi bir web sitesi (sosyal medya devleri) kesintiye uğradığında, bu site için normal olan trafiğin büyük bir kısmı başka bir web sitesine yığılmakta ve bu da bir anomali oluşturmaktadır. Bu durum da, aslında bir saldırı olmayıp, dengelerin bozulması kaynaklı bir anomali oluşturur. Dengeler bozulduğunda, anomalilerin saptanması ve önlem alınması da zorlaşacaktır. Bu sebeplerle, bu çalışmanın günümüzdeki internet kullanımının boyutu ve önemi düşünüldüğünde, tüm alanlarda büyük yarar sağlayabileceği aşikardır.

Bu tezin amacı, kullanıcı trafiğindeki anormal olan isteklerin tespit edilmesi ve hızlı bir şekilde engellenmesini sağlamaktır. Bunun temelinde, normal olan web istekleriyle, anormal olan web isteklerinin ayrıştırılması (isteklerin sınıflandırılması) yer almaktadır. Anormallik durumu belirli bir zaman aralığında (son 1 saat, son 6 saat gibi) yapılan isteklerin tümüne bakılarak hesaplanmıştır. Anlık ya da çok kısa sürede tespit edilen anormal trafik engellenerek, web sitesinin hizmet devamlılığı ya da çok kısa süreli kesinti sağlanabilecektir.

Popüler makine öğrenmesi yöntemleri (yapay sinir ağı, desteklenmiş karar ağacı, Naive Bayes sınıflandırıcı, lojistik regresyon sınıflandırıcı, destek vektör makinesi, otomatik kodlayıcı) kullanılmuş ve karşılaştırmaları yapılmıştır. Çalışmanın temelinde, belirli bir zaman aralığındaki ziyaretlerin, belirli metriklerden faydalanarak, sınıflandırılmasını sağlamak yatmaktadır. Örneğin, ortalama bir isteğin işlem süresi 100 milisaniye ise, 500 milisaniyelik bir istek anomali oluşturabilir. Bu ve buna benzer birçok metrik kullanılarak, istek yapan IP adresinin normal davranıştan vektörel olarak ne kadar uzakta olduğu hesaplanacak ve eşik düzey aşılmışsa bunun bir anomali olduğuna karar verilmiştir. Yapılan karşılaştırmalar, doğruluk yüzdeleri, F1 skorları ve çalışma zamanı üzerinden olmuştur.

Gerçek bir websitesine ait, IP adresi bazında isteklerden oluşan, belirli bir tarih aralığındaki oluşturulan veri üzerinde denemeler yapılmıştır. Ham olan bu veri öncelikle işlenecek ve makine öğrenmesi metodlarından birine başlamadan önce hazır hale getirilmiştir. Ham veri tarih bazlı ve istek özelinde satırlardan oluşmaktadır.

Hazırlanmış veri ise IP adresi bazlı olup, o IP adresinin belirli zaman aralığında; saniyede kaç istekte

bulunduğu, isteklerin ortalama kaç milisaniye sürdüğü, isteklerin boyutu gibi parametrelerden oluşmaktadır. Ayrıca bu veri gerçek bir saldırı durumu da içermektedir ve bunun hangi IP adresleri tarafından yapıldığı bilinmektedir. Yapılacak çalışmanın denemelerinde, gerçek anomali durumları, sonucun doğruluğunu güçlendirmiştir. Aynı zamanda makine öğrenmesine de destek olmuştur.

Veri üzerinde çalışmak üzere en uygun programa dillerinden biri olan Python ile geliştirmeler yapılmıştır. Python programlama dilinin en güncel ve stabil versiyonlarından biri olan 3.7 versiyonu kullanılmıştır. Ayrıca veri işleme, makine öğrenmesi metodlarının algoritmaları ve veri görselleştirme işlemleri için dünyada en çok kullanılan "numpy", "sklearn" ve "tensorflow" kütüphanelerinden yararlanılmıştır. Her bir makine öğrenmesi, metodu aynı veri ile çalışan, ayrı birer proje şeklinde hazırlanacaktır. Böylece metodların birbiriyle kıyaslanması, verimliliğin daha doğru bir şekilde ölçülmesi hedeflenmiştir.

Bu çalışmanın sonucunda, birçok web sitesinin ve sağlayıcılarının kullanabileceği, web sitesinin ve trafiğin türünden bağımsız olarak çalışabilen bir uygulama oluşturulmaktadır. Bu sayede web siteleri maddi kayıplardan ve itibar kaybından korunarak, müşterilerine kesintisiz hizmet sağlayabileceklerdir.

Anahtar Kelimeler: *Anomali Tespiti, Sınıflandırma, Web Saldırıları, Makine Öğrenmesi*

ABSTRACT

ANOMALY DETECTION IN WEB REQUESTS USING MACHINE LEARNING METHODS

One of the most important priorities of a website is to provide uninterrupted service. In addition to heavy user traffic, malicious attacks can also cause insufficient system resources and interruption of service. Interruption of the service will cause both financial loss and loss of reputation for the website. In addition, it is thought to have psychological effects on people. Today, there are plenty of examples of news about such cuts and their effects in our country and in the world. Therefore, there is a great interest in the development of anomaly detection. Such service interruptions entail the loss of billions of dollars for large companies and the danger of being wiped out of the market for small companies. In addition; When any very heavy traffic website (social media giants) is interrupted, most of the traffic that is normal for that site is piled up on another website, which creates an anomaly. This situation, in fact, is not an attack, but creates an anomaly caused by the disruption of balances. When the balances are disturbed, it will be difficult to detect anomalies and take precautions.

The aim of this thesis is to detect abnormal requests in user traffic and to prevent them quickly. The basis for this is the separation (classification of requests) of normal web requests and abnormal web requests. Anomaly status was calculated by looking at all requests made in a certain time interval (such as the last 1 hour, the last 6 hours). Abnormal traffic detected instantly or in a very short time will be blocked, and the continuity of service of the website or a very short interruption will be ensured.

Popular machine learning methods (artificial neural network, supported decision tree, Naive Bayes classifier, logistic regression classifier, support vector machine, autoencoder) were used and compared. The basis of the study is to classify the visits in a certain time period by using certain metrics. For example, if the processing time of an average request is 100 milliseconds, a 500 millisecond request may generate an anomaly. By using this and many similar metrics, it will be calculated how far the requesting IP address is vectorally from the normal behavior, and if the threshold level is exceeded, it is decided that this is an anomaly. Comparisons were made on percentages of accuracy, F1 scores, and runtime.

Experiments have been made on the data of a real website, consisting of requests on the basis of IP addresses, in a certain date range. This raw data will be processed first and made ready before starting one of the machine learning methods. Raw data consists of date-based and request-specific lines. Prepared data, on the other hand, is based on IP address, within a certain time interval of that IP address; It consists of parameters such as how many requests per second, how many milliseconds the requests take on average, and the size of the requests. In addition, this data includes a real attack situation and it is known by which IP addresses it was made. In the trials of the study, the real anomaly cases strengthened the accuracy of the result. It also supported machine learning.

Developed with Python, one of the most suitable programming languages to work on data. One of the most up-to-date and stable versions of the Python programming language, version 3.7, was used. In addition, the world's most widely used "numpy", "sklearn" and "tensorflow" libraries were used for data processing, algorithms of machine learning methods and data visualization. Each machine learning method will be prepared as a separate project that works with the same data. Thus, it is aimed to compare the methods with each other and to measure the efficiency more accurately.

As a result of this study, an application is created that can be used by many websites and their providers, and that can work regardless of the type of website and traffic. In this way, websites will be protected from financial losses and loss of reputation, and they will be able to provide uninterrupted service to their customers.

Although open source datasets used in similar studies are ideal for simulation environment, it is clear that they cannot produce accurate outputs in practice. When a study conducted in 2019 is examined, there are 67,343 normal requests and 45,927 labeled request data in attack types in the training dataset. Less class imbalance facilitates the solution of the problem. However, what is expected in real life is that anomaly situations are much less than normal ones. In line with this expectation, situations where the class imbalance is much higher should also be evaluated during the modeling phase. In the aforementioned research, the accuracy value of the experiment with the support vector machine is 99%. In this study, the result of the experiment with the support vector machine is 97%. However, as mentioned before, the user access records of a real website were used in this study, the class imbalance in the data is quite high.

In another study, which includes an open source user request log data with proper class balance, the decision tree method was used as in this study. The accuracy score in the experiment was found to be 97.7%. In this study, the decision tree method was strengthened with the "AdaBoost" method, and the accuracy value was found to be 99.9% even though the class imbalance was very high.

NSL-KDD dataset with open source access was used in all of the literature researches. In addition, while preparing the training data in most of the studies, there is some normal request and the same amount of data labeled as attack, so as to create the class balance. Each condition that qualifies as an anomaly should be less numerous than the normal condition. On the contrary, it even contradicts the meaning of the word. In this way, since all the conditions in the simulation environment are in the most ideal situation, the results will be more successful and predictable. In fact, the simulation environment is expected to be very similar to the real environment.

There are evaluations of six different applications written in the python programming language, using the same data set. These applications are respectively; It uses artificial neural network, assisted decision tree, pure Bayesian classifier, logistic regression classifier, support vector machine and automatic encoder methods.

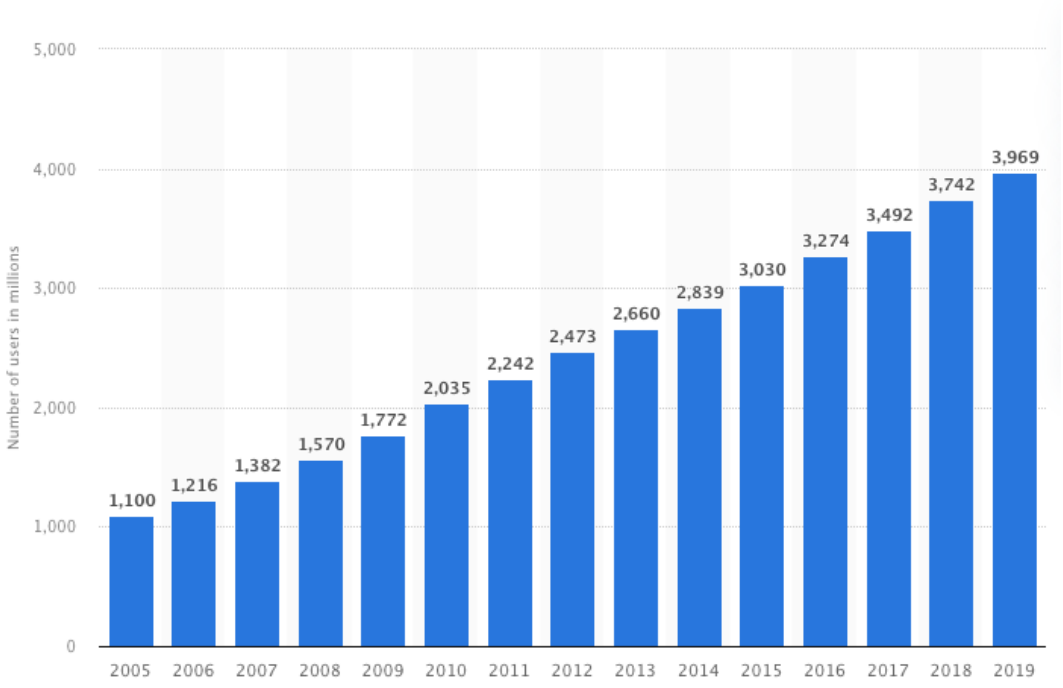
Findings are evaluated on true accuracy score, F1 score and confusion matrix. In addition, the working time for each application, in the same physical environment, with the same data set is also taken into account.

Each different method used achieves different successful results in different types of data sets. Examples of this can be seen in many studies. However, in the data set used in this thesis, which of these methods is or is more efficient is shown in this section.

Keywords: *Anomaly, Classification, Web Attacks*

1. GİRİŞ

Son yıllarda internet kullanımın hacminin çok ciddi bir şekilde artması, resmi ve ticari internet sitelerinin çok önemli bir duruma gelmesini sağladı. Şekil 1.1'de görüldüğü gibi, dünya çapında internet kullanıcılarının sayısı 2005 yılında yaklaşık 1 milyar iken, 2009 yılına gelindiğinde 4 milyar seviyelerine yaklaşmıştır. Bu durumla beraber bir çok risk faktörü de oluşmaya başladı. İnternet üzerinden yapılabilen resmi işlemler, gizliliği bulunan dosyaların internete açılması, internet üzerinden yapılan ticaretler, bankacılık işlemleri gibi ciddi konular, oluşan riskler nedeniyle tehlikeli hale geldi. İçinde bulunduğumuz bu çağda, bu tehlikeleri önleme amaçlı olarak sektörde, siber güvenlik kavramı oluştu. Birçok farklı tehlike unsuru farklı yöntemlerle tespit edilmeye ve engellenmeye çalışılmaktadır.



Şekil 1.1. İnternet Kullanıcı Sayılarının Yıllık Artışı

Kaynak: www.statista.com (27.11.2021)

İnternet trafiği bu kadar yoğunken, kötü niyetli girişimleri insan gücüyle engellemek de imkansız hale gelmiştir. Milyonlarca hatta milyarlarca istek yapılan

bir web sitesine yapılan sıradışı istekleri tespit etmek ancak bunu otomasyon haline getirerek mümkün olacaktır.

Web sitelerine yapılan anormal isteklerin birçok sebebi olabilir. Bunlardan en yaygın olanı, temel olarak web sitesi sunucunun kaynaklarını tüketip hizmet kesintisine uğraması için yapılan kısa sürede çok fazla istek ile yapılan saldırılardır. Bir diğeri ise, içerik hırsızlığıdır. Bu durum hem kaynak tüketmekte hem de haksız kazanç sağlamaktadır. Örnek vermek gerekirse; sahibinden.com sitesinde emlak ilanlarını analiz edip, bu veriyi emlakçılara hizmet olarak satan bazı kurumlar bulunmaktadır. Bu analizi yapabilmek için belirli aralıklarda tüm ilanların görüntülenmesi gerektiği ve bunun sürekli tekrar etmesi gerektiği düşünüldüğünde ne kadar ciddi bir trafiğin oluştuğu anlaşılmaktadır. Bu konu özelinde sahibinden.com'un, bu tarz istekleri analiz edip engellemek üzere uygulamalar geliştiren mühendislerden oluşan bir ekibi bulunmaktadır.

Bu tezde, kullanıcı trafiğindeki anormal olan isteklerin tespit edilmesi ve hızlı bir şekilde engellenmesini sağlamaktır. Bunun temelinde, normal olan web istekleriyle, anormal olan web isteklerinin ayrıştırılması (isteklerin sınıflandırılması) yer almaktadır. Anomali durumu belirli bir zaman çerçevesinde yapılan isteklerin tümüne bakılarak hesaplanacaktır. Anlık ya da çok kısa sürede tespit edilen anormal trafik engellenebilecektir.

Anormal kullanıcı isteklerinin tespit edilmesiyle, kaynak tüketim amacıyla yapılan saldırılar, içerik hırsızlığı için yapılan istekler gibi tüm normal kullanıcı davranışları içerisinde bulunmayan durumların engellenmesi sağlanabilecektir. Bu sayede kaynakların daha verimli kullanılmasıyla maliyetler düşecek, veri hırsızlığının önüne geçilecek ve istenmeyen bot yazılımları engellenmiş olacaktır.

1.1. Literatür Taraması

Ng ve ark., internet kullanımının yaygınlaşmasıyla büyüyen siber güvenlik sorununa değinmiş, yetkisiz giriş ve saldırı ataklarını (DoS) tespit etmek üzere veri madenciliği yöntemlerini kullanmışlardır. Bu çalışmada veri madenciliği yöntemlerinin bu tespit işlemi için işlevselliği sorgulanmıştır. Saldırı ataklarının tespiti için kümeleme (clustering) tekniklerinden faydalanılmıştır (Ng ve ark., 2015).

Bienias ve ark., internet trafiğindeki artış ve buna bağlı artan tehdit nedeniyle güvenliğin daha önemli hale geldiğine vurgu yapmışlardır. Bu çalışmada yarı

gözetimli eğitimin başarı oranını ciddi şekilde arttıracağı belirtilmiştir. Model eğitimi için çok sayıda normal kullanıcı isteği incelenmiş. En büyük kaygı, gerçek olmayan anormal istek tespitleridir. Veri gizliliğini korumak için kullanılan erişim kayıtlarının anonimleştirilmesi gerektiği savunulmuştur. Destek vektör makinesi (DVM), k-en yakın komşu ve açı tabanlı aykırı değer tespiti yöntemleri değerlendirilmiştir. DVM yöntemi için; benzer problemlerde yaygın olarak kullanıldığı ve başka algoritmalarla desteklendiği durumlarda oldukça başarılı olduğu vurgulanmıştır. Ancak veri kümesinin boyutunun ve özellik sayısının fazla olmasının, eğitim maliyetini ciddi şekilde arttırdığı, ayrıca diğer yöntemlere göre oldukça fazla yanlış pozitif çıktı ürettiği gözlemlenmiştir (Bienias ve ark., 2019).

Kbar ve Alazab, saldırı tipleri hakkında literatür taraması yapmışlardır. Bu çalışmada, anomali durumlarının tespitinin yanısıra, saldırı tiplerinin analizi ve bununla ilgili bir veritabanı oluşturulması üzerine çalışılmıştır. Bu sayede yanlış pozitif ve yanlış negatif değerleri azaltmayı hedeflemiştir. Ayrıca anomali tespiti ile birlikte bir kural tabanlı çalışma da bu işlemin bir parçasıdır. Katmanlı bir mimariden söz edilmekte, hem anomali tespiti, hem kural tabanlı uygulama, hem de saldırı tipinin tespit edilmesi üzerine çalışılmıştır (Kbar ve Alazab, 2019).

Komazec ve Gajin, modern ağların gelişmiş bir şekilde, birçok farklı tehdide karşı koruma sağlaması gerektiğini vurgulamıştır. Bu çalışmada artan ağ saldırı eğiliminden bahsedilmiştir. Kötü niyetli saldırıları engellemek için erken tespitin çok önemli olduğu belirtilmiştir. IP bazlı analizler ile saldırı şekilleri değerlendirilmiştir. Davranış modelinin çıkarılmasının, konu özelinde çok zor olduğu belirtilmiştir. Bunun sebebinin çok yoğun normal, çok az seviyede anormal ya da durumu anlaşılamayan trafik olduğu saptanmıştır. Çok büyük veri setleri içerisinden, veri setine oranla çok küçük bir kümenin anormal olarak, doğru şekilde tespit edilebilmesi bu çalışmanın en önemli sonucudur (Komazec ve Gajin, 2019).

Merrill ve Eskandarian, otomatik kodlayıcının, anomali tespiti için temel bir derin öğrenme yaklaşımı olduğunu belirtmişlerdir. Bu çalışmada, otomatik kodlayıcının eğitimi, anormal isteklerin normal olanlardan daha yüksek yeniden yapılandırma hatası üreteceği varsayımı ile eğitilmiştir. Eğitim için yeterli süre verildiği takdirde başarılı sonuç üreteceği belirtilmektedir. Bu çalışmada otomatik kodlayıcı, kümülatif hata puanlama, yüzdelik kayıp ve erken durdurma yöntemleri ile güçlendirilmiştir. Model eğitiminin çok maliyetli olduğu sonucu elde edilmiştir.

Ayrıca kullanılan farklı veri setlerine göre başarı skoru ciddi şekilde farklılık göstermiştir (Merrill ve Eskandarian, 2020).

Kim ve Kim, siber saldırıların yıkıcı bir etkiye sahip olabileceğini vurgulamışlardır. Dolayısıyla bu çalışmada anomalilerin çok hızlı bir şekilde tespit edilmesi gerekliliği belirtilmiştir. Neredeyse gerçek zamanlı olarak çalışan bir erken uyarı sisteminin gerekliliğinden bahsedilmiştir. Veri setlerinin çok büyük olmasının ve verilerin çok büyük bölümünün normal olmasının, verilerin analiz edilmesinin ve anomali tespitini güçleştirdiğinden bahsedilmiştir (Kim ve Kim, 2020).

Upman ve Goranin, nesnelerin interneti (IoT) kullanımının hızla artmasıyla, bu sistemlerin merkezi olarak güvenlik tehditlerinin de arttığını belirtmişlerdir. Bu çalışmada önerilen yöntem ile yanlış pozitif oranının %0.2, doğruluk oranının ise %99.3 olduğu belirtilmiştir. Bu çalışmada k-en yakın komşu, DVM, lojistik regresyon, karar ağacı, rastgele orman, yapay sinir ağı ve radyal tabanlı fonksiyon yöntemleri denenmiştir. Bu araştırma sonucunda radyal tabanlı fonksiyon yönteminin çok az sayıda yanlış pozitif tahminde bulunduğu ve başarı oranının çok yüksek olduğu açıklanmıştır. Ancak araştırmada kullanılan veri setinin laboratuvar ortamında oluşturulduğu, boyutunun çok küçük olduğu, içinde uygun miktarda anormal istek barındırdığı belirtilmiş ve aynı çalışmanın daha büyük, gerçek bir veri seti üzerinde denenmesi gerekliliği, denenmesi durumunda sonuçların değişebileceği vurgulanmıştır (Upman ve Goranin, 2020).

Ullah ve Mahmoud, nesnelerin interneti (IoT) cihazlarının sayısı ve bunlarla ilgili uygulamaların artmış olmasıyla birçok altyapı potansiyel hasarlara sebep olacak tehditlerin hedefi haline geldiğini belirtmişlerdir. Bu çalışmada, anomali tabanlı iyi yapılandırılmış IoT veri setlerinin olmaması sebebiyle bu ağların analiz edilmesinin ne kadar zor olduğu vurgulanmıştır. Bu çalışmada Gauss saf Bayes, lojistik regresyon, karar ağacı, rastgele orman yöntemleri kullanılmıştır. Kendi veri setleri üzerinde, ikili sınıflandırma yapıldığında tüm yöntemler %99 ve üzeri başarı göstermiştir. En başarılı yöntem ise karar ağacı algoritması olmuştur. Yapılan testler 10 tekrarın ortalaması üzerinden hesaplanmıştır (Ullah ve Mahmoud, 2020).

Sunny ve ark., araçlara özel ağlar (VANET) ve akıllı ulaşım sistemi (ITS) gibi teknolojilerin ortaya çıkması ile birlikte modern otomobillerin sadece izole edilmiş bir mekanik cihazdan ziyade sofistike bir siber-fizik sisteme dönüştüğünü belirtmişlerdir. Bu çalışmada aracın harici arayüzlerini korumanın da önemli bir

gereklilik olduğu vurgulanmıştır. Bu çalışmada önerilen sistem, simüle edilmiş saldırı senaryoları ile gerçek bir ağ üzerinde denenmiştir. Elde edilen sonuçların hem başarı oranları hem de hızlı tepkime süreleri verilmiştir. Yanlış pozitif sonuçların çok düşük olması gerekliliği üzerinde vurgu yapılmıştır. Tepki süreleri milisaniyeler mertebesinde değerlendirilmiştir (Sunny ve ark., 2020).

Yang ve ark., enerji şirketlerinin hızla büyümesi ve güç dağıtım verilerinin büyük miktarlara çıkmasıyla bu şirketlerin daha güvenli çalışmasının önemli bir hale geldiğini belirtmişlerdir. Bu çalışmada otomatik kodlayıcı kullanılarak anomali tespiti yapılmıştır. Kullanılan veri seti üzerinde otomatik kodlayıcı yöntemi çalıştırılmış, her iterasyondaki hata eşik değerleri güncellenerek optimum bir hale getirilmeye çalışılmıştır. Farklı eşik değerlerinde, farklı F1 skorları elde edilmiştir. Sonuç olarak kullanılan veri seti değişiklik gösterdikçe ya da alışkanlıklar değiştikçe, eşik değerinin de güncellenmesi gerekliliği ortaya çıkmıştır ve bu da ekstra bir bakım maliyeti olarak değerlendirilmiştir (Yang ve ark., 20201).

Bu çalışmada literatürde yer almayan özgün bir veri seti kullanılarak yapay sinir ağı, destekli karar ağacı, saf Bayes sınıflandırıcı, lojistik regresyon sınıflandırıcı, destek vektör makinesi ve otomatik kodlayıcı yöntemleriyle anomali tespiti yapılmıştır. Bu yöntemlerin sonuçları kabul görmüş performans metrikleri ile verilmiştir.

Bölüm 2'de bu tez için oluşturulan veri seti hakkında bilgiler yer almakta, Bölüm 3'te kullanılan algoritmalar tanıtılmakta, Bölüm 4'te tezdeki uygulamalara dair bulgular verilmekte, Bölüm 5'te tezin tartışma kısmı yer almakta, Bölüm 6'da ise sonuçlar belirtilmektedir.

2. GENEL BİLGİLER

Veri seti üzerinde çalışmaya başlamadan önce verinin doğru analiz edilip, gerekiyor ise değişiklikler yaparak uygulamaya hazır hale getirilmesi gerekir. Bu tezde kullanılan veri seti, her bir satırında bir web isteğinin olduğu bir zaman serisinden oluşmaktadır.

Uygulamalarda ise bir zaman aralığındaki kullanıcı davranışları analiz edilmiştir. Bu sebeple veri setinin zaman bazlı değil, kullanıcı bazlı olması gerekmektedir. Bunu sağlamak için uygulamaya başlamadan önce veri seti işlenerek, kullanıcı bazlı hale getirilmiştir. Yeni veri seti, her bir kullanıcın birim zamanda kaç adet istek yaptığı, isteklerin kaç milisaniye sürdüğü gibi kümülatif bilgilerden oluşmaktadır. Bu bilgiler sınıflandırılarak da anomali tespiti yapılmıştır.

2.1. Veri Seti

Bu çalışmada, küçük-orta ölçekli, dünyada birçok farklı ülkeden binlerce kullanıcısı olan "shippn.com" adresine ait, 14 Kasım 2020 ile 3 Aralık 2020 tarihleri arasındaki erişim kayıtları kullanılmıştır. Bu kayıtlar web sitesi sunucusu üzerinde bulunan "nginx" (ters vekil sunucusu) kullanıcı erişim kayıtlarıdır. Tüm web isteklerinin satırlar halinde, diskte bir dosya üzerinde saklanmasından oluşur. Bu verinin kullanımına dair gerekli izinler alınmıştır.

Erişim kayıtları, istek yapılan tarih-zaman, kullanıcının IP adresi, uzantısı, kullanılan HTTP metodu, kullanıcının "user-agent" bilgisi, isteğin kaç milisaniye sürdüğü, kaç kilobayt veri içerdiği bilgilerinden oluşmaktadır. Şekil 2.1'de tipik bir isteğin erişim kaydı görülmektedir.

```
157.49.214.226 - - [02/Dec/2020:06:29:00 +0000] "GET /assets/img/footer/stripe-logo.png HTTP/1.1"
157.49.214.226 - - [02/Dec/2020:06:29:00 +0000] "GET /favicon.ico HTTP/1.1" 304 0 "https://www.sh
157.49.214.226 - - [02/Dec/2020:06:29:00 +0000] "GET /api/host/countries HTTP/1.1" 200 883 "https
157.49.214.226 - - [02/Dec/2020:06:29:00 +0000] "GET /api/countries HTTP/1.1" 200 5794 "https://
157.49.214.226 - - [02/Dec/2020:06:29:00 +0000] "GET /api/host/homepage HTTP/1.1" 200 767 "https
157.49.214.226 - - [02/Dec/2020:06:29:00 +0000] "GET /assets/fonts/glyphicons-halflings-regular.
```

Şekil 2.1. Nginx Kullanıcı Erişim Kayıtları Örneği

Bu çalışmada anomali tespiti istek bazında değil, kullanıcı (yani IP adresi) bazında gerçekleştirilmiştir. Bunun için ise erişim kayıtlarının belirli bir zaman aralığındaki kısmı alınarak, bu kayıtları kullanıcı bazlı, anlamlı bir hale getirmek gerekmektedir.

Bu veri seti içerisinde gerçek anomali oluşturan (gerçek saldırı durumları) bulunmakta ve bunlar bilinmektedir. Bu özellikle makine öğrenmesi algoritmalarının eğitimi için değerli bir bilgidir. Çalışmada kullanılan modeller, bu bilgi doğrultusunda eğitilmiştir.

2.2. Verinin Hazırlanması

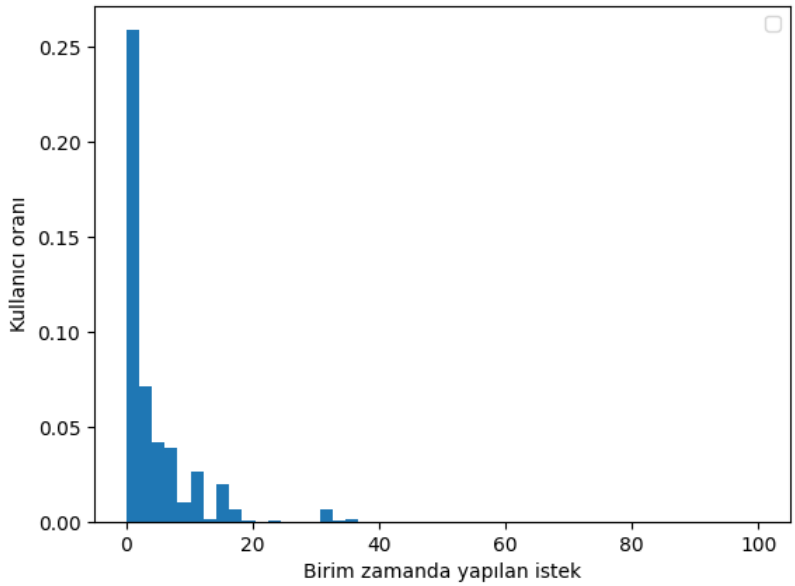
Ham olarak bulunan erişim kayıtlarından, kullanıcı (IP) bazlı yeni bir veri seti oluşturulmuştur. Oluşturulan bu veride her IP adresi için birim zamanda kaç tane istek attığı, bu isteklerin süreleri, isteklerin veri boyutları, kullanılan uzun süreli erişim kayıtları baz alınarak hesaplanmış "user-agent" popülerliği bilgileri yer almaktadır. Şekil 2.2'de görüldüğü gibi, yeni veri seti bir zaman serisi değil, IP adresi bazında hazırlanmıştır. Çalışmada, hazırlanan bu yeni veri seti kullanılmıştır. Ayrıca anomali oluşturduğu tespit edilen IP adresleri, veri üzerinde etiketlenmiştir.

remote_addr	time_minus	requests_per_seconds	body_max	body_mean	body_median	user_agent_popularity	suspect_ratio
1.124.104.15	8.0	4.0	576482.0	36306.6875	7352.0	0.0144632605062794	0.0
1.124.105.174	8.0	4.0	576672.0	36250.375	7352.0	1.893552735878994e-05	0.0
1.124.110.136	294.0	0.17	576672.0	28099.5	3028.0	1.0882486987810307e-05	0.0
1.125.107.116	1.0	4.0	19594.0	8343.75	6499.5	1e-06	0.0
1.126.110.199	132.0	0.47	576672.0	17863.83870967742	2456.5	0.0002039378061515	0.0
1.127.105.226	630.0	0.47	1082725.0	19305.164429530203	0.0	8.923639330004453e-06	0.0
1.127.108.211	2.0	2.0	19594.0	8343.75	6499.5	1e-06	0.0
1.127.110.223	25.0	2.6	576672.0	17878.292307692307	0.0	0.0001007718295071	0.0
1.128.104.44	1.0	4.0	19594.0	8343.75	6499.5	0.0001543136654871	0.0

Şekil 2.2. Eğitim İçin Hazırlanan Veri Seti Örneği

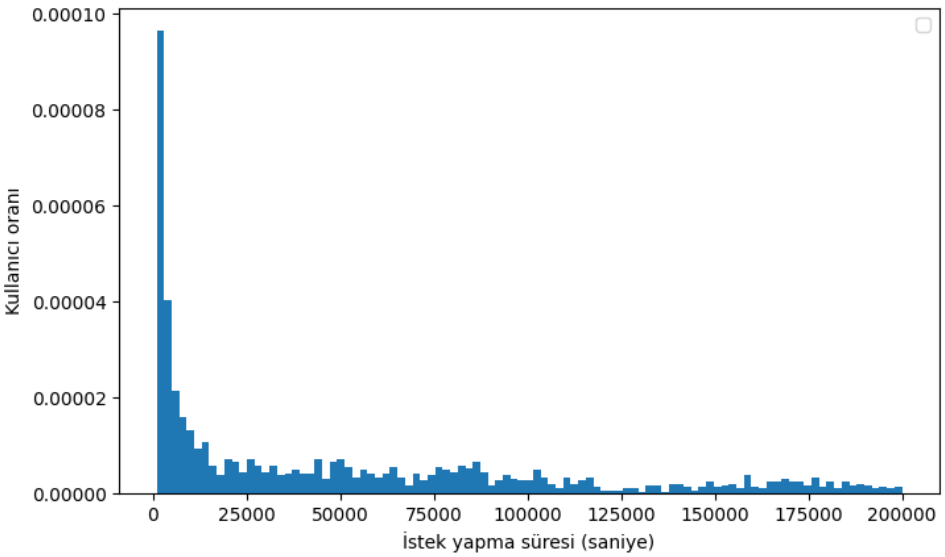
2.3. Oluşturulan Veri Setine Genel Bakış

Birim zamanda yapılan istek sayıları incelendiğinde; kullanıcı bazlı olarak saniyede ortalama 4.4 istek yapıldığı gözlemlenmektedir. Şekil 2.3'te birim zamanda yapılan istek sayısının histogramı görünmektedir. Bu veri setinde yaklaşık 12.619 normal, 2.550 anomali oluşturan kayıt bulunmaktadır. Her bir satırda, bir kullanıcıya ait bilgiler bulunmaktadır.



Şekil 2.3. Birim Zamanda Yapılan Ortalama İstek Sayılarının Kullanıcı Oranına Bağlı Histogramı

Her kullanıcının son istek zamanı ile ilk istek zamanının farkı, yani bu zaman aralığında sunucuya istek yönelttiği süre bazında incelendiğinde de aşağıdaki histogram görülmektedir. Bu süre, Şekil 2.4'de görüldüğü gibi, ortalama 23,053 saniye olarak hesaplanmıştır.



Şekil 2.4. Kullanıcı Aktif Kullanım Süresinin Kullanıcı Oranına Bağlı Histogramı

3. GEREÇ VE YÖNTEM

Bu bölümde uygulamaların geliştirme ortamı, kullanılan kütüphaneler ve her bir farklı uygulamada kullanılan makine öğrenmesi yöntemleri hakkında teorik bilgilerden bahsedilmiştir.

3.1. Geliştirme Ortamı

Bu çalışmada yapılan uygulamalar "python" programlama dilinin 3.8 versiyonu ile geliştirilmiştir. Veri üzerinde işlem yapmak için "numpy" ve "pandas" kütüphaneleri, makine öğrenmesi algoritmalarının uygulanması için "sklearn" ve "tensorflow" kütüphaneleri ve veri görselleştirme için "matplotlib" kütüphanelerinden faydalanılmıştır. Çalışma yapılan ortam "Mac OS X"tir.

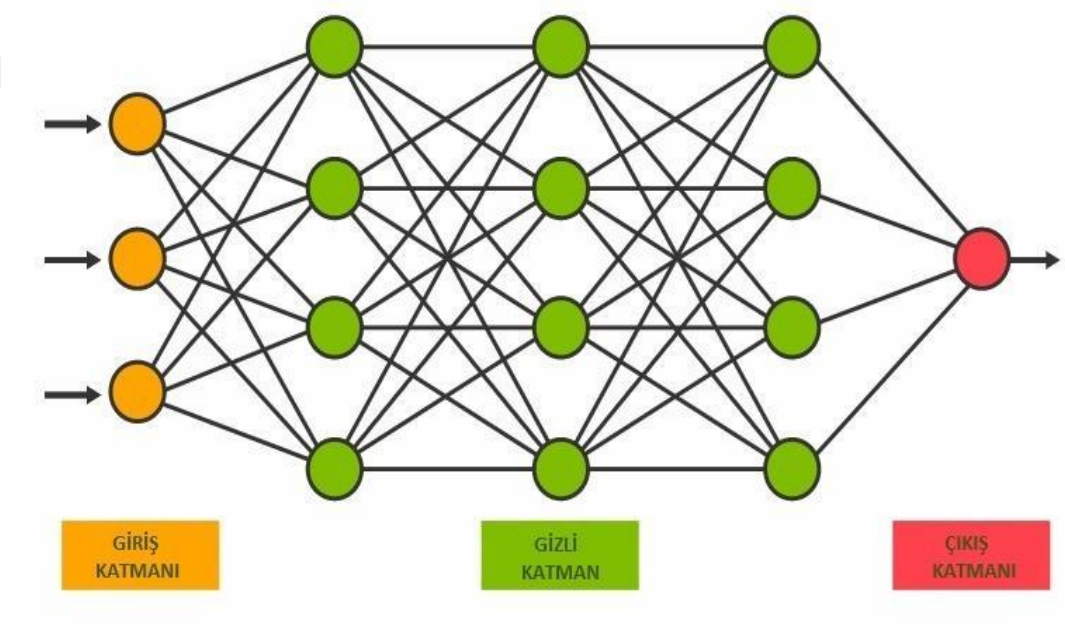
3.2. Anomali Tespitinde Kullanılan Metodlar

Çalışmada kullanılan veri düşünüldüğünde, normal kullanıcıların normal olan web isteklerine ait bilgiler ve kötü niyetli kullanıcıların ya da bot yazılımlarının yapmış olduğu web isteklerine ait bilgiler yer almaktadır. Elimizdeki bu bilgiye dayanarak problemin çözümü, sınıflandırma yapılarak mümkün olmaktadır. Yapılan istekler, normal ve anormal olarak gruplandırarak sonuca ulaşılmıştır. Bunun için birkaç farklı sınıflandırma metodu kullanılmış ve karşılaştırılmaları yapılmıştır. Veri setinin %10'luk kısmı test, %90'lık kısmı ise eğitim için kullanılmıştır.

3.2.1. Yapay Sinir Ağı

Yapay sinir ağları, şimdiye kadar icat edilen en güçlü programlama paradigmalarından biridir. Ancak geleneksel yaklaşımdan ayrıldığı nokta, geleneksel yaklaşımda programda satır satır ne yapılması gerektiğinin belirtilmesine karşın, yapay sinir ağlarında programın veri yapısını inceleyerek öğrenmesi sonucunda problemi çözmesidir. Verilerden otomatik olarak öğrenme yapay zekanın temelinde yer almaktadır (Aggarwal C.C., 2018).

Yapay sinir ağları (YSA), hayvan beyninin çalışma sisteminden esinlenerek geliştirilmiş bir bilgi işleme sistemidir. YSA ile biyolojik sinir sisteminin çalışma şekline benzer, yani sinir hücrelerinin ve bu hücrelerin birbirleri arasında kurduğu sinaptik bağın dijital olarak modellenmesi sağlanır. Şekil 3.1'de de görüldüğü gibi, her bir bağlantı, diğer nöronlara (sinir hücrelerine) bir sinyal iletebilir. Bu bağlantılara dal denir. Yapay bir nöron bir sinyali alır ve ardından onu işleyerek kendisine bağlı diğer bir nörona iletebilir. Her bir nöronun çıktısı, girdilerinin toplamının doğrusal olmayan bir fonksiyonu tarafından hesaplanır. Dallar birer ağırlığa sahiptir. Bu ağırlıklar, öğrenme devam ettikçe hesaplanır. Ağırlık, her bir bağlantıda sinyali arttırır ya da azaltır.



Şekil 3.1. Örnek bir YSA

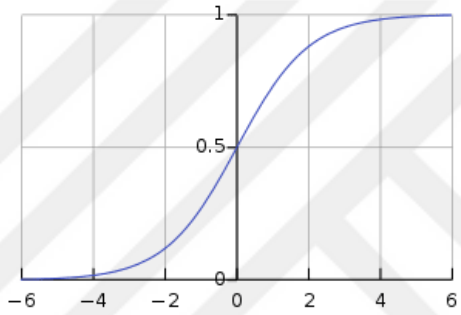
Kaynak: www.tibco.com (29.11.2021)

Yapay sinir ağları, bilinen girdilere karşılık gelen sonuçları içeren veri kümelerinin kullanımıyla eğitilir ve ağın kendi veri yapısı içinde depolanan ağırlıklı değerleri oluşturulur. Bu eğitim tekrarı arttıkça, hedef çıktıya giderek daha benzeyen bir çıktı üretmesine neden olacaktır. Önceden belirlenen bir performans kriterine ulaşıldığında eğitim sonlandırılır.

Yüksek karmaşıklığa sahip bir veriyi, dağ ağırlıkları ile çarpmak ve toplamak yetersiz olacaktır. Veri kümesi içindeki karmaşıklığı kazandıracak olan bir aktivasyon fonksiyonuyla, karmaşık fonksiyonun modellenmesi sağlanmaktadır.

YSA'da nöronların çıkış değerlerleri dallar aracılığıyla diğer nöronlara iletilmektedir ve bu aşamada bir katsayı ile çarpılmalıdır. Bu katsayı ile çarpılma işlemi matematiksel olarak basit olsa da, veri kümesi çok karışık olabilir. YSA'yı ne kadar derinleştirirsek derinleştirelim, bu kadar karmaşık modelleri yaratamayız. Bunu yapabilmek için uygun bir aktivasyon fonksiyonu kullanılır. Sigmoid, relu vb. problemin yapısına göre farklı bir aktivasyon fonksiyonu kullanılabilir. Doğru/yanlış, evet/hayır gibi iki durumlu bir değerle uğraştığımızda, çıkış katmanında sigmoid fonksiyonu tercih edilmelidir. Sınıflandırma için uygun bir fonksiyondur.

YSA'ya bir aktivasyon fonksiyonu olarak, Şekil 3.2'deki gibi "S" şeklinde eğri çizen bir "sigmoid" fonksiyonu kullanılmıştır.



Şekil 3.2. Sigmoid Fonksiyonu

Kaynak: en.wikipedia.org (03.11.2021)

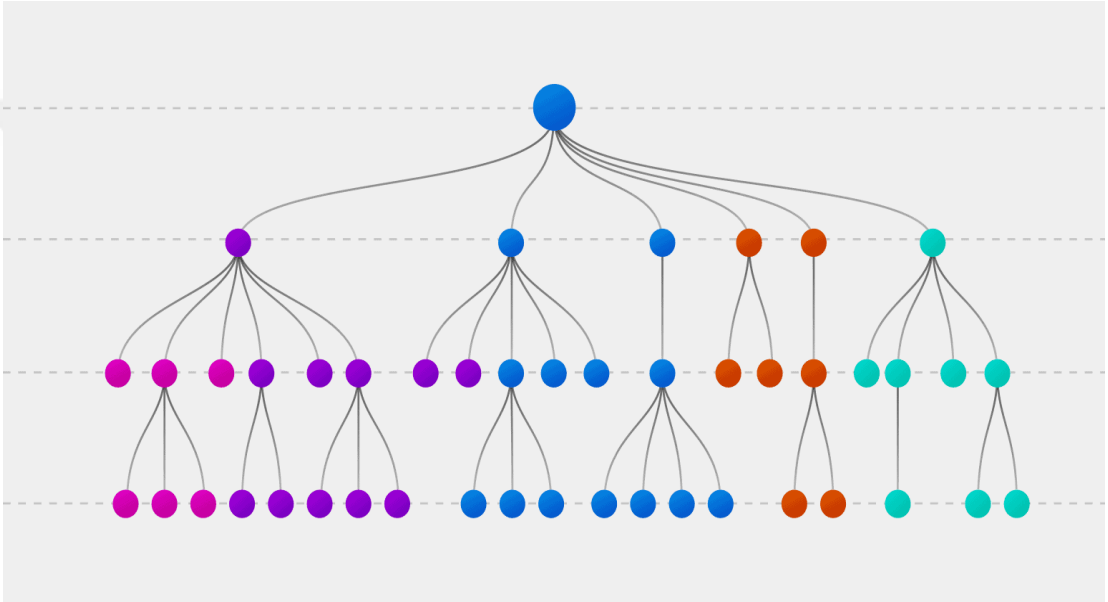
Bu fonksiyonun temel görevi, herhangi bir değeri 0 ile 1 arasındaki bir değere eşitlemektir. Böylece girdilerin ağırlıklı toplamını normalize etmemize yardımcı olacaktır.

Modeli eğitme noktasında, her girdiğinin pozitif ya da negatif bir ağırlığı hesaplanacaktır. Çok büyük negatif ya da pozitif olan ağırlık değerleri içeren girdiler, sonuç için daha önemli özellikler olacaktır.

Başlangıç ağırlık değerleri sıfır olarak da seçilebilir. Ancak bu şekilde algoritmanın yakınsaması ya mümkün olmayacak ya da çok uzun sürecektir. Bu sebeple, başlangıçta tüm ağırlık değerleri belli bir aralıkta, rastgele sayılardan oluşur. Eğitim veri seti işlenerek bu ağırlık değerleri hesaplanır. Eğitim veri setinin büyüklüğü ve çeşitliliği ne kadar fazla olursa, hata payı da o kadar az olacaktır.

3.2.2. Destekli Karar Ağacı

Karar ağaçları, sınıflandırma ve regresyon problemlerinde kullanılan, ağaç tabanlı algoritmalardan biridir. Şekil 3.3'de bir örneği görülen karar ağaçlarının ilk hücrelerine kök adı verilir, kök hücrelerinin altında kalan hücrelere düğüm denir. Karar ağaçlarının en alt kısmında yapraklar bulunur. Her gözlem, kökteki koşula göre "evet" ya da "hayır" olarak sınıflandırılarak bir alt düğüme geçer ve düğümlerde de bu şekilde devam eder. Düğüm sayısı arttıkça modelin karmaşıklığı da artar. Yapraklar ise bu sınıflandırmanın sonucu olarak belirlenir.



Şekil 3.3. Karar Ağacı

Kaynak: ai-pool.com (03.11.2021)

Seçilecek kök hücresinin, veri setini en iyi ayrıştıran etken olması beklenir. Aynı şekilde bu durum alt düğümler için de geçerlidir. Kök hücreden aşağı doğru dallandıkça önem azalır ve son olarak çıktı üretilir.

$$G_{ini} = 1 - \sum_{j} {p_j}^2 \qquad \textbf{(3.1)}$$

Denklem 3.1.'deki gibi, kök hücresi ve düğüm hücrelerinin seçimi için alt kümelerin saflık değerini veren "G_{ini}" hesaplanır. P_j, j sınıfının gerçekleşme olasılığıdır. Her sınıf için bu değer hesaplanarak karelerin toplamı birden çıkarılır. Sonuç sıfıra ne kadar yakınsa o kadar iyi ayrım yapılmıştır (Maimon O.Z., Rokach L. 2007). Her düğüm için bu hesap yapılarak en iyi ayrımı yapan etken, kök hücre olarak seçilir ve bu seçim aşağıya doğru her bir düğüm hücresi için tekrarlanır.

Bu yöntem ile birlikte "AdaBoost" algoritması kullanılarak güçlendirme (boosting) uygulanmıştır. Güçlendirme işleminin temel fikri, tek bir sınıflandırıcı kullanmak yerine, birden fazla zayıf sınıflandırıcının tahmin güçlerini birleştirerek, sonunda kompleks ve daha güçlü bir sınıflandırıcı elde etmek üzerinedir. Lojistik regresyon, karar ağacı gibi performansı diğer metodlara göre biraz daha düşük olabilen sınıflandırıcılar her iterasyonda veri seti üzerinde tekrar tekrar kurulur ve bir önceki aşamada kurulan sınıflandırıcının yaptığı hataları yapmamaya çalışır.

AdaBoost algoritması sonucunda, birden fazla sınıflandırıcı bir topluluk (ensemble) oluşturarak sonuç tahmini her iterasyonda oluşan farklı sınıflandırıcıların yaptığı farklı tahminlerin katkısıyla oluşur. Her farklı sınıflandırıcının sonuç tahminine ne kadar katkı sağlayacağı, üzerilerine atanan farklı ağırlık değerleri ile belirlenir.

3.2.3. Saf Bayes Sınıflandırıcı

Saf Bayes sınıflandırıcı, sınıflandırma işlemleri için kullanılan olasılık temelli, denetimli bir makine öğrenme modelidir. Temeli, "Bayes" teoremine dayanmaktadır. Saf Bayes sınıflandırıcı tarafından kullanılan model, güçlü koşullu bağımsızlık varsayımları yapar. Koşullu bağımsızlık varsayımı, bir sınıf verildiğinde, özellik değerlerinin tamamen bağımsız olmasıdır. Belirli bir sınıfın özellikleri arasında herhangi bir ilişki yoktur.

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)} \quad \textbf{(3.2)}$$

Denklem 3.2.'de görülen Bayes teoremi ile, B olayının oluştuğunu göz önüne alarak, A olayının oluşma olasılığını bulabiliriz.

3.2.4. Lojistik Regresyon Sınıflandırıcı

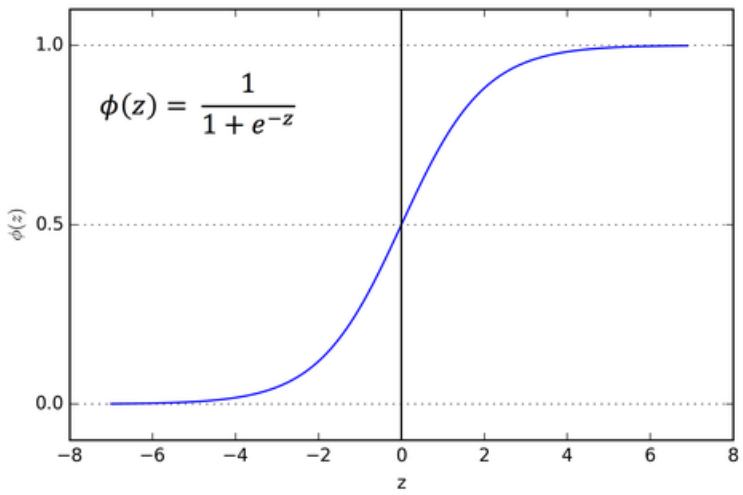
Lojistik regresyon, bağımlı değişken ikilileri durumunda yapılacak tahmine dayalı bir analizdir. Verileri tanımlamak ve bir bağımlı değişken ile bir veya daha fazla nominal, sıra, aralık veya oran düzeyinde bağımsız değişkenler arasındaki ilişkiyi açıklamak için kullanılır. Bu tür bir analiz, bir olayın olma olasılığını tahmin etmenize yardımcı olabilir (Harrell F.E. 2015).

Lojistik regresyon, istatistik ve makine öğrenmesinde oldukça fazla kullanılan sınıflandırma yöntemlerinden biridir. Adı ne kadar regresyon olsa da nümerik bir çıktıyı değil, bir sınıfı tahmin etmeye çalışmaktadır. Zaten bu tarz tahminler yapan yöntemlere genel olarak sınıflandırma yöntemleri denmektedir. Bu yöntemle iki ya da daha fazla sınıfın tahminlemesi yapılabilir. Sınıf tahminlemesi dendiğinde, ilk olarak olasılıksal sonuçlar düşünülmektedir. Çıktı olarak sadece tahmin edilen sınıf değil, bunun yanında o sınıfa ait olma olasılığı da belirtilir. Sunulan olasılık değeri sonucun keskinliğini de belirler.

$$b_0 + b_1x_1 + b_2x_2 + b_3x_3 + ... + b_px_p \qquad \textbf{(3.3)}$$

Denklem 3.3.'teki, b_0, b_1, …, b_p değerleri katsayıları gösterirken, x_1, x_2, …, x_p değeeleri de değişkenleri ya da bir diğer adıyla özellikleri gösterir. Bu denklem sonucunda elde edilen değer aslında bir çeşit skordur. Elde edilen bu skorun işaretine göre de lojistik regresyon modeli tahmin edilen değerin hangi sınıfa ait olduğuna dair bir çıkarım yapar.

Lojistik regresyon bir lineer (doğrusal) sınıflandırıcıdır. Bunun sebebi ise üstteki denklemde görüldüğü gibi, denklemin katsayılar bazında doğrusal olmasıdır. Denklemin genel yapısı "katsayı" x "özellik" şeklinde ilerlemektedir. Özellikler doğrusal olmasa dahi lojistik regresyon hala doğrusal olarak kalacaktır. Çünkü katsayılar doğrusallığını koruyacaktır. Bu denklem sonucunda elde edilen skor - sonsuz ve + sonsuz arasında olabilir. Skor değerini 0 ile 1 arasına indirgeyebilmek için Şekil 3.4'teki gibi bir bağlantı fonksiyonu kullanılır. Bu fonksiyona "sigmoid fonksiyonu" denir.



Şekil 3.4. Sigmoid Grafiği

Kaynak: towardsdatascience.com (16.11.2021)

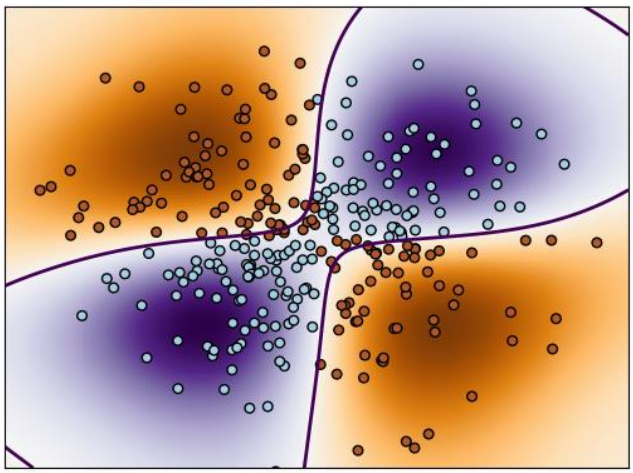
3.2.5. Destek Vektör Makinesi

1963 yılında Vladimir Vapnik ve Alexey Chervonenkis tarafından temelleri atılan Destek Vektör Makineleri, istatistiksel öğrenme teorisine dayalı bir gözetimli öğrenme algoritmasıdır. Her ne kadar temelleri 60'lı yıllara dayansa da 1995 yılında Vladimir Vapnik, Berhard Boser ve Isabella Guyon tarafından geliştirilmiştir (Boser B.E., Guyon I.M., Vapnik V.N. 1992).

Destek vektör makinesi (DVM), eğitim veri setindeki herhangi bir noktadan en uzaktaki iki sınıf/kategori arasında bir karar sınırı bulan, vektör uzayı tabanlı bir makine öğrenme yöntemidir. Vektör uzayının boyutu, veri setindeki özellik sayısı kadardır. Sınıflandırma ve regresyon analizinde kullanılır. Denetimli bir eğitim modelidir.

Destek vektör makineleri, veri setinin doğrusal olup olmaması durumuna göre ikiye ayrılmaktadır. Bu tezde kullanılan veri seti çok özellikli olduğu için doğrusal bir ayrım söz konusu değildir.

Doğrusal olmayan, Şekil 3.5'teki gibi bir veri kümesinde DVM'ler doğrusal bir hiper-düzlem çizemez. Bu nedenle çekirdek numarası (kernel trick) kullanılır. Çekirdek yöntemi, doğrusal olmayan verilerde makine öğreniminin etkisini yüksek oranda arttırmaktadır. En çok kullanılan çekirdek yöntemleri olarak; "polinom çekirdek" ve "Gauss RBF (Radial Basis Function) çekirdek" örnek verilebilir. Bu tezde RBF çekirdeği kullanılmıştır (Soman P. K., 2009).



Şekil 3.5. Doğrusal Olmayan DVM

Kaynak: medium.com/@k.ulgen90 (16.11.2021)

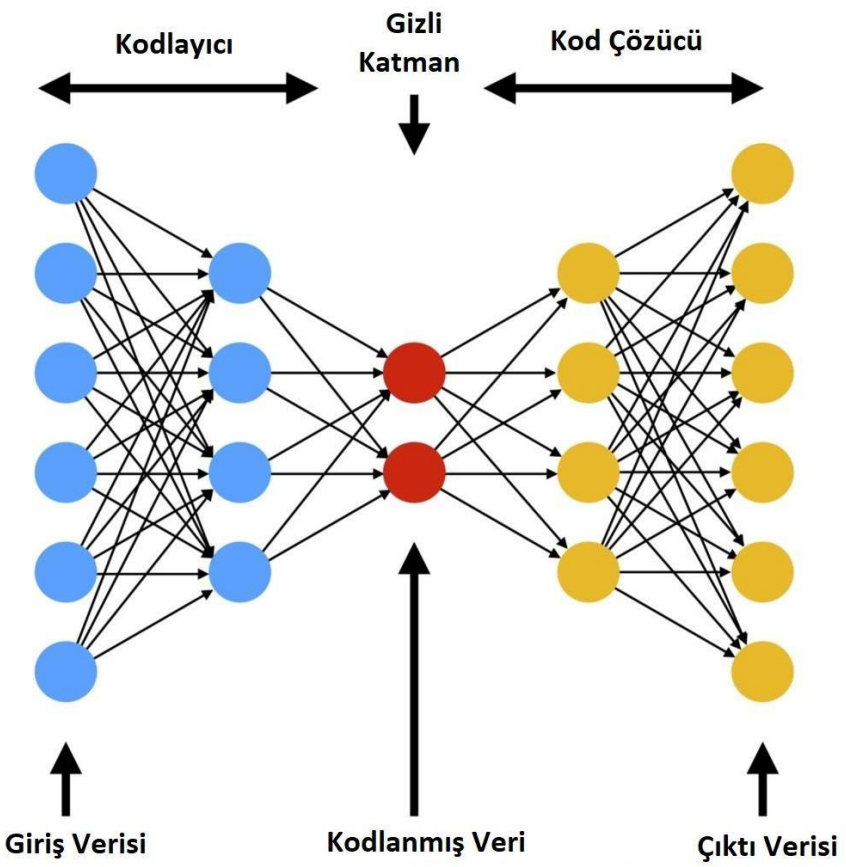
RBF, sonsuz boyuttaki destek vektör makinelerini bulur ve her bir noktanın belirli bir noktaya ne kadar benzediğini normal dağılım ile hesaplar, ona göre sınıflandırır. Dağılımın genişliği "gamma" hiperparametresi ile kontrol edilir. Gamma değeri ne kadar küçük olursa, dağılım o kadar geniş olur.

3.2.6. Otomatik Kodlayıcı

Otomatik kodlayıcı, denetimsiz bir öğrenme modeli oluşturmak için kullanılan bir tür yapay sinir ağıdır. Genel fikri veriyi temsil eden bir timsal vektörü öğrenmektir. Otomatik kodlayıcı, girdi katmanındaki değeri çıktı katmanına kopyalayan bir sinir ağıdır. Veri eğitilirken kendi etiketlerini ürettiği için öz-denetimli bir öğrenme modeli olarak tanımlanabilir.

Otomatik kodlayıcılar, birden fazla gizli katmana sahip olabilen ileri beslemeli sinir ağlarıdır. Bu ağlar, çıktı katmanındaki girdi verilerini yeniden oluşturmaya çalışır. Otomatik kodlayıcılardaki gizli katmanın boyutu, girdi verilerinin boyutundan daha küçük olduğundan, girdi verilerinin boyutu, gizli katmanda daha küçük boyutlu bir kod alanına indirgenir. Ancak, çok katmanlı bir otomatik kodlayıcıyı eğitmek zordur. Bunun nedeni, derin gizli katmanlardaki ağırlıkların neredeyse hiç optimize edilmemiş olmasıdır (Tan C. C., Eswaran C., 2010).

Genel mimarisi, Şekil 3.6'da da görüleceği üzere, kodlayıcı ve kod çözücü olarak adlandırılan iki ayrı bölümden meydana gelir. Kodlayıcı, veriyi özümseyen bir vektör yaratırken kod çözücü bölüm ise bu vektörü kullanarak tekrar yeni bir veri oluşturur. Kodlayıcıya verinin girdi boyutu ile kod çözücünün sonucunda oluşan çıktı boyutu aynıdır.



Şekil 3.6. Otomatik Kodlayıcı

Kaynak: medium.com/autoencoder-for-anomaly-detection (03.11.2021)

4. BULGULAR

Bu bölümde python programlama dili ile yazılmış, aynı veri setini kullanan altı (6) farklı uygulamanın değerlendirmeleri yer almaktadır. Bu uygulamalar sırasıyla; yapay sinir ağı, destekli karar ağacı, saf Bayes sınıflandırıcı, lojistik regresyon sınıflandırıcı, destek vektör makinesi ve otomatik kodlayıcı yöntemlerini kullanmaktadır.

Bulgular, gerçek doğruluk skoru, F1 skoru ve karışıklık matrisi üzerinden değerlendirilmektedir. Ayrıca her uygulama için, aynı fiziksel ortamda, aynı veri setiyle çalışma süresi de göz önünde bulundurulmaktadır.

Kullanılan her farklı yöntem, farklı tipteki veri setlerinde farklı başarılı sonuçlar elde etmektedir. Yapılan birçok çalışmada bunun örnekleri görülebilir. Ancak bu tezde kullanılan veri setinde bu yöntemlerden hangisinin ya da hangilerinin daha verimli olduğu bu bölümde gösterilmiştir.

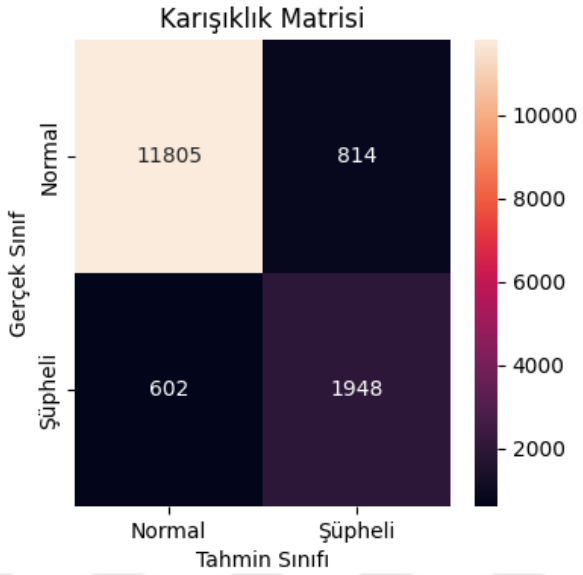
4.1. Yapay Sinir Ağı

Temel olarak bir YSA, nöronlar ve dallardan ve bu dalların ağırlık değerlerinden oluşmaktadır. Bu uygulamada eğitime başlamadan önce her bir ağırlık değeri rastgele sayılar ile belirlenir. Eğitimin her aşamasında “sigmoid” fonksiyonu kullanılarak, ağırlık değerleri güncellenir. Optimum bir ağırlık matrisi oluşana kadar eğitim devam eder.

Uygulamanın toplam çalışma süresi ortalama olarak 166 saniye sürmektedir. Doğruluk (accuracy) skoru 0.907, kesinlik (precision) skoru 0.910, duyarlılık (recall) skoru 0.907 ve F1 skoru 0.908 olarak bulunmuştur.

Şekil 4.1’teki karışıklık matrisine bakarak, skorlar genel olarak değerlendirildiğinde, sonuçlar başarılı görülmektedir. Ayrıca aşağıdaki karışıklık matrisine bakıldığında da, normal olan isteklerin %94’ünün normal olarak saptandığı, şüpheli işlemlerin ise yaklaşık %76’sının şüpheli olarak saptandığı görülebilmektedir.

Her ne kadar şüpheli işlemlerin bir kısmını normal bir istek olarak tahmin etmiş olsa da, normal isteklerin %94'ünü normal olarak tahmin ettiği için büyük ölçüde başarılı olduğu söylenebilir. Ancak modelin çalışma süresi dakikalar mertebesindedir ve bu tarz bir uygulama için uzun bir süre olduğundan tercih sebebi olmayacaktır.



Şekil 4.1. YSA Karışıklık Matrisi

Bu uygulamadaki veri seti ile optimum ağırlık matrisi ve yüksek bir doğruluk skoru elde edebilmek adına, eğitim 15,000 iterasyon ile çalışmıştır. Uygulama çalışma süresinin uzunluğu da bununla doğru orantılıdır. Buradan yola çıkacak olursak; çalışma süresi kısalır ise, yani iterasyon sayısı daha az olursa, doğruluk skoru da düşük olacaktır. Kabul edilebilir bir doğruluk skoru için hem büyük miktarda eğitim veri seti hem de bolca iterasyon gerekecektir.

Şekil 4.2'de görülen kod bloğunda bu YSA modelinin eğitim sınıfı görülmektedir. Eğitim aşamasında, optimum ağırlık değerlerinin oluşması için, eğitim veri seti belirlenen iterasyon kadar çalışmış olacaktır. Bu uygulamada, aynı veri seti 15,000 iterasyon ile çalışmıştır.

```
9     def __init__(self):...
15
16    def sigmoid(self, x):...
19
20    def sigmoid_derivative(self, x):...
24
25    def train(self, training_inputs, training_outputs, training_iterations):
26        training_inputs = training_inputs.astype(float)
27        training_outputs = training_outputs.astype(float)
28
29        # training the model to make accurate predictions while adjusting weights continually
30        for iteration in range(training_iterations):
31            # siphon the training data via  the neuron
32            output = self.think(training_inputs)
33
34            # computing error rate for back-propagation
35            error = training_outputs - output
36
37            # performing weight adjustments
38            adjustments = np.dot(training_inputs.T, error * self.sigmoid_derivative(output))
39
40            self.synaptic_weights += adjustments
41
42    def think(self, inputs):
43        # passing the inputs via the neuron to get output
44        # converting values to floats
45
46        inputs = inputs.astype(float)
47        output = self.sigmoid(np.dot(inputs, self.synaptic_weights))
48        return output
```

Şekil 4.2. YSA Uygulaması Kod Bloğu

Şekil 4.2'de görülen ekran görüntüsünde, bu uygulamadaki eğitim sonucu oluşmuş ağırlık değerleri verilmiştir. Her bir özellik için farklı bir ağırlık değeri olduğu açıkça görülmektedir. Tahmin aşamasına gelindiğinde, her özellik bu ağırlık değerleri ile çarpılır ve "sigmoid" fonksiyonu çalıştırılarak, 0 ila 1 arasında bir değer elde edilir. Dolayısıyla çıktı olarak elimizde istatiksel bir sonuç oluşur. Bir eşik değeri geçenleri normal, geçemeyenleri anomali olarak değerlendirilir.

```
1    -6.883347225749686360e+06
2    4.754699091671790612e+00
3    -9.517551861120207608e+07
4    -9.691648107250422239e+06
5    7.026405560178128071e+06
6    9.297405583391241635e+01
```

Şekil 4.3. YSA Eğitim Sonucu Oluşan Ağırlık Değerleri

4.2. Destekli Karar Ağacı

Destekli karar ağacı uygulamasında çeşitli denemeler sonucunda iki farklı hiperparametre belirlenmiştir. Şekil 4.4'teki kod bloğunda görüleceği üzere; bunlardan

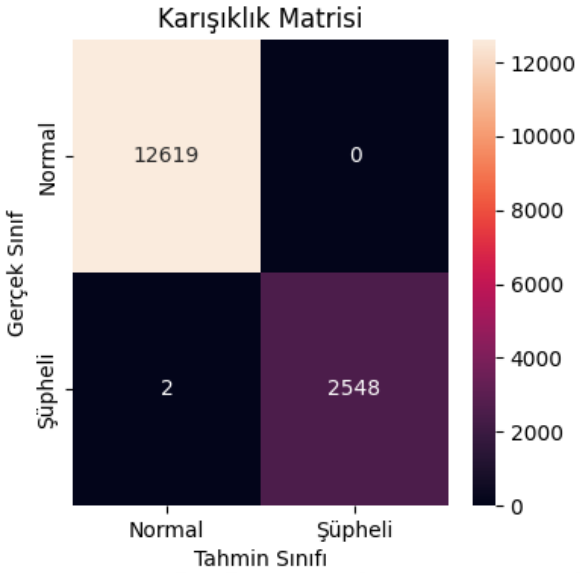
bir tanesi bir yaprak düğümde olması gereken minimum örnek sayısını ifade eden "min_samples_leaf" parametresidir. Bu hiper parametre belirtilmediği takdirde, varsayılan değeri 1'dir . Bu çalışmada ise bu değer 100 olarak belirlenmiştir.

Bir diğer hiperparametre ise ağacın maksimum derinliğini ifade eden "max_depth" parametresidir. Bu değer üç olarak belirlenmiştir. Belirleme işlemi yapılmadığı takdirde, tüm yapraklar saf olana kadar ya da tüm yapraklar bir yaprakta olması gereken minimum örnek sayısının altına düşene kadar dallanmaya devam edecektir.

```
86  DS = DecisionTreeClassifier(max_depth=3, min_samples_leaf=100)
87
88  n_estimators = 5
89
90  ada_real = AdaBoostClassifier(
91      base_estimator=DS,
92      learning_rate=0.1,
93      n_estimators=n_estimators,
94      algorithm="SAMME.R")
95
96  ada_real.fit(X_train, Y_train)
97
98  # discrete_pre = ada_discrete.predict(X_validation)
99  real_pre = ada_real.predict(X_validation)
```

Şekil 4.4. Destekli Karar Ağacı Uygulaması Kod Bloğu

Uygulamanın toplam çalışma süresi ortalama 1.7 saniye sürmektedir. Ortalama süre hesaplanırken, eğitim seti her defasında karıştırılarak, toplam 100 kere çalıştırılmıştır. Doğruluk (accuracy), kesinlik (precision), duyarlılık (recall) ve F1 skorları 0.999 olarak bulunmuştur. Aşağıdaki karışıklık matrisinde de görüleceği üzere, sadece 2 şüpheli işlemi normal olarak bulmuştur. Şekil 4.5'teki karışıklık matrisinde de görüldüğü gibi, normal olan tüm istekleri ise normal olarak işaretlemiştir. Sadece bu sonuçlara bile bakarak, uygulamanın oldukça başarılı çalıştığı söylenebilir.



Şekil 4.5. Destekli Karar Ağacı Karışıklık Matrisi

4.3. Saf Bayes Sınıflandırıcı

Bu uygulamada sonuçları iyileştirmek adına iki hiper parametre kullanılmıştır. Bunlardan biri sınıf önceliklerini belirten "priors" değeri. Bir diğeri ise hesaplama kararlılığını arttırmak için kullanılan, yumuşatma hiper parametresi "var_smoothing" dir. Bu hiperparametre değerleri çeşitli denemeler sonucunda elde edilmiştir. Kullanılan veri setine göre değişiklik göstermektedir. Şekil 4.6'taki kod bloğunda kullanımı görülmektedir.

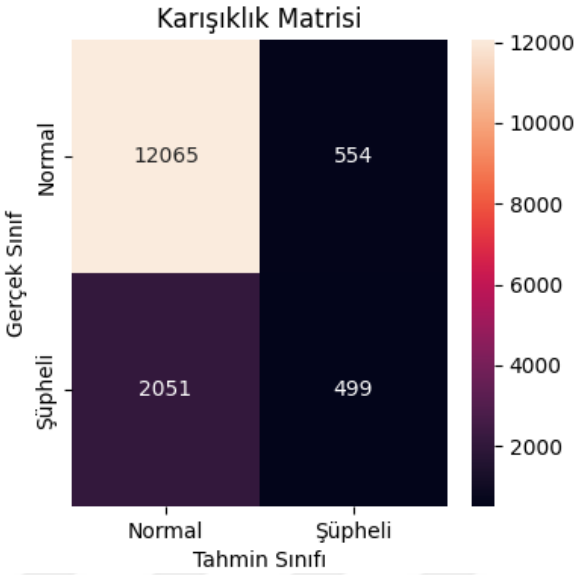
```
87  GaussianClassifier = GaussianNB(priors=[0.93, 0.07], var_smoothing=1e-100)
88  GaussianClassifier.fit(X_train, Y_train)
89
90  # print(GaussianClassifier.class_prior_)
91
92  y_pred = GaussianClassifier.predict(X_validation)
```

Şekil 4.6. Saf Bayes Uygulaması Kod Bloğu

Uygulamanın toplam çalışma süresi ortalama olarak 3.8 saniyedir. Ortalama süre hesaplanırken, eğitim seti her defasında karıştırılarak, toplam 100 kere çalıştırılmıştır. Doğruluk (accuracy) skoru 0.828, kesinlik (precision) skoru 0.791, duyarlılık (recall) skoru 0.828 ve F1 skoru 0.797 olarak bulunmuştur.

Şekil 4.7'deki karışıklık matrisine bakacak olursak, çok da başarılı bir sonuç elde edilemediği görülecektir. Normal olan isteklerin yaklaşık %4'ü şüpheli olarak

tahmin edilmiştir. Gerçek şüpheli işlemlerin de ancak %20'lik kısmı doğru tahmin edilmiştir.



Şekil 4.7. Saf Bayes Karışıklık Matrisi

4.4. Lojistik Regresyon Sınıflandırıcı

Bu uygulamada, Şekil 4.8'teki kod bloğunda görüleceği üzere, regresyon işlemine başlamadan önce normalizasyon işlemi yapılmıştır. Lojistik regresyon için normalizasyon aslında bir gereklilik değildir. Özellikleri standartlaştırmanın temel sebebi, optimizasyon için kullanılan tekniğin yakınsamasına yardımcı olmaktır. Özellikleri standartlaştırmak yakınsamayı daha hızlı hale getirecektir. Aksi takdirde, özellikler üzerinde herhangi bir normalizasyon işlemi yapılmadan da lojistik regresyon çalıştırılabilir.

Bu uygulamada, çeşitli denemelerin sonucunda, aşağıdaki kod bloğunda görülen bazı hiperparametreler kullanılmıştır. Bunlardan en önemlilerinden biri C değeridir. Makine öğreniminde C'ye hiperparametre denir. Diğer parametreler ise hiperparametreden ziyade, birer yardımcı ya da özellik değerleri olarak düşünülebilir.

Yüksek bir C değeri, modele eğitim veri setindeki özellik verilerine yüksek ağırlık vermesini ve maliyet fonksiyonu (penalty) daha düşük bir ağırlık verilmesini işaret eder. Düşük bir değer ise, karmaşıklık cezasına daha fazla ağırlık verilmesini söyler. Temel olarak, yüksek bir C değeri, eğitim verilerindeki özelliklerin önemini daha çok arttıracaktır. C değeri, 0.00001 ile 1,000 arasında denenmiş ve optimum olarak 1.0 belirlenmiştir.

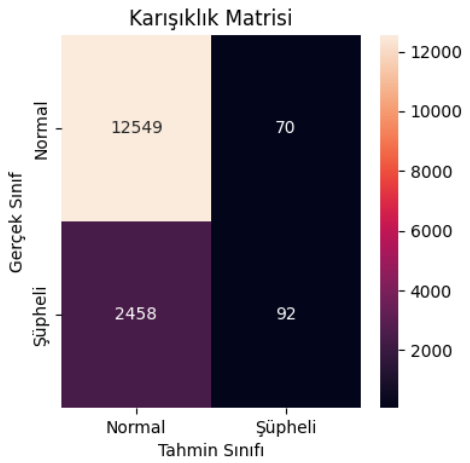
Diğer parametrelere bakılacak olursa, “random_state” veriyi karıştırmak için kullanılmaktadır. “Max_iter” parametresi, maksimum iterasyon sayısını ifade etmektedir. “Tol” parametresi, regresyonu durdurmak için olan tolerans değeridir.

```
71  # fitting
72  ss = StandardScaler()
73  x_train_scaled = ss.fit_transform(X_train)
74  x_test_scaled = ss.transform(X_test)
75  x_validation_scaled = ss.transform(X_validation)
76
77  LR = LogisticRegression(penalty='l2', C=1.0, verbose=0, random_state=12345, max_iter=1000, tol=0.0001)
78  LR.fit(x_train_scaled, Y_train)
79
80  real_pre = LR.predict(x_validation_scaled)
```

Şekil 4.8. Lojistik Regresyon Uygulaması Kod Bloğu

Uygulamanın toplam çalışma süresi ortalama 0.9 saniye sürmektedir. Doğruluk (accuracy) skoru 0.833, kesinlik (precision) skoru 0.791, duyarlılık (recall) skoru 0.833 ve F1 skoru 0.767 olarak bulunmuştur.

Şekil 4.9’taki karışıklık matrisi incelendiğinde, normal olan isteklerin yaklaşık %99 u normal olarak bulunmuş ancak 2550 anormal isteğin sadece 92 tanesi şüpheli olarak bulunmuştur. Bu da oran olarak yaklaşık %0.4 yapmaktadır. Sonuç olarak bu yöntemin, çalışmada kullanılan veri setinde çok da başarılı olduğu söylenemez.



Şekil 4.9. Lojistik Regresyon Karışıklık Matrisi

4.5. Destek Vektör Makinesi

Destek vektör makinesi (ya da destek vektör sınıflandırıcı) çalışma süresi, veri setindeki satır sayısıyla en azından ikinci dereceden ölçeklenir ve binlerce satırlık bir veri setinde pratik olmadığı bilinmektedir.

Hiper parametrelerden C değerine bakıldığında, değeri ne kadar büyürse işlem maliyeti (programın çalışma süresi) arttığı gözlenmektedir. C değeri 100 olduğunda ortalama çalışma süresi 180 saniye iken, 1000 olduğunda çalışma süresi ortalama 280 saniye olarak ölçülmüştür. Ortalama süre hesaplanırken, eğitim seti her defasında karıştırılarak, toplam 100 kere çalıştırılmıştır. Ancak bu maliyet artışı ile birlikte doğruluk oranı da yükselmiştir.

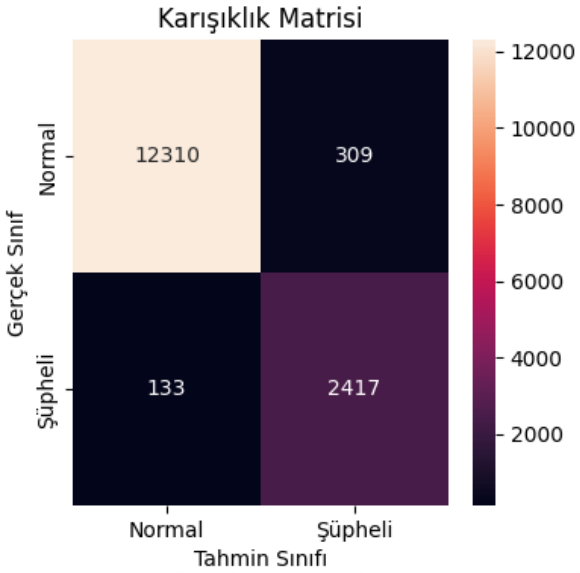
"Gamma" parametresi, kullanılan çekirdeğin, çekirdek sayısını ifade etmektedir. "Kernel" parametresi kullanılacak çekirdek metodunu ifade etmektedir. Şekil 4.10'daki kod bloğunda görünen "rbf" değeri, radyal tabanlı çekirdek (radial basis kernel) kullanımını ifade etmektedir. Veri setindeki özellik sayısı ikiden fazla olduğu için, yani karmaşık bir uzayda çalışıldığı için bu çekirdek metodu kullanılmaktadır.

```
121  SVMModel_Final = SVC(C=1000,
122                       gamma=10,
123                       kernel='rbf',
124                       random_state=0)
125
126  SVMModel_Final.fit(X_train, Y_train)
127  real_pre = SVMModel_Final.predict(X_validation)
```

Şekil 4.10. DVM Uygulaması Kod Bloğu

Uygulamanın toplam çalışma süresi ortalama 280 saniye sürmektedir. Ortalama süre hesaplanırken, eğitim seti her defasında karıştırılarak, toplam 100 kere çalıştırılmıştır. Doğruluk (accuracy) skoru 0.971, kesinlik (precision) skoru 0.972, duyarlılık (recall) skoru 0.971 ve F1 skoru 0.971 olarak bulunmuştur. Bu metrikler içinde en dikkat çeken değer, uygulamanın çalışma süresi olmuştur.

Skorlar genel olarak değerlendirildiğinde, sonuçlar başarılı görülmektedir. Ayrıca Şekil 4.11'teki karışıklık matrisine bakıldığında da, normal olan isteklerin %98'inin normal olarak bulunduğu, şüpheli işlemlerin ise yaklaşık %95'inin şüpheli olarak bulunduğu görülebilmektedir.



Şekil 4.11. DVM Karışıklık Matrisi

4.6. Otomatik Kodlayıcı

Bir otomatik kodlayıcı, bir kodlayıcı ve bir kod çözücü alt modellerinden oluşur. Kodlayıcı girişi sıkıştırır ve kod çözücü, kodlayıcı tarafından sağlanan sıkıştırılmış sürümden girişi yeniden oluşturmaya çalışır.

Kodlayıcıyı, Şekil 4.12'deki kod bloğunda görüleceği üzere iki gizli katmana sahip olacak şekilde tanımlanmıştır, birincisi iki kat daha fazla girdi (12) ve ikincisi aynı sayıda girdi (6) ile, ardından darboğaz (bottleneck) katmanıdır. Darboğaz katmanı 4 girdiden oluşmaktadır. Eğitim kısmı ise Şekil 4.13'te görülmektedir.

```
# define encoder
visible = tf.keras.layers.Input(shape=(n_inputs,))

# encoder level 1
e = tf.keras.layers.Dense(n_inputs*2)(visible)
e = tf.keras.layers.BatchNormalization()(e)
e = tf.keras.layers.LeakyReLU()(e)

# encoder level 2
e = tf.keras.layers.Dense(n_inputs)(e)
e = tf.keras.layers.BatchNormalization()(e)
e = tf.keras.layers.LeakyReLU()(e)

# bottleneck
# n_bottleneck = round(float(n_inputs) / 2.0)
n_bottleneck = 4
bottleneck = tf.keras.layers.Dense(n_bottleneck)(e)


# define decoder, level 1
d = tf.keras.layers.Dense(n_inputs)(bottleneck)
d = tf.keras.layers.BatchNormalization()(d)
d = tf.keras.layers.LeakyReLU()(d)

# decoder level 2
d = tf.keras.layers.Dense(n_inputs*2)(d)
d = tf.keras.layers.BatchNormalization()(d)
d = tf.keras.layers.LeakyReLU()(d)

# output layer
output = tf.keras.layers.Dense(n_inputs, activation='linear')(d)
```

Şekil 4.12. Otomatik Kodlayıcı Uygulamasındaki Katmanlar

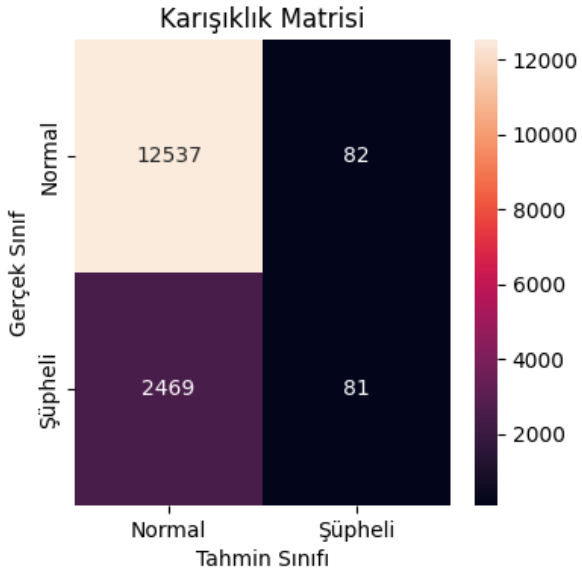
```
# fit the autoencoder model to reconstruct input
history = model.fit(
    normal_train_data,
    normal_train_data,
    epochs=500,
    batch_size=128,
    verbose=2,
    validation_data=(X_test, X_test)
)
```

Şekil 4.13. Otomatik Kodlayıcı Uygulamasındaki Eğitim Aşaması

Uygulamanın toplam çalışma süresi ortalama 1,300 saniye sürmektedir. Ortalama süre hesaplanırken, eğitim seti her defasında karıştırılarak, toplam 100 kere çalıştırılmıştır. Doğruluk (accuracy) skoru 0.832, kesinlik (precision) skoru 0.779,

duyarlılık (recall) skoru 0.832 ve F1 skoru 0.765 olarak bulunmuştur. Bu metrikler içinde en dikkat çeken değer, uygulamanın çalışma süresi olmuştur. Sonuç çok da başarılı gözükmediği halde, çalışma süresi oldukça uzun sürmüştür. Özellikle çalışma süresinin bu kadar uzun oluşu, bu tarz bir uygulama için pek de uygun olmadığını göstermektedir.

Genel skorlar değerlendirildiğinde başarılı denebilecek durumda olduğu halde, Şekil 4.14'teki karışıklık matrisi incelendiğinde, normal olan isteklerin yaklaşık %99'u normal olarak bulunmuş ancak 2550 anormal isteğin sadece 92 tanesi şüpheli olarak bulunmuştur. Bu da oran olarak yaklaşık %0.3 yapmaktadır. Sonuç olarak bu yöntemin, çalışmada kullanılan veri setinde başarılı olduğu söylenemez.



Şekil 4.14. Otomatik Kodlayıcı Karışıklık Matrisi

5. TARTIŞMA

Benzer araştırmalarda kullanılan, açık kaynaklı veri setleri simülasyon ortamı için ideal olsa da, pratikte doğru çıktılar üretemeyeceği açıktır. 2019 yılında yapılan bir çalışma incelendiğinde, eğitim veri setinde 67,343 adet normal istek, 45,927 adet saldırı türünde etiketlenmiş istek verisi bulunmaktadır (Khorram T. 2019). Sınıf dengesizliğinin az olması problemin çözümünü kolaylaştırmaktadır. Oysa ki gerçek hayatta beklenen, anomali durumlarının normal durumlardan çok daha az olmasıdır. Model oluşturma aşamasında, bu beklentiye uygun bir şekilde, sınıf dengesizliğinin çok daha fazla olduğu durumlar da değerlendirilmelidir. Bahsi geçen araştırmada, destek vektör makinesi ile yapılan denemenin doğruluk değeri %99'dur. Bu çalışmada ise destek vektör makinesi ile yapılan denemenin sonucu %97'dir. Ancak biraz önce de belirtildiği gibi, bu çalışmadaki gerçek bir web sitesinin kullanıcı erişim kayıtları kullanılmıştır, verideki sınıf dengesizliği oldukça fazladır.

Sınıf dengesi düzgün olan, açık kaynaklı bir kullanıcı istek günlüğü verisi içeren başka bir çalışmada, bu çalışmada da olduğu gibi karar ağacı yöntemi kullanılmıştır (Kotan B. 2019). Yapılan denemedeki doğruluk skoru %97.7 olarak bulunmuştur. Bu çalışmada ise "AdaBoost" yöntemi ile karar ağacı metodu güçlendirilmiş, sınıf dengesizliği çok yüksek olduğu halde doğruluk değeri %99.9 olarak bulunmuştur.

Yapılan literatür araştırmalarının tümünde açık kaynaklı erişime sahip NSL-KDD veri seti kullanılmıştır. Ayrıca yapılan çalışmaların çoğunda eğitim verisi hazırlanırken, sınıf dengesini oluşturacak şekilde bir miktar normal istek ve aynı oranda saldırı olarak etiketlenmiş veri bulunmaktadır. Anomali olarak nitelendirilen her durum, normal duruma göre daha az sayıda olmalıdır. Aksi, kelime anlamına bile zıt düşmektedir. Bu şekilde simülasyon ortamındaki bütün şartlar en ideal durumda olduğundan sonuçlar da daha başarılı ve öngörülebilir olacaktır. Aslında simülasyon ortamının, gerçek ortama oldukça fazla benziyor olması beklenmektedir.

6. SONUÇLAR

Uygulamalar karşılaştırıldığında, minimum doğruluk oranının %82.8, maksimum doğruluk oranının ise yaklaşık %100 olduğu görülmektedir. Sadece doğruluk skorları üzerinden başarı ya da başarısızlık analizi bu çalışmadaki problemin çözümü olmayacaktır.

Veri setindeki normal/şüpheli istek dağılımı Çizelge 6.1'de görüldüğü gibidir. Bu çalışma bir sınıflandırma problemi olarak değerlendirilmiştir ve tüm uygulamalar bu fikirle geliştirilmiştir. Çizelge 6.1'de, veri seti üzerindeki sınıf dağılımının düzensiz olduğu gözle görülmektedir. Bu sebeple doğruluk skorları, başarı ya da başarısızlık anlamında çok da belirleyici olmayacaktır. Örnek verilecek olursa; tüm isteklerin normal olduğunu bulan bir sınıflandırıcı modeli yaklaşık olarak %83 doğru işlem yapmış olacaktır. Ancak aslında, şüpheli isteklerin hiçbirini bulamamış da olabilir.

Çizelge 6.1. İsteklerin Dağılımları

Normal İstek	12,619	%83
Şüpheli İstek	2,550	%17

Çizelge 6.2'de sınıflandırmaların gerçek pozitif/yanlış pozitif ve gerçek negatif/yanlış negatif istek adetleri verilmiştir. Şüpheli işlemlerin bir kısmının tespit edilip, bir kısmının tespit edilemediğinde, belli bir oranda başarı elde edilebildiği söylenebilir. Ancak normal olan isteklerin, şüpheli istek olarak tespit edilmesi ve bunun oranının yüksek olması gerçek bir uygulamada kabul edilemez. Çizelgeye bakılarak, bazı uygulamaların belirgin şekilde normal olan istekleri, şüpheli olarak tespit ettiği görülmektedir. Bunlardan en göze çarpan saf Bayes Sınıflandırıcı, normal olan isteklerin yaklaşık %0.4'ünü şüpheli olarak bulmuştur. Destek Vektör Makinesi yönteminde de benzer bir sonuç elde edilmiştir

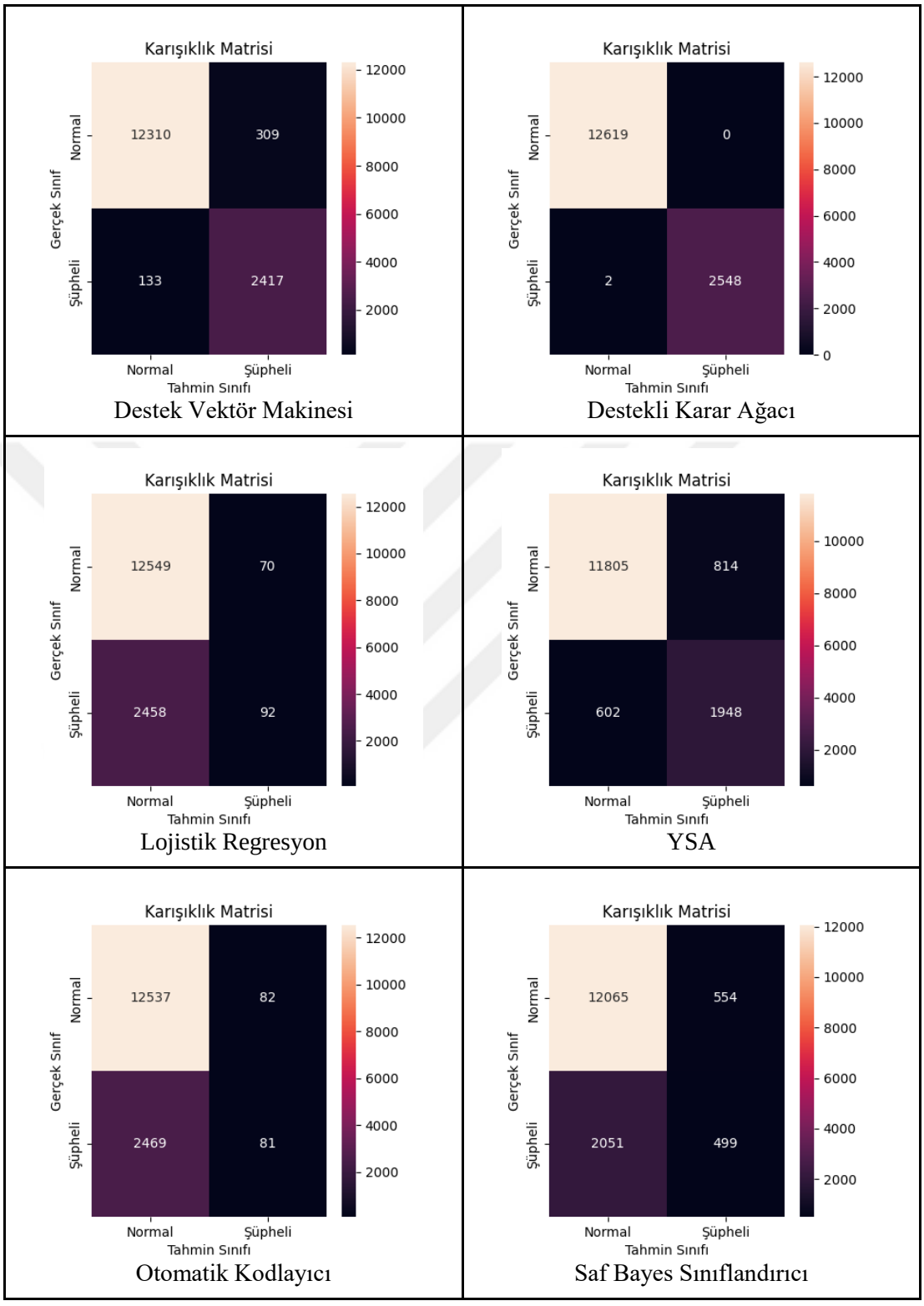
Çizelge 6.2. Yöntemlerin Normal ve Şüpheli Tahmin Adetleri

	DVM	Karar Ağacı	Lojistik Regresyon	YSA	Otomatik Kodlayıcı	Saf Bayes
Gerçek Normal (TP)	12310	12619	12549	11805	12537	12065
Yanlış Normal (FP)	133	2	133	602	2469	2051
Gerçek Şüpheli (TN)	2417	2548	92	1948	81	499
Yanlış Şüpheli (FN)	309	0	70	814	82	554

Normal olan isteklerin şüpheli ya da şüpheli olan isteklerin normal olarak tespiti, radar problemine benzemektedir. Bir hava savunma sistemine ait radarının, alarm durumuna geçmemesi gerekirken geçmesi ne kadar kabul edilemezse, normal bir kullanıcının yaptığı isteklerin engellenmesi de benzer şekilde kabul edilemez. Aynı şekilde alarm durumuna geçmesi gerekirken geçmemesi ve saldırının önlenemesi de, şüpheli işlem yapan bir kullanıcının normal olarak tespit edilmesine çok benzemektedir.

Çizelge 6.3'te tüm yöntemlerin karışıklık matrisleri bir arada verilerek görsel olarak karşılaştırma imkanı sağlanmıştır. En büyük başarı kriteri, yanlış normallerin en az olması ve doğru şüphelilerin en yüksek olması durumudur.

Çizelge 6.3. Karışıklık Matrislerinin Toplu Olarak Değerlendirilmesi



Uygulama skorlarının ve çalışma sürelerinin yer aldığı Çizelge 6.4'e bakacak olursak, aralarındaki en bariz fark çalışma süreleri olacaktır. Bazı uygulamalar saniyeler mertebesinde çalışırken, bazıları ise dakikalar mertebesinde çalışmıştır. Bu

çalışmanın amacı web isteklerindeki anomali durumlarının anlık ya da anlığa çok yakın seviyelerde doğru olarak tespit edilmesidir. Bu yüzden dakikalar seviyesinde işlem süresine sahip destek vektör makinesi ve otomatik kodlayıcı yöntemlerinin bu tarz bir problemin çözümü için çok da uygun olmayacağı söylenebilir. Pratik olarak da, özellikle otomatik kodlayıcı küçük veri setleri için daha uygun olduğu söylenebilir.

Ortalama süre hesaplanırken, eğitim seti her defasında karıştırılarak, toplam 100 kere çalıştırılmıştır.

Çizelge 6.4. Uygulama Sonuçlarının Karşılaştırılması

Metrik / Uygulama	DVM	Karar Ağacı	Lojistik Regresyon	YSA	Otomatik Kodlayıcı	Saf Bayes
Doğruluk (accuracy)	0.971	0.999	0.833	0.907	0.832	0.828
Kesinlik (Precision)	0.972	0.999	0.791	0.910	0.779	0.791
Duyarlılık (Recall)	0.971	0.999	0.833	0.907	0.832	0.828
F1 Skoru	0.971	0.999	0.767	0.908	0.765	0.797
Çalışma Süresi (sn)	286	1.4	1.3	166	1,300	3.8

Tüm sonuçlar değerlendirildiğinde, en başarılı uygulamanın Destekli Karar Ağacı uygulaması olduğu kolaylıkla söylenebilir. Başarı oranı %100'e çok yakın olmakla beraber, kullanılan veri seti üzerindeki çalışma süresi ortalama 1.4 saniyedir. Ayrıca hiçbir normal olan isteği şüpheli olarak sınıflandırmamış ve tüm şüpheli işlemlerden yalnızca 2 tanesini normal olarak sınıflandırmıştır.

Destek Vektör Makinesi her ne kadar Destekli Karar Ağacı uygulamasına yakın sonuçlar vermiş olsa da çalışma süresi dakika mertebesinde olduğundan, gerçek bir uygulamada, benzeri bir veri seti üzerinde tercih edilmeyecektir.

7. ÖNERİLER

Bu çalışmada kullanılan veri seti belirli bir zaman aralığına aittir. Bu zaman aralığının seçimi sonuçların doğruluğunu ve işlem süresini direkt olarak etkilemektedir. Zaman aralığının küçük olarak seçilmesi, örneklem sayısının azalmasına sebep olacaktır. Büyük seçilmesi ise bariz olarak işlem süresini arttıracaktır. Ayrıca modelin eğitimi için bolca gözlem gerekmektedir.

İdeal olarak yapılması gereken şey, işlem zamanı uzun dahi olsa, büyük bir zaman aralığında, bol gözlemli bir veri seti kullanarak, belirli aralıklarla modelin eğitimini güncellemektir.

Modelin eğitimini uzun süren farklı bir iş olarak düşündüğümüzde, oluşturulan modelin kullanım maliyeti (yani işlem zamanı) çok düşük olacaktır. Örnek verecek olursak; model her hafta 1 kere eğitilir, eğitilmiş model her 5 dakikalık erişim kayıtları üzerinde çalıştırılır. Anomali olarak çıkan sonuçlardaki IP adresi, bir yöntem ile direkt olarak engellenebilir ya da bir gözlemciye rapor edilebilir. Gözlemci olması durumunda, daha sonraki eğitim süreci için daha doğru örneklemler elde edilebilir.

Anomali tespit işlemi herhangi bir web isteği ile doğrudan çalıştırılmamalıdır. Bir web isteğinin milisaniyeler seviyesinde, olabildiğince hızlı olarak cevap vermesi beklenir. Bu tarz kontroller, istekten bağımsız olarak, ayrı bir işlem şeklinde çalışmalıdır.

8. KAYNAKLAR

Aggarwal C.C. (2018). Neural Networks and Deep Learning: A Textbook, Springer

Akpınar H. (2014). DATA - Veri Madenciliği Veri Analizi, Papatya Bilim

Al-Bayatı, T. A. (2019). Anomaly Detection in Time Series. Kadir Has Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul (Danışman: Prof. Dr. Arif Selçuk Öğrenci)

Bienias P., Kolaczek G., Warzynski A. (2019) Architecture of Anomaly Detection Module for the Security Operations Center. 28th International Conference on Enabling Technologies.

Boser B.E., Guyon I.M., Vapnik V.N. (1992). A Training Algorithm for Optimal Margin Classifiers

Coşan S. (2020). Payload-Based Network Intrusion Detection Using LSTM Autoencoders. Bilkent Üniversitesi Mühendislik ve Bilim Enstitüsü, Yüksek Lisans Tezi, Ankara (Danışman: Süleyman Serdar Kozat)

Çintir G. (2020). Deep Neural Network-Based Anomaly Detection System on Multivariate Time-Series Data. Bahçeşehir Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul (Danışman: Prof. Dr. Cemal Okan Şakar)

Harrell F.E. (2015). Regression Modeling Strategies, Springer

Karadayı Y. (2020). A Hybrid Deep Learning Framework for Unsupervised Anomaly Detection in Multivariate Spatio-Temporal Data. Kadir Has Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi, İstanbul (Danışman: Prof Dr. Mehmet N. Aydın)

Kbar G., Alazab A. (2019) A Comprehensive Protection Method for Securing the organization's network against Cyberattacks. Cybersecurity and Cyberforensics Conference.

Khorram T. (2019). Network Anomaly Detection Using Optimized Machine Learning Algorithms. T.C. Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Konya (Danışman: Prof. Dr. Nurdan Baykan)

Kim Y., Kim H.K. (2020) Anomaly Detection using Clustered Deep One-Class Classification. 15th Asia Joint Conference on Information Security.

Komazec T., Gajin S. (2019) Analysis of flow-based anomaly detection using Shannon's entropy. 27th Telecommunications forum.

Kotan B. (2019). Network Monitoring System Machine Learning Comparative Analysis of Classification Techniques for Network Traffic Monitoring. T.C. Hasan Kalyoncu Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul (Danışman: Prof. Dr. Mohammed K. M. Madi)

Öney M. U. (2019). A Comparative Study of Neural Network Approaches in Network Anomaly Detection. Atılım Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Ankara (Danışman: Prof. Dr. Serhat Peker)

Maimon O.Z., Rokach L. (2007). Data Mining with Decision Trees, World Scientific

Merrill N., Eskandarian A.A. (2020) Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning.

Ng J., Joshi D., Banik M.S. (2015) Applying data mining techniques to intrusion detection. 12th International Conference on Information Technology.

Poyraz O. (2016). Anomaly Detection in Time Series. Boğaziçi Üniversitesi Fen Bilimleri ve Mühendislik Lisansüstü Eğitim Fakültesi, Yüksek Lisans Tezi, İstanbul (Danışman: Prof. Dr. Ali Taylan Cemgil)

Sağında B. (2020). Deep Learning Based Log Anomaly Detection With Time Differences. Çankaya Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Ankara (Danışman: Prof. Dr. Roya Choupani)

Soman P. K. (2009). Machine Learning with SVM and Other Kernel Methods, Prentice-Hall of India

Sunny J., Sankaran S., Saraswat V. (2020) A Hybrid Approach for Fast Anomaly Detection in Controller Area Networks. 2020 IEEE International Conference on Advanced Networks and Telecommunications Systems.

Tan C. C., Eswaran C. (2010). Autoencoder Neural Networks, LAP Lambert Academic Publishing

Ullah I, Mahmoud Q.H. (2020) A Technique for Generating a Botnet Dataset for Anomalous Activity Detection in IoT Networks. IEEE International Conference on Systems, Man, and Cybernetics.

Upman V., Goranin N. (2020) Investigation of RBFN Application for Anomaly-Based Intrusion Detection on IoT Networks.

Vural G. (2006). Anomaly Detection From Personal Usage Patterns In Web Applications. Ortadoğu Teknik Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Ankara (Danışman: Prof. Dr. Meltem Turhan Yöndem)

Yang X., Zhou L., Liu Y. (2021) An anomaly detection method for daily line loss rate based on deep autoencoder. 2021 IEEE International Conference on Power Electronics, Computer Applications.

İnternet Kaynakları

Url-1 <https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/>, alındığı tarih: 27.11.2021.

Url-2 <https://www.tibco.com/reference-center/what-is-a-neural-network>, alındığı tarih: 29.11.2021.

Url-3 <https://en.wikipedia.org/wiki/Sigmoid_function>, alındığı tarih: 03.11.2021.

Url-4 <https://ai-pool.com/a/s/decision-trees>, alındığı tarih: 03.11.2021.

Url-5 <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>, alındığı tarih: 16.11.2021.

Url-6 <https://medium.com/@k.ulgen90/makine-%C3%B6%C4%9Frenimi-b%C3%B6l%C3%BCm-4-destek-vekt%C3%B6r-makineleri-2f8010824054>, alındığı tarih: 16.11.2021.

Url-7 <https://medium.com/autoencoder-for-anomaly-detection/autoencoder-for-anomaly-detection-db6178ad07b2>, alındığı tarih: 03.11.2021.

9. ÖZGEÇMİŞ

İlk, orta ve lise eğitimini tamamladıktan sonra lisans eğitimini T.C. Haliç Üniversitesinde Bilgisayar Mühendisliği bölümünde aldı. Halen T.C. Haliç Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim dalı Tezli Yüksek Lisans öğrencisi olarak devam etmektedir. 2008 yılında başladığı serbest yazılım geliştirme sürecine daha sonra kendi firmasını kurarak devam etti. 2014 yılından itibaren ise başta Digitürk olmak üzere, çeşitli şirketlerde çalışmaya devam etti. Bugün ise ING Bank Türkiye'de, Uzman Yazılım Danışmanı olarak iş hayatına devam etmektedir. 2022 yılı itibari ile toplamda 12 yıllık sektör tecrübesi bulunmaktadır.