KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

APACHE SPARK VE MAKİNE ÖĞRENMESİ ALGORİTMALARI İLE AĞ SALDIRISI TESPİTİ

ELİF MERVE KURT

KOCAELİ 2019

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

APACHE SPARK VE MAKİNE ÖĞRENMESİ ALGORİTMALARI İLE AĞ SALDIRISI TESPİTİ

ELİF MERVE KURT

Prof.Dr. Yaşar BECERİKLİ
Danışman, Kocaeli Üniversitesi .....

Doç.Dr. Ahmet SAYAR
Jüri Üyesi, Kocaeli Üniversitesi .....

Prof.Dr. Nejat YUMUŞAK
Jüri Üyesi, Sakarya Üniversitesi .....

Tezin Savunulduğu Tarih: 24.01.2019

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması, makine öğrenmesi algoritmalarının büyük ağ verileri üzerinde yer alan saldırıları tespit etmedeki performanslarını incelemek ve karşılaştırmalı olarak analiz etmek amacıyla gerçekleştirilmiştir.

Tez çalışmamda benden maddi manevi desteğini hiçbir zaman esirgemeyen ve başaracağıma her zaman inanan danışmanım sayın Prof.Dr. Yaşar BECERİKLİ'ye sonsuz teşekkürlerimi sunarım.

Yüksek lisans öğrenimime başlamamda beni yüreklendiren ve bana güvenen hocam sayın Doç.Dr. Sevinç İLHAN OMURCA'ya teşekkürlerimi sunarım.

Yüksek lisans öğrenimim boyunca görüşleri ile çalışmalarıma katkıda bulunan, karşılaştığım her zorlukta desteğini ve zamanını esirgemeyen hocalarım Arş.Gör. Fidan KAYA GÜLAĞIZ ve Arş.Gör. Ekin EKİNCİ'ye teşekkürlerimi sunarım.

Tez çalışmamda gösterdikleri anlayış ve destek için CTech Bilişim Teknolojileri genel müdürü sayın Dr. Cüneyd FIRAT'a, arkadaşlarım Hüseyin YENER ve Kevser Betül KALYON başta olmak üzere tüm CTech ailesine teşekkürü borç bilirim.

Tez sürecim boyunca yanımda olan ve beni her daim destekleyen, benden bilgi ve tecrübelerini eksik etmeyen değerli arkadaşlarım Hayrunnisa SARI, Seda KUL, Mine TÜTÜNCÜ ve Yasemin KAYA'ya teşekkürlerimi sunarım.

Lisans ve yüksek lisans öğrenimim boyunca benden maddi manevi desteğini esirgemeyen kuzenim Ayşe ARSLAN AYBEK'e teşekkürü borç bilirim.

Ailemden uzak oluşumu hissettirmemeye çalışan, her sıkıntımda her mutluluğumda yanımda olan kıymetli komşum Nimet YILMAZ ÇAM'a ve ailesine maddi manevi destekleri için sonsuz teşekkürlerimi sunarım.

Tez teslim sürecim boyunca her zaman yanımda olan, benden sevgisini ve desteğini bir an bile eksik etmeyen sevgili eşim İbrahim Semih KURT'a teşekkürlerimi sunarım.

Hayatım boyunca bana güç veren, en büyük destekçilerim olan, hayatımın her aşamasında sıkıntılarımı ve mutluluklarımı paylaşan sevgili babam Ali TÜTÜNCÜ, annem Fikriye TÜTÜNCÜ, abim Yunus Emre TÜTÜNCÜ, kuzenim Kübra ATAY, arkadaşlarım Merve GÜLAY ve Şelale Gamze ÖZTÜRK'e teşekkürlerimi borç bilirim.

Ocak – 2019 Elif Merve KURT

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

API	:	Application Programming Interface (Uygulama Programlama Arayüzü)
BKB	:	Bire Karşı Bir
BKD	:	Bire Karşı Diğer
CART	:	Classification and Regression Tree (Sınıflandırma ve Regresyon Ağacı)
CGI	:	Common Gateway Interface (Genel Geçiş Arayüzü)
CPU	:	Central Processing Unit (Merkezi İşlem Birimi)
DARPA	:	Defense Advanced Research Projects Agency (Savunma İleri Araştırma Projeleri Ajansı)
DoS	:	Denial of Service Attack (Servisi Reddetme Saldırıları)
DVM	:	Destek Vektör Makineleri
FTP	:	File Transfer Protocol (Dosya Transfer Protokolü)
HDFS	:	Hadoop Distributed File System (Hadoop Dağıtık Dosya Sistemi)
HTTP	:	Hyper-Text Transfer Protocol (Hiper-Metin Transfer Protokolü)
ICMP	:	Internet Control Message Protocol (İnternet Kontrol Mesaj Protokolü)
IMAP	:	Internet Message Access Protocol (İnternet Mesaj Erişim Protokolü)
IoE	:	Internet of Everything (Her Şeyin İnterneti)
IP	:	Internet Protocol (İnternet Protokolü)
KDD	:	Knowledge Discovery in Databases (Veri Tabanı Bilgi Keşfi)
LAN	:	Local Area Network (Yerel Alan Ağları)
MAN	:	Metropolitan Area Network (Metropolitan Ağlar)
Netstat	:	Network Statistics (Ağ İstatistikleri)
Nmap	:	Network Mapper (Ağ Eşleyicisi)
OOB	:	Out of Bag (Torba Dışı)
OR	:	Odds Ratio (Odds Oranı)
OS	:	Operating System (İşletim Sistemi)
PS	:	Process Status (İşlem Durumu)
R2L	:	Remote to Local Attack (Yerele Uzaktan Saldırılar)
SQL	:	Structured Query Language (Yapısal Sorgulama Dili)
SU	:	Substitute User (Yedek Kullanıcı)
SYN	:	Synchronization (Senkronizasyon)
TCP	:	Transmission Control Protocol (Aktarma Kontrol Protokolü)
U2R	:	User to Root Attack (Root Kullanım Saldırıları)
UDP	:	User Datagram Protocol (Kullanıcı Veribloğu İletişim Kuralları)
URL	:	Uniform Resource Locator (Birörnek Kaynak Konumlayıcı)
WAN	:	Wide Area Network (Geniş Alan Ağları)

APACHE SPARK VE MAKİNE ÖĞRENMESİ ALGORİTMALARI İLE AĞ SALDIRISI TESPİTİ

ÖZET

İnternet tabanlı hizmetlerin sürekli olarak artış göstermesi, ağ trafik verilerini günden güne daha büyük ve karmaşık hale getirmektedir. Bu durum, ağ saldırılarının tespitini giderek zorlaştırmakta dolayısıyla ağ güvenliğinin sağlanması için daha etkili ve hızlı veri işleme yöntemlerini gerekli kılmaktadır. Bu amaçla pek çok saldırı tespit sistemi geliştirilmiştir ve geliştirme çalışmaları devam etmektedir.

Bu çalışmada amaçlanan, makine öğrenmesi algoritmalarının aynı ağ verileri üzerindeki performanslarını karşılaştırarak geliştirilmekte olan saldırı tespit sistemlerine referans kaynak oluşturmaktır. Çalışmada; büyük veri teknolojisi Apache Spark kullanılarak KDD Cup'99 verilerinin tamamı makine öğrenmesi algoritmalarından Lojistik Regresyon, Destek Vektör Makineleri, Naive Bayes ve Rastgele Orman üzerinde koşturulmuş; sonuçlar karşılaştırmalı olarak analiz edilmiştir. Elde edilen sonuçlar, Apache Spark teknolojisinin büyük ağ verileri üzerindeki saldırıları tespit etmede giderek daha etkili olduğunu göstermiştir.

Anahtar Kelimeler: Apache Spark, Büyük Veri, KDD Cup'99, Makine Öğrenmesi, Saldırı Tespit Sistemi.

NETWORK INTRUSION DETECTION ON APACHE SPARK WITH MACHINE LEARNING ALGORITHMS

ABSTRACT

The continuous increase in internet-based services makes network traffic data larger and more complex day by day. This makes it increasingly difficult to detect network attacks, and therefore requires more efficient and faster data processing methods to ensure network security. For this purpose, many intrusion detection systems have been developed and development works are continuing.

This study; by comparing the performance of machine learning algorithms on the same network data, aims to establish a reference source for the developed intrusion detection systems. In this study; all data of KDD Cup'99 were run on Logistic Regression, Support Vector Machine, Naive Bayes and Random Forest from machine learning algorithms using Apache Spark a big data technology; and the results were analyzed comparatively. The obtained results show that the Apache Spark technology has become increasingly effective in detecting attacks on big network data.

Keywords: Apache Spark, Big Data, KDD Cup'99, Machine Learning, Intrusion Detection System.

GİRİŞ

Bilgisayar teknolojilerinin günden güne artış göstermesiyle bilgisayar özellikleri giderek iyileşmekte, böylelikle bilgisayarlar ve bilgisayar ağları hızlanmaktadır. Bu gelişimin sonucu olarak, bilgisayar kullanım oranında büyüme meydana gelmekte; dolayısıyla internet erişimi sağlayanların sayısı da giderek artmaktadır. Bu durum, internet tabanlı hizmetlerin gelişimini zorunlu hale getirmektedir.

İnternet tabanlı hizmetlerin gelişimi ile birlikte çok büyük bir hal alan ağ trafik verileri, yönetim açısından giderek karmaşıklaşmaktadır. Bu durum, ağ saldırılarının tespitini zorlaştırarak ağ güvenliğini tehdit etmektedir.

Tehdidi ortadan kaldırmak için korunma amaçlı daha etkili ve hızlı veri işleme yöntemleri geliştirildikçe saldırganlar da sürekli yeni ataklar ortaya çıkarma yolunda ilerlemişlerdir. Bu döngüsel durum, ağ saldırılarını araştırmacıların ilgi odağı haline getirmiştir.

1980'lerden bu yana gelişmekte olan ve çeşitli öğrenme algoritmaları ile birlikte kullanılarak farklı şekillerde oluşturulan ağ saldırısı tespit sistemleri, iki ana başlıkta incelenebilir [1]:

- Bilgi Bazlı: Yalnızca öğrendiği saldırıları yakalayabilen sistemlerdir.
- Davranış Bazlı: Davranışların normal hallerini öğrenen, öğrendiklerinin dışında kalanları saldırı olarak tanımlayan sistemlerdir.

Sürekli değişim ve gelişim içinde olan saldırıları tespit edebilmek için hem daha önce karşılaşıp öğrendiği hem de ilk kez karşılaştığı saldırıları yakalayabilecek sistemler geliştirilmek istenmiştir. Bu amaçla yapılan çalışmalar sonucu sınıflandırma tabanlı ağ saldırısı tespit sistemleri ortaya çıkmıştır. Örüntü sınıflama ile bilgi ve davranış bazlı sistem mantığı birlikte kullanılarak, saldırılar daha etkili bir şekilde yakalanabilmektedir. İki ana metodu vardır [2]:

- Eğiticili Sınıflama: Bu tür sınıflamada makineden beklenen; problem hakkında bilgi sahibi olması için kendisine verilen, problemin bir çeşit örneğini içeren veri

setinden gerekli bilgileri öğrenmesi ve yeni verilerle karşılaştığında önceden öğrendiği bilgileri kullanarak çıkarım yapmasıdır. Yani bir diğer deyişle makine eğitilir ve gerektiğinde öğrendiği bilgileri kullanması beklenir. Bu sınıflama türünde, veri seti hem özellik hem de sınıf bilgisi içerir.

- Eğiticisiz Sınıflama: Bu tür sınıflamada makine, problemi tanımlayıcı bilgi ile doğrudan karşılaşmaz. Makineden, verideki benzer özellikleri belirleyerek çıkarım yapması beklenir. Yani bir diğer deyişle makine eğitilmez, kendi kendini eğitir. Bu sınıflama türünde veri seti sadece özellik bilgisi içerir, sınıf bilgisi içermez.

Literatürdeki saldırı tespit sistemi çalışmaları incelenecek olursa; örüntü sınıflamada kullanılan makine öğrenmesi algoritmaları, algoritma parametreleri, deneysel veri setleri veya sınıflandırma çeşitleri (ikili veya çoklu) gibi değişimler olsa da bu çalışmada tasarlanan sisteme benzer birçok çalışmaya rastlanır.

Govind P Gupta, Manish Kulariya çalışmalarında [3], hızlı ağ trafiği verilerini daha çabuk analiz edebilmek ve saldırı trafiğini daha çabuk yakalayabilmek için büyük veri işleme aracı olan Apache Spark'ı kullanmışlardır. Çalışmada, nitelikli özelliklerin çıkarımı için iyi bilinen özellik seçme algoritmalarından faydalanılmış olup çıkarılan bu nitelikli özellikler sınıflandırma tabanlı yöntemlerin kullanıldığı ağ saldırı tespit sistemine veri olarak sürülmüştür. Bahsi geçen sınıflandırma tabanlı yöntemler, iyi bilinen makine öğrenmesi algoritmalarından olan; Lojistik Regresyon, Destek Vektör Makineleri, Rastgele Orman, Gradient Boosted Karar Ağacı Algoritmaları ve Naive Bayes'tir. Gerçek zaman verisi olan DARPA'nın verilerinden oluşan KDD Cup'99 veri seti, çalışmada önerilen sınıflandırma tabanlı yöntemler için deneysel veri seti olarak kullanılmıştır. Saldırı tespit yöntemi sonuçları doğruluk, eğitim ve tahminleme süreleri açısından değerlendirilmiştir.

Kamran Siddique, Zahid Akhtar, Haeng-gon Lee, Woongsup Kim ve Yangwoo Kim çalışmalarında [4] yüksek hızlı büyük veri ağlarında, paralel işleme tabanlı makine öğrenmesi teknikleri ile anormallik tespiti yapmaya çalışmışlardır. Kapsamlı araştırmalar sonucunda gerçekleştirdikleri çalışmalarında, özellik seçiminin ardından sınıflandırma yapmak için Lojistik Regresyon ve Gradient Boosted Karar Ağacı Algoritmalarını kullanmışlar ve deneysel sonuçlarını sunmuşlardır.

Sasan Harifi, Ebrahim Byagowi ve Madjid Khalilian çalışmalarında [5], büyük ağ verilerini işlemede tekrarlama gerektiren makine öğrenmesi algoritmaları için Apache Spark'ın oldukça uygun bir araç olduğunu vurgulamışlar ve Apache Spark'ın makine öğrenmesi algoritmaları kütüphanesine genel bir bakış sunmuşlardır. Forest Cover Type, KDD Cup'99 ve deneyler için kullanılan İnternet Reklamcılığı verilerini Apache Spark kütüphanesinin sunduğu farklı makine öğrenmesi algoritmaları üzerinde koşturmuşlar ve elde ettikleri sonuçları raporlamışlardır.

Hae-Duck J. Jeong, Gil-Seong Jeong, Won-Jung Kim, Jinwon Kim, Hanbin Song, Myeong-Un Ryu, Jongsuk R. Lee çalışmalarında [6], büyük veri trafiğine çözüm olması amacıyla bilinen beş makine öğrenmesi algoritmasını kullanarak bir saldırı tespit sistemi tasarlamışlar ve tasarladıkları sistemi doğrulamak için KDD Cup'99 veri setini kullanmışlardır. Sistemleri ağ verilerine ait bilgilerin öğrenilmesi ve öğrenilen bilgilerle yeni gelen ağ verilerinin normal ya da saldırı olarak sınıflandırılması mantığı üzerine kurulmuştur. Doğruluk açısından Stokastik Gradyan Descent kullanarak Destek Vektör Makineleri ve Lojistik Regresyon algoritmalarını, işlem süresi açısından ise Rastgele Orman ve Karar Ağacı Algoritmalarını en iyi olarak raporlamışlardır.

Se Won Oh, Hyeon Soo Kim, Ho Sung Lee, Sun Jin Kim, Hongkyu Park, Woongshik You, Apache Spark'a dayalı geliştirdikleri sistemlerinde [7], büyük IoE ağı verileri ile makine öğrenmesi algoritmalarını kullanmışlardır. Geliştirdikleri sistemi, KDD Cup'99 veri setini kullanarak deneyler yoluyla doğrulamışlardır.

Bu çalışmada; deneysel veri seti olarak büyük ağ verisi KDD Cup'99 seçilmiştir. Verilerin daha hızlı işlenmesi amacıyla, büyük veri işleme teknolojisi olan Apache Spark kullanılmıştır. Eğiticili sınıflama yöntemlerinden Lojistik Regresyon, Destek Vektör Makineleri, Naive Bayes ve Rastgele Orman makine öğrenmesi algoritmaları kullanılarak bilgi bazlı ağ saldırısı tespit sistemleri oluşturulmuştur. KDD Cup'99 verilerinin tamamı ile makine öğrenmesi algoritmaları üzerinde parametre değişimleri yapılarak hız ve doğruluk açısından testler yapılmış; algoritmalar süre ve doğruluk yönünden en optimize sonuçları veren parametreleri ile raporlanmıştır.

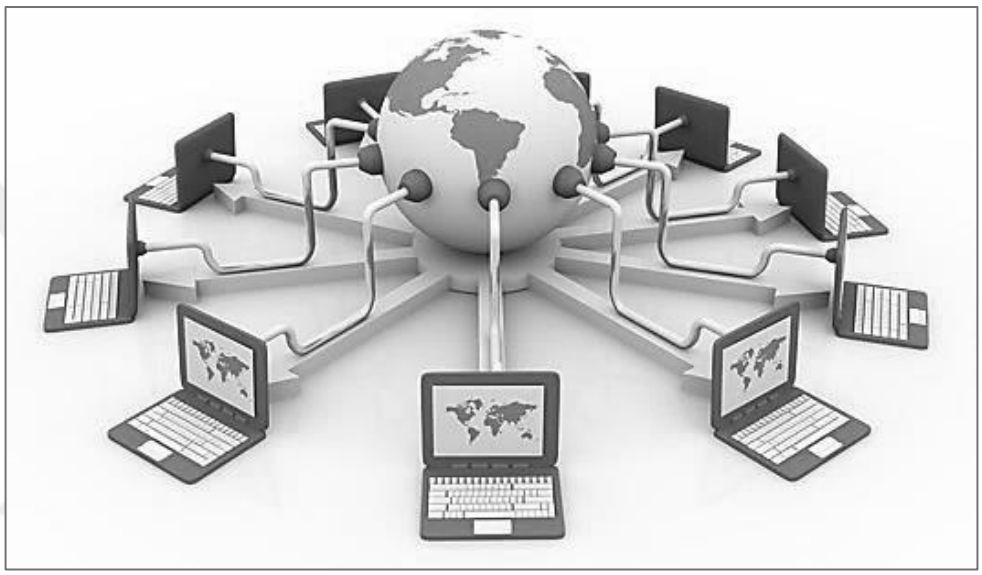
Çalışma çıktısı, oluşturulan ağ saldırısı tespit sistemlerinin performans yönünden analizi ve daha önceden aynı yöntemlerle geliştirilen bir sistemin sonuçlarıyla elde

edilen sonuçların karşılaştırılmasıdır. Bu karşılaştırmalı performans analizi, araştırmacılar için referans kaynak niteliğindedir. Çalışmada elde edilen sonuçlar literatürdeki benzer çalışmalarla [3] kıyaslandığında Apache Spark'ın çıkardığı yeni sürümlerle, büyük ağ verileri üzerindeki saldırıları tespit etmede daha başarılı bir hale geldiğini ortaya koymaktadır.

Çalışmanın ilk bölümünde konuya giriş niteliğinde bilgisayar ağlarından genel olarak bahsedilmiştir. İkinci bölümde; ağ saldırısı tespit sistemi geliştirmede kullanılan teknoloji, veri seti ve makine öğrenmesi algoritmaları detaylandırılmıştır. Üçüncü bölümde, sistem mimarisi üzerinden geliştirilen uygulama genel olarak anlatılmış; dördüncü bölümde ise geliştirilen yazılım detaylandırılmıştır. Son bölüm olan beşinci bölümde ise çalışma sonuçları raporlanmış, sonuçlar değerlendirilmiş ve konuyla ilgilenen araştırmacılar için önerilere yer verilmiştir.

1. BİLGİSAYAR AĞLARI VE TİPLERİ

Bilgisayar ağı en genel tanımla; kullanıcıların donanım ya da yazılım kaynaklarını paylaşmasını sağlayan, bilgisayarlar arasında kurulan iletişim sistemi yapısıdır (Şekil 1.1).



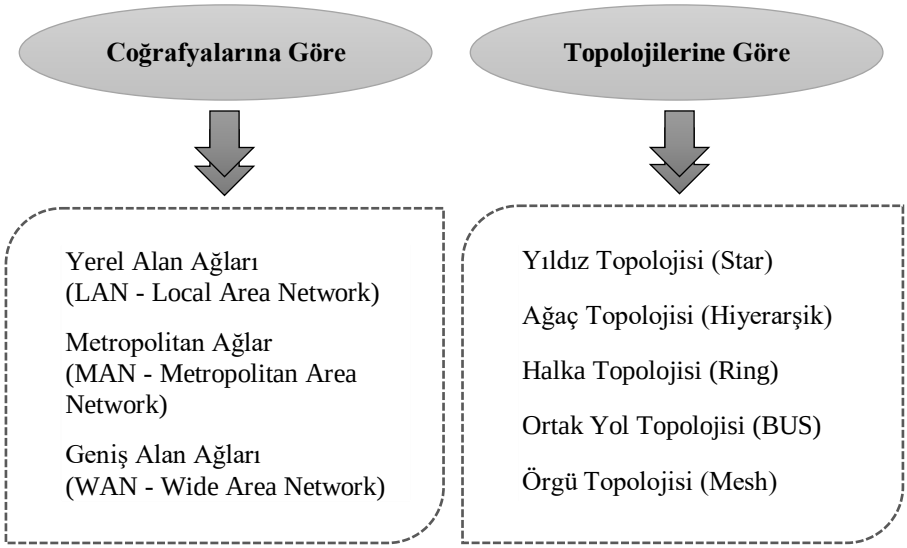
Şekil 1.1. Bilgisayar ağlarının genel gösterimi [8]

Bir ağın oluşabilmesi için en az iki bilgisayar birbirine bağlanmalıdır. Yakın veya uzak mesafedeki bilgisayarlar, iletişim cihazları ve iletişim hatları aracılığıyla birbirlerine bağlanırlar.

Bilgisayar ağları ile; bilgi ve sistem kaynakları, farklı kullanıcılar tarafından paylaşılır. Böylelikle bir yerden başka bir yere veri aktarımı mümkün olur. Ağa bağlı olan tüm bilgisayarlar birbirleriyle iletişim kurabilir ve aynı kaynakları paylaşabilirler.

1.1. Ağ Tipleri

Ağ tipleri Şekil 1.2'de görüldüğü gibi coğrafya ve topoloji bazında iki ana grupta incelenebilir.



Şekil 1.2. Bilgisayar ağı tipleri

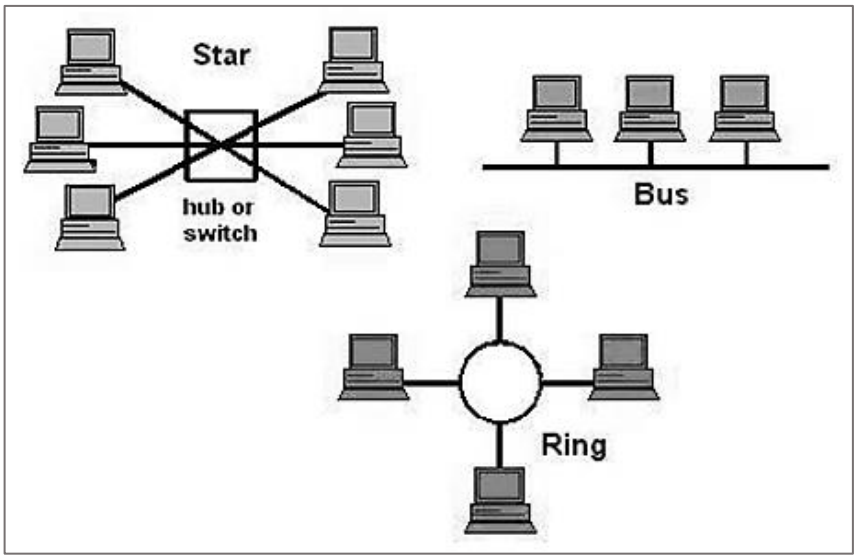
1.1.1. Coğrafyalarına göre ağ tipleri

- Yerel alan ağları (LAN, Local Area Network): Bina, firma gibi küçük alanları kapsayan dışarıya kapalı yüksek hızlı veri ağlarıdır. Bilgisayarlar arası mesafenin 1 kilometreyi geçmediği bu ağ çeşidi genellikle internete bağlanabilmek, basit uygulamaları ortak kullanabilmek gibi işlemler için kullanılır.
- Metropolitan ağlar (MAN, Metropolitan Area Network): Metropolitan ağların iletişim sağlayabildiği kapsama alanı; yerel alan ağlarınınkinden daha geniş, geniş alan ağlarınınkinden ise daha dardır. Genellikle üniversite yerleşkeleri, büyük iş yerleri gibi yerlerde kullanılırlar.
- Geniş alan ağları (WAN, Wide Area Network): Aralarındaki mesafe yüzlerce, binlerce kilometre kadar olan bilgisayarların iletişimini sağlayabilen ağlardır. Bu tür ağlar üzerinde on binlerce bilgisayarın çalışma imkânı vardır.

1.1.2. Topolojilerine göre ağ tipleri

Bilgisayar ağlarının yapılarına göre ayrımı topolojik bir ayrımdır. Topoloji bilgisayarların birbirine nasıl bağlandıklarını tanımlayan genel bir terimdir.

Topolojik anlamda Tablo 1.1'de detaylandırılmış olan yerel alan ağları, Şekil 1.3'te görüldüğü gibi doğrusal, halka ve yıldız yerleşim olmak üzere üç tipten oluşur [9].



Şekil 1.3. Topolojilerine göre yerel alan ağı tipleri [9]

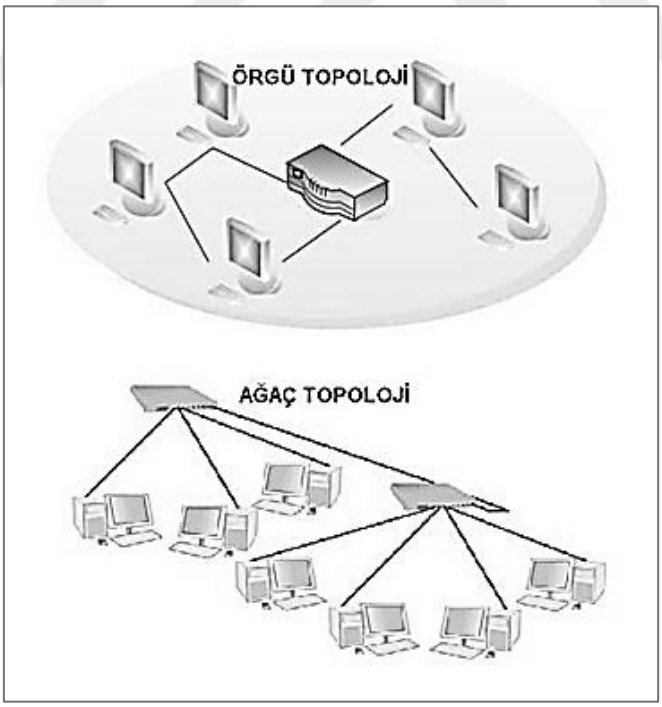
Tablo 1.1. Topolojilerine göre yerel alan ağı tiplerinin tanım, avantaj ve dezavantajları

	Tanımı	**Avantajları**	**Dezavantajları**
Doğrusal Yerleşim (Bus)	• Bilgisayarlar bir ana kablo üzerinde toplanırlar. • Performansı en düşük topolojiler arasındadır.	• Daha az kablo gerektirirler. Kablo yapıları güvenli olan topolojiler arasındadır. • Merkez birim ihtiyacı bulunmaz, ağa bilgisayar dâhil etmek kolaydır.	• Ana kabloda bir sorun çıktığında tüm ağ kullanılamaz duruma geldiğinden sorun kaynağının tespit edilmesi kolay değildir. • Ağda maksimum 30 bilgisayar yer alabilir.
Halka Yerleşim (Ring)	• Bilgisayarlar bir halkanın elemanları konumundadırlar. • Ağa düşen her veri ağa dâhil olan her bilgisayar tarafından görülür; fakat sadece kendi adresi ile uyum gösteren bilgisayarlara iletilir. Gelen veri hiçbir bilgisayar tarafından alınmazsa, devre dışı kalır.	• Kablo yapıları güvenli olan topolojiler arasındadır. • Merkez birim ihtiyacı bulunmaz, ağa bilgisayar dâhil etmek kolaydır.	• Bilgisayardan biri bozulduğunda tüm ağ kullanılamaz duruma geldiğinden sorun kaynağının tespit edilmesi kolay değildir. • Ağda maksimum 30 bilgisayar yer alabilir.

Tablo 1.1. (Devam) Topolojilerine göre yerel alan ağı tiplerinin tanım, avantaj ve dezavantajları

	Tanımı	Avantajları	Dezavantajları
Yıldız Yerleşim (Star)	• Bilgisayarlar, ağın merkezi olan switch'ten çekilen kablolar ile ağa bağlanırlar. • Ağa düşen veriler önce switch'e gelir; daha sonra switch'ten bilgisayarlara iletilir.	• Bilgisayarlar hub veya switch denilen ortak bir merkeze bağlıdır, ağı genişletmek kolaydır. • Bilgisayarlar bir switch ile birbirlerine bağlandıklarından bir bilgisayar çalışmadığında ağ çökmez. Bu durum, ağdaki sorunları tespit etmede kolaylık sağlar.	• Switch kullanıldığı için trafik artar. • Daha fazla kablo gerektirir. • Ağ switch'i bozulduğunda tüm ağ çöker. • Kurulumu pahalıdır.

Topolojik anlamda Tablo 1.2'de detaylandırılmış olan geniş alan ağları, Şekil 1.4'te görüldüğü gibi ağaç ve örgü topolojisi olmak üzere iki tipten oluşur [9].



Şekil 1.4. Topolojilerine göre geniş alan ağı tipleri [9]

Tablo 1.2. Topolojilerine göre geniş alan ağı tiplerinin tanım, avantaj ve dezavantajları

	Tanımı
Ağaç Topolojisi (Hiyerarşik)	● Ağın merkezinde, önemli bütün sorumlulukların üzerinde toplandığı bir bilgisayar yer alır. ● Dallanma seviyesinin arttığı yönde bilgisayarların görev seviyeleri düşer.
Örgü Topolojisi (Mesh)	● Ağın yapısında belirli bir yerleşim düzeni bulunmaz. ● Ağdaki bilgisayarların bağlantı mekanizması, kendilerine en yakın bilgisayara bağlanma şeklindedir.

2. AĞ SALDIRISI TESPİT SİSTEMİ ARAÇLARI

2.1. Büyük Veri Teknolojisi: Apache Spark

Apache Spark; büyük veri işleme amacıyla tasarlanmış, veri kümeleri üzerinde paralel olarak işlem yapma olanağı sağlayan, Scala dili ile yazılmış, açık kaynak kodlu bir platformdur [10].

MapReduce ve HDFS olmak üzere iki bileşenden oluşan bir diğer büyük veri işleme teknolojisi Hadoop ile karşılaştırılacak olursa; büyük verileri paralel işlemeyi sağlayan MapReduce alternatifi gibi düşünülebilir. MapReduce'e göre bellek işlemlerini 100 kat daha hızlı bir şekilde gerçekleştirir [10]. Hadoop ile veriler HDFS üzerinde saklanabilirken, Apache Spark ile aynı veriler daha kolay ve hızlı bir biçimde işlenebilir.

Apache Spark, birçok programlama dilini destekler. Scala, Python, Java ve SQL için kullanımı kolay bir API sağlar. Ayrıca içerdiği makine öğrenmesi kütüphanesi olan MLlib ile makine öğrenmesi algoritmaları birden fazla makinede çalıştırılabilir; böylelikle büyük veriler üzerinde daha hızlı analizler gerçekleştirilebilir.

2.2. Veri Seti: KDD Cup'99

KDD Cup'99, bir ağ saldırı tespit edicisi yapmak amacıyla gerçekleştirilen "Third International Knowledge Discovery and Data Mining Tools" yarışmasında kullanılan veri setidir. Askeri ağ ortamlarındaki çeşitli sahtekârlık saldırılarını geniş bir yelpazede kapsar, ağ üzerinde denetlenecek verilerin standart bir kümesini içerir.

1999'da Stolfo ve arkadaşları tarafından hazırlanan veri seti, anomali tespit yöntemlerinin değerlendirilmesi için en yaygın kullanılan veri seti olmuştur.

KDD Cup'99 eğitim veri seti, her biri 41 özellikten oluşan normal veya özel bir saldırı tipi olarak etiketlenen yaklaşık olarak 5 milyon bağlantı vektörü içerir. Sınıf anlamında temelde 5 başlıkta toplanabilen bağlantı vektörleri; 22'si çeşitli saldırı

tipleri ve biri normal bağlantı olmak üzere toplamda 23 çeşit bağlantı sınıfını kapsamaktadır.

Deneysel çalışmalar için farklı örnek sayıları ve farklı veri düzenlenme şekilleriyle oluşturulmuş birkaç KDD Cup'99 veri seti; içeriğindeki temel bağlantı tiplerine göre karakteristik dağılımları ile birlikte Tablo 2.1'de gösterilmiştir.

Tablo 2.1. Deneysel KDD Cup'99 veri setleri ve temel karakteristik dağılımları [11]

Dataset	DoS	Probe	U2R	R2L	Normal
10% KDD	391458	4107	52	1126	97277
Corrected KDD	229853	4166	70	16347	60593
Whole KDD	3883370	41102	52	1126	972780

Eğitim veri seti ile test veri setinin içeriği aynı değildir. Gerçekliğe uygun olması açısından; test veri setinde, eğitim veri setindekinden farklı saldırılara da yer verilmiştir. Bazı saldırı uzmanları, yeni geliştirilen saldırıların tespitinin işaretlenmiş bilinen saldırılar ile gerçekleşebileceğine inanmışlardır. Onlara göre bilinen saldırılar bilinmeyenlerin tespiti için yeterlidir [12].

2.2.1. İçerdiği saldırı tipleri

Toplamda 22 tipten oluşan eğitim verisi saldırı tipleri, Tablo 2.2'de gösterilmektedir.

Tablo 2.2. KDD Cup'99 saldırı tipleri

DoS	Probe	R2L	U2R
smurf	portsweep	ftp_write	buffer_overflow
teardrop	ipsweep	guess_passwd	perl
neptune	satan	imap	loadmodule
back	nmap	multihop	rootkit
pod		phf	
land		spy	
		warezmaster	
		warezclient	

Tablo 2.2'de de görüldüğü üzere KDD Cup'99 eğitim veri setinde yer alan saldırılar temel olarak 4 başlıkta incelenebilir [12]:

- Servisi reddetme saldırıları (DoS-Denial of Service Attack): Smurf, Teardrop, Neptune (SYN Flood), Back, Pod (Ping of Death), Land.
- İnceleme saldırıları (Probe Attack): PortSweep, IPSweep, Satan, Nmap.
- Yerele uzaktan saldırılar (R2L-Remote to Local Attack): Ftp-write, Guess password, IMAP, Multihop, Phf, Spy, Warezmaster, Warezclient.
- Root kullanım saldırıları (U2R-User to Root Attack): Buffer overflow, Perl, Loadmodule, Rootkit.

2.2.1.1. Servisi reddetme saldırıları (DoS-Denial of Service Attack)

Bu tip saldırılarda saldırgan, mantıklı isteklerini işlemek için yaptığı bazı hesaplamalarla çok dolu veya çok meşgul bellek kaynakları oluşturarak kullanıcıların makineye erişimini reddeder. Çoğu durumda saldırgan, kendi IP adresini kurban kullanıcının IP adresine değiştirir (IP address spoofing). Bu nedenle, saldırganın konumunu belirlemek ve ağ sistemini saldırgandan korumak kolay değildir [13].

Bir DoS saldırısı, ağ kaynaklarını devre dışı bırakır. Risk altındaki kaynakların yapılandırma dosyalarını değiştirme, ağ bileşenlerine fiziksel olarak zarar verme, kaynakları tüketme gibi yöntemlerle gerçekleştirilebilir [14]. Araştırmacılar kaynak açlığı saldırılarını (resource starvation attacks); güvenlik açığı saldırıları (vulnerability attacks) ve sel saldırıları (flooding attacks) olmak üzere iki genel gruba ayırmışlardır [15]. Güvenlik açığı saldırıları; bellek, CPU zamanı, disk alanı, veri yapıları gibi sistem kaynaklarını tüketmek için yazılım veya protokol hatalarından faydalanır. Sel saldırıları ise, sistemin yönetebileceğinden daha fazla paket veya istek gönderir.

KDD Cup'99 eğitim veri setinde yer alan DoS saldırıları şunlardır:

- Smurf: Bu tip saldırılarda saldırgan, bir DoS saldırısı oluşturmak için uzak konumlardan IP broadcast adreslerine yönelik "ICMP echo request" paketlerini kullanır. Bir Smurf saldırısında üç taraf vardır: saldırgan, aracı ve kurban (aracı da kurban olabilir). Aracı, kendi ağının IP broadcast adresine yönlendirilen bir "ICMP echo request" paketini alır. Eğer IP broadcast adreslerine yönlendirilen ICMP trafiği

aracı tarafından filtrelenmezse, ağdaki bilgisayarların çoğu bu ICMP paketini alır ve bir "ICMP echo reply" paketini geri gönderir. Bir ağdaki tüm bilgisayarlar bu ICMP isteğine yanıt verdiğinde, ciddi ağ tıkanıklığı veya kesintileri ortaya çıkabilir. Saldırganlar bu paketleri oluşturduklarında, kendi bilgisayarlarının IP adresini kaynak adres olarak kullanmazlar. Bunun yerine, saldırganın kurban olarak hedeflediği bilgisayarın sahte kaynak adresini içeren sahte paketler oluştururlar. Bu durumun sonucu olarak, aracıdaki tüm bilgisayarlar "ICMP echo request" lere cevap verdiğinde, kurbanın bilgisayarına cevap göndermiş olurlar ve kurban, ağı kullanılamaz hale getiren bir ağ tıkanıklığına maruz bırakılmış olur. Aracı, kurban olarak nitelendirilmemiş olsa da kurbanın yaşadığı problemlere maruz kalarak mağdur edilebilir [16].

- Teardrop: Bu tip saldırılar, eski TCP/IP yığınlarının implementasyonundaki bir hatadan faydalanan DoS saldırılarıdır [17]. Standart işleyişte, ağ üzerinden gelen paketler bilgisayarlarda bölünerek aktarılır. Paketler, içeriklerinde yer alan ofsetler kullanılarak verilere ayrıştırılır. Verinin doğru bir şekilde algılanması açısından bu ofsetlerin çakışmaması önemlidir. Bir Teardrop saldırısı yapan saldırgan, kurbanın bilgisayarına gönderdiği pakete üst üste ofsetler ekler. Eğer kurban bu karmaşıklığı çözemezse sistemi çöker [18].
- Neptune (SYN Flood): Bu tip saldırılar, her TCP/IP implementasyonunun bir dereceye kadar korunmasız olduğu DoS saldırılarıdır. Bir bilgisayara yapılan her yarı açık TCP bağlantısı; sunucunun bekleyen tüm bağlantıları tanımlayan, bilgileri depolayan veri yapısına bir kayıt eklemesine neden olur. Bu veri yapısı sonlu büyüklüktedir ve kasıtlı çok fazla kısmi açık bağlantı oluşturma ile taşma yapılabilir. Bir Neptune saldırısı ile kurbanın sunucu sistemindeki yarı açık bağlantı veri yapısı doldurulur ve sistem tablo boşaltılıncaya kadar yeni gelen bağlantıları kabul edemeyecek hale getirilir. Normalde beklemede olan bağlantıyla ilişkilendirilen bir zaman aşımı vardır; böylece yarı açık bağlantıların süresi dolduğunda kurban sunucusu kurtarılabilir. Ancak saldırıyı gerçekleştiren sistem, kurbanın sisteminde beklemedeki bağlantıların süresinin dolmasından daha hızlı bir şekilde yeni bağlantı istekleri yapan sahte IP paketleri (IP spoofed packets) göndermeye devam edebilir. Bu gibi durumlarda; sistem belleği tükenebilir, çökebilir veya başka şekilde çalışamaz duruma getirilebilir [17].
- Back: Bu tip saldırılar, Apache web sunucusuna karşı yapılan DoS saldırılarıdır.

Saldırgan, URL'lerin çok sayıda ön eğik çizgi karakterini (" / ") içeren istekler gönderir. Kurban sunucu bu istekleri işlemeye çalıştığında yavaşlar ve diğer isteği işleyemez hale gelir [17].

- Pod (Ping of Death): Bu tip saldırılar, TCP/IP implementasyonundaki bilinen bir hatadan faydalanılır. Birçok eski işletim sistemini etkileyen DoS saldırılarıdır**.** Saldırgan, IP spesifikasyonu tarafından izin verilen maksimum veri kapasitesini aşan bir IP paketi oluşturmak için ping sistemini kullanır [19]. Büyük boyutlu bir paket alan sistemlerde çökme, donma ve yeniden başlatma gibi reaksiyonlar görülebileceği gibi; bu durumlara bazı sistemlerin öngörülemeyen bir şekilde tepki verebileceği konusu çalışmalarda geniş çapta rapor edilmiştir [17].
- Land: Bu tip saldırılar, bazı eski TCP/IP implementasyonlarına karşı etkili olan DoS saldırılarıdır [17]. Bir Land saldırısında saldırgan, hedeflediği sisteme göndereceği paketlerin hedef IP adresi ile kaynak IP adresinin yerlerini değiştirir. Sahte hedef adresi içeren paketleri ağa yağdırır [19]. Böylelikle kurban sistem, farkında olmadan kendi sistemine sürekli olarak paket göndermek durumunda olur. Bu durumun devam etmesi sonucunda ağ hizmet veremez hale gelir.

2.2.1.2. İnceleme saldırıları (Probe Attack)

Bu tip saldırılarda saldırgan, güvenlik kontrollerini atlatarak bilgisayar ağı hakkında bilgi toplamaya çalışır.

KDD Cup'99 eğitim veri setinde yer alan Probe saldırıları şunlardır:

- PortSweep-IPSweep: PortSweep saldırıları, bir ağda veya alt ağda, belirli bir bilgisayarın hangi port'larının açıldığını taramak için kullanılan saldırılardır. IPSweep saldırıları ise, bir gözetleme taraması yoluyla hangi host'ların ağ üzerinde dinlemede olduğunu saptayan saldırılardır. PortSweep ve IPSweep saldırıları, çalışan host'u ve bu host'un servis tiplerini saptar. Bu toplanan bilgiler, saldırganlar tarafından saldırıların yürütülmesi için güvenlik açığı bulunan bilgisayarların aranmasında kullanılabilir. Lineer tek kaynaklı şekilde gerçekleştirilenler başta olmak üzere devam eden bir PortSweep veya IPSweep saldırısını tespit etmek oldukça kolaydır. Ancak tarama sırasında birden fazla host'un mu, sahte kaynak IP'lerinin mi, zamanı olmayan lineer bir yöntemin mi kullanıldığını tespit etmek oldukça zordur [20].

- Satan: C ve Perl programlarının koleksiyonu olan bir programdır. Saldırganların daha fazla saldırı başlatabilmesi için faydalanabileceği nitelikte olan büyük miktarda bilgiyi ağ servislerinden toplar. Bu bilgiler temel olarak ağ servislerinin yanlış yapılandırılmasını, işletim sistemlerinde veya ağın yardımcı programlarındaki iyi bilinen güvenlik açıklarını, zayıf politika kararlarını içerir. Hafif mod, normal mod ve ağır mod olmak üzere üç çalışma modu vardır. Her bir modda tarama yapılan güvenlik açığı listesi farklıdır. Taranmak istenilen güvenlik açıklarını kapsayan mod seçilerek tarama gerçekleştirilir [20].
- Nmap: Temel olarak IP taramalarını, port taramalarını, güvenlik duvarı taramalarını ve kurban bilgisayarlara yönlendirilen ham IP paketlerini kullanarak OS parmak izi taraması yapan; ücretsiz, açık kaynaklı bir programdır. Paket içi zamanlama ayarlanabilir ve portlar sırayla veya nadiren taranabilir. Nmap tarafından gerçekleştirilen taramaları tespit etmek, çoklu kaynakların dağıtılması ve inceleme işleminin uzun süre gizli kalmasına neden olan saldırı zamanlama çizelgesinin yavaşlaması nedeniyle oldukça zordur [20].

2.2.1.3. Yerele uzaktan saldırılar (R2L-Remote to Local Attack)

Bu tip saldırılarda saldırgan, makineye paket göndererek bazı güvenlik açıklarından faydalanır; herhangi bir hesaba sahip olmadığı hedef makinenin bir kullanıcısıymış gibi yerel erişimi sağlamaya çalışır.

KDD Cup'99 eğitim veri setinde yer alan R2L saldırıları şunlardır:

- Ftp-write: Bu tip saldırılar, ortak bir anonim FTP yanlış yapılandırmasından faydalanan R2L saldırılarıdır. Anonim FTP root dizini ve onun alt dizinleri FTP hesabına ait olmamalı veya FTP hesabıyla aynı grupta olmamalıdır. Bu dizinlerden herhangi biri FTP'ye aitse veya FTP hesabıyla aynı gruptaysa ve yazmaya karşı korumalı değilse, izinsiz giren kişi dosyaları ekleyebilir ve nihayetinde sisteme yerel erişim kazanabilir [17].
- Guess password: Bu tip saldırılar, çeşitli yöntemler kullanılarak yapılan şifre tahminlemeleri ile yerel erişimin sağlanmaya çalışıldığı R2L saldırılarıdır.
- IMAP: Bu tip saldırılar, Redhat Linux 4.2'nin IMAP sunucusunda saldırganların root ayrıcalıklarıyla rastgele komutlar çalıştırmasına izin veren bir buffer overflow'dan faydalanır. IMAP sunucusu; e-posta klasörlerine erişim, oturum açan

kullanıcı adına bazı dosya manipülasyonları yapma gibi işlemlerin gerçekleştirilebilmesi için root ayrıcalıklarıyla çalıştırılmalıdır. Normal işleyişte, root olarak oturum açıldıktan sonra bu ayrıcalıklar atılır. Ancak; oturum açma işleminin doğrulama kodunda buffer overflow hatası varsa, sunucuya root erişim sağlamak için bu hatadan yararlanılabilir. Saldırganlar; IMAP sunucusunun güvenlik açığı bulunan bir sürümünü çalıştıran sisteme, özel olarak hazırlanmış bir metin gönderme ile buffer overflow'a neden olabilir ve root ayrıcalıklarıyla rastgele komutlar çalıştırabilir [17].

- Multihop: Bazı ağ saldırısı tespit sistemleri, yönlendiricinin hemen dışındaki trafiği izler ve yalnızca bu ağa giren-çıkan trafiği görür. Saldırganlar; bahsi geçen ağ saldırısı tespit sistemleri ile korunan, yapısal olarak iletim yolu boyunca bir veya daha fazla ara düğüme sahip olan multihop ağ üzerinde, sistem için şüphe uyandırmayacak bir şekilde multihop senaryosundan faydalanarak giriş yaptığı makineyi iç ağda saldırıları için kullanır. İzlenen ağın hemen dışında bulunan ağ saldırısı tespit sistemi, iç ağdan kaynaklanan bir saldırı göremeyecektir [17]. Saldırganlar, bu yöntem ile oldukça etkili ve tespit edilemeyen DoS saldırıları başlatabilir.
- Phf: Bu tip saldırılar; HTTP sunucusunun ayrıcalık seviyesine sahip komutları çalıştırmak için, iyi yazılmamış bir CGI script'ini kötüye kullanır. CGI programları, web sunucuları ile bu sunucuların dışındaki programlar arasında ortak çalışma platformu oluştururlar. Shell tabanlı kütüphane çağrılarının kötüye kullanımını önlemek için escape_shell_cmd() fonksiyonuna güvenen herhangi bir CGI programı, saldırılara açık olabilir. Bu güvenlik açığı, Apache web sunucusu örnek koduyla birlikte dağıtılan "phf" programı tarafından açıkça gösterilir [17]. Saldırgan; güvenlik açığı bulunan CGI programlarını, "phf" programını kullanarak tespit eder.
- Spy: Spy, bilgi toplamak için birkaç kez tehlikeye atılmış makineye geri dönen bir çeşit bilgi toplayıcıdır. Bir Spy, gizli veri dosyalarını arayabilir veya kullanıcının kişisel e-postalarını okuyabilir. Tespit edilme olasılığını en aza indirecek adımlar atarak bilgi toplar [17]. Saldırgan; bu bilgi toplayıcıları kullanarak saldırılarını gerçekleştirir.
- Warezmaster: Bu tip saldırılar, bir dosya aktarma protokolü (FTP) sunucusuyla ilişkilendirilmiş bir sistem hatasını kullanır. Normalde misafir kullanıcıların hiçbir zaman bir FTP sunucusunda yazma izinleri olmaz. Dolayısıyla sunucuya hiçbir

zaman dosya yükleyemezler. Çoğu public domain FTP sunucusu, veri indirmek için misafir hesaplarına sahip olur. Herkes misafir hesaplarını kullanarak bir FTP sunucusunda oturum açabilir. Warezmaster saldırıları, bir FTP sunucusunun yanlışlıkla sistemdeki kullanıcılara yazma izni vermesi durumunda gerçekleşir. Dolayısıyla herhangi bir kullanıcı giriş yapabilir ve dosya yükleyebilir. Saldırının yürütülmesi sırasında saldırgan, misafir hesabını kullanarak sunucuda oturum açar. Saldırgan daha sonra gizli bir dizin oluşturur ve kendi yasadışı yazılımının kopyaları olan "warez" i sunucuya yükler. Diğer kullanıcılar daha sonra bu dosyaları indirebilir. Bu saldırıları önlemek için, FTP sunucusundaki kullanıcılara doğru izinler verilmelidir. Warezmaster saldırısı FTP bağlantısı sırasında dosya yüklemesi gerektirdiğinden gözlenebilecek ilgili özellikler; devam etmekte olan bir FTP oturumu, yüklenmekte olan dosyalar ve oluşturulan gizli dizinlerdir [21].

- Warezclient: Warezclient saldırısı, Warezmaster saldırısı yapıldıktan sonra FTP bağlantısı sırasında herhangi bir yasal kullanıcı tarafından başlatılabilir. Warezclient saldırısı sırasında, kullanıcılar başarılı bir Warezmaster saldırısı ile daha önce yayınlanan yasadışı "warez" yazılımını indirirler. Bu işlem FTP sunucusundan dosya indirmeyi gerektirdiğinden, saldırı dinamikleri tamamen yasal bir işlem yansıtır. Bir Warezclient saldırısını tespit etmek için gözlemlenebilecek tek özellik, FTP sunucusundaki misafir kullanıcılar tarafından normal olarak erişilemeyen gizli dizinlerden veya diğer dizinlerden yapılan dosya indirme işlemidir. Bu işlem, tüm yasal dizinlerin izlenmesini ve FTP oturumları sırasında indirilen dosyaların yasal dizinlere ait olup olmadığını kontrol etmeyi gerektirir [21].

2.2.1.4. Root kullanım saldırıları (U2R-User to Root Attack)

Bu tip saldırılarda saldırgan, sisteme root erişimi sağlamak için bazı güvenlik açıklarından faydalanarak sistemdeki normal bir kullanıcının hesabına erişim ile ataklarına başlar. U2R saldırılarına sosyal mühendislikte çokça rastlanır.

KDD Cup'99 eğitim veri setinde yer alan U2R saldırıları şunlardır:

- Buffer overflow: Buffer, C dilindeki dizi veya işaretçi yapılarında olduğu gibi bitişik olarak ayrılmış bir bellek öbeğidir. Genellikle bir buffer'da otomatik sınır kontrolü yoktur. Buffer overflow, bir program veya işlem beklenenden daha fazla veriyi buffer'da depolamaya çalıştığında gerçekleşir. Buffer'lar sınırlı miktarda veri

depolamak için oluşturulduğundan ekstra bilginin bir yere gitmesi veya komşu tamponlara taşması; içlerinde tutulan geçerli veri üzerine veri yazılması ve böylece geçerli verinin bozulması sonucunu doğurur. Buffer overflow saldırıları bu güvenlik açığından faydalanır. Saldırgan, buffer'ın taşma alanına yürütmeye çalıştığı kodu yerleştirir ve fonksiyonların dönüş adresinin üzerine yazarak buffer'a geri dönüşü işaret eder. Root izinleri olan saldırgan; kullanıcı dosyalarına zarar verme, verileri değiştirme veya gizli bilgileri ifşa etme gibi belirli eylemleri tetiklemek için içerdeki kodunu çalıştırır [20].

- Perl: Bu tip saldırılar, bazı Perl implementasyonlarındaki bir hatadan faydalanır. Bir Perl sürümü olan Suidperl, kayıtlı set-user-ID ve set-group-ID script'lerini destekler. Suidperl'in ilk sürümlerinde, etkin kullanıcı ve grup kimlikleri değiştirilirken root ayrıcalıkları tamamen bırakılmaz. Bu nedenle, Suidperl'e sahip olan bir sistemde kayıtlı set-user-ID ve set-group-ID script'leri sürekli olarak desteklenir. Dolayısıyla, sistemde bir hesaba erişimi olan herkes root yetkisi kazanabilir [17]. Saldırganlar, bu durumu kötüye kullanarak sisteme root erişim sağlar ve saldırılarını gerçekleştirir.
- Loadmodule: Bu tip saldırılar, xnews pencere sistemini kullanan SunOS 4.1 sistemlerine karşı yapılan saldırılardır. SunOS 4.1.x içindeki Loadmodule programı, xnews pencere sistemi sunucusu tarafından mevcut çalışan sisteme dinamik olarak yüklenebilen iki çekirdek sürücüsü yüklemek ve "/dev" dizininde bu modülleri kullanmak amacıyla özel aygıtlar oluşturmak için kullanılır. Loadmodule programının ortamını temizleme yöntemindeki bir hata nedeniyle, yetkisiz kullanıcılar yerel makineye root erişimi sağlayabilir [17]. Saldırganlar, bu durumu kötüye kullanarak Loadmodule tarafından gerekli şekilde temizlenmemiş olan makinede root erişim sağlar ve saldırılarını gerçekleştirir.
- Rootkit: Rootkit, işletim sisteminden varlığını gizleyen bir çeşit program koleksiyonudur. Kendisini tehlikeye atarak hedeflediği makineye bir kez erişim sağlayan saldırganların, kurban makineye erişimini sürdürmeye yardımcı olmak için tasarlanmış kötü amaçlı Rootkit'ler vardır. Genellikle bu tip rootkitler; sniffer, login araçları, su, sisteme izinsiz erişime izin veren diğer programlar ile ps ve netstat'ın yeni versiyonlarından oluşurlar. Bir sniffer'ın çalışmakta olduğu gerçeğini ve belirli dizinlerdeki dosyaları gizlerler. Saldırgan, Rootkit kurulduktan sonra sniffer

log'larını indirmek için birkaç kez geri gelir [17]; kötü amaçlı Rootkit programını kullanarak saldırılarını gerçekleştirir.

2.2.2. İçerdiği özellikler

KDD Cup'99 veri setinin içerdiği özellikler temel olarak 3 grupta sınıflandırılabilir [12]:

- Temel özellikler
- Trafik özellikleri
- İçerik özellikleri

Tablo 2.3'te yer alan veri seti özelliklerine ait değerler incelenecek olursa; özellikler kategorik değerler içerebilmekte, belirlenmiş bazı sayısal değerleri alabilmekte veya kendisi için tanımlanmış sayısal değer aralıklarından herhangi bir değere sahip olabilmektedir.

Tablo 2.3. KDD Cup'99 veri seti tüm özellikleri ve özellik değer bilgileri [22]

	Özellik ismi	**Değerleri**		**Özellik ismi**	**Değerleri**
1	duration length	[0 - 58329]	22	is_guest_login	{0, 1}
2	protocol_type	{icmp, tcp, udp}	23	count	[0 - 511]
3	service	{auth,..., Z39_50}	24	srv_count	[0 - 511]
4	flag	{OTH,…, SH}	25	serror_rate	[0 - 1]
5	src_bytes	[0 - 6,9337564E8]	26	srv_serror_rate	[0 - 1]
6	dst_bytes	[0 - 5155468]	27	rerror_rate	[0 - 1]
7	land	{0, 1}	28	srv_rerror_rate	[0 - 1]
8	wrong_fragment	{0, 1, 3}	29	same_srv_rate	[0 - 1]
9	urgent	{0, 1, 2, 3}	30	diff_srv_rate	[0 - 1]
10	hot	[0 - 30]	31	srv_diff_host_rate	[0 - 255]
11	num_failed_logins	{0, 1, 2, 3, 4, 5}	32	dst_host_count	[0 - 255]
12	logged_in	{0, 1}	33	dst_host_srv_count	[0 - 1]
13	lnum_compromised	[0 - 884]	34	dst_host_same_srv_rate	[0 - 1]

Tablo 2.3. (Devam) KDD Cup'99 veri seti tüm özellikleri ve özellik değer bilgileri [22]

	Özellik ismi	**Değerleri**		**Özellik ismi**	**Değerleri**
14	lroot_shell	{0, 1}	35	dst_host_diff_srv_rate	[0 - 1]
15	lsu_attempted	{0, 1, 2}	36	dst_host_same_src_port_rate	[0 - 1]
16	lnum_root	[0 - 993]	37	dst_host_srv_diff_host_rate	[0 - 1]
17	lnum_file_creations	[0 - 28]	38	dst_host_serror_rate	[0 - 1]
18	lnum_shells	{0, 1, 2}	39	dst_host_srv_serror_rate	[0 - 1]
19	lnum_access_files	{0, 1, 2, 3,…, 8}	40	dst_host_rerror_rate	[0 - 1]
20	lnum_outbound_cmds	{0}	41	dst_host_srv_rerror_rate	[0 - 1]
21	is_host_login	{0}			

2.2.2.1. Temel özellikler

Bu kategori, TCP/IP bağlantısından elde edilebilir bütün özellikleri kapsar. Bu özelliklerin çoğu, saldırı tespitinde önemli gecikme nedenidir.

Tablo 2.4'te detaylandırılan KDD Cup'99 veri setinin içerdiği bağımsız TCP bağlantılarının temel özellikleri şunlardır:

- Bağlantı devam ederken geçen süre
- Bağlantıda kullanılan protokol tipi
- Bağlantı ile hedeflenen ağ servisi
- Bağlantı kaynağından hedefe, bağlantı hedefinden kaynağına akan veri baytlarının sayısı
- Bağlantının normal veya hatalı olduğu bilgisi
- Bağlantının anlık bağlantı ile aynı host ya da port'tan olup olmadığı bilgisi
- Bağlantıda yer alan yanlış-önemli parçaların sayısı

Ayrık veya sürekli veriler içerebilen özelliklerin alabileceği değer bilgileri, Tablo 2.3'te gösterildiği gibidir.

Tablo 2.4. KDD Cup'99 veri seti temel özellikleri [11]

Özellik ismi	**Açıklama**	**Tip**
duration length	Bağlantı süresi (saniye)	Sürekli
protocol_type	Protokolün tipi (TCP, UDP…)	Ayırık
service	Hedef ağ servisi (HTTP, Telnet…)	Ayrık
src_bytes	Kaynaktan hedefe veri baytlarının sayısı	Sürekli
dst_bytes	Hedeften kaynağa veri baytlarının sayısı	Sürekli
flag	Bağlantının normal ya da hatalı durumu	Sürekli
land	Bağlantı, aynı host/porttan ise: 1 dir. Aksi halde: 0 dır.	Ayrık
wrong_fragment	Yanlış parçaların sayısı	Sürekli
urgent	Önemli paketlerin sayısı	Sürekli

2.2.2.2. Trafik özellikleri

Bu özellikler, host ya da portları tarama sıklıklarına göre gruplanırlar. 2 saniye zaman aralıkları kullanarak tarama yapanlara zaman tabanlı trafik özellikleri denir. 2 saniyelik zaman penceresi yerine 100 bağlantılık bağlantı penceresi kullananlar ise bağlantı tabanlı trafik özellikleri olarak adlandırılırlar.

Zaman tabanlı trafik özellikleri 2 gruba ayrılır:

- Aynı host özellikleri: Sadece anlık bağlantı ile aynı hedef host'a sahip son 2 saniye içerisindeki bağlantıları inceler. Servis gibi protokol davranışlarıyla ilgili istatistikleri hesaplar.
- Aynı servis özellikleri: Sadece anlık bağlantı ile aynı servise sahip son 2 saniye içerisindeki bağlantıları inceler.

Tablo 2.5'te detaylandırılan KDD Cup'99 veri setinin içerdiği, 2 saniyelik zaman penceresi kullanılarak hesaplanan trafik özellikleri şöyledir:

- Anlık bağlantı ile aynı host'a, aynı servise olan son 2 saniye içindeki bağlantı sayısı
- "SYN"-"REJ" hataları bulunan bağlantılar

- Anlık bağlantı ile aynı servis-farklı servis bağlantıları
- Anlık bağlantı ile farklı host bağlantıları

Sürekli veriler içerebilen özelliklerin alabileceği değer bilgileri, Tablo 2.3'te gösterildiği gibidir.

Tablo 2.5. KDD Cup'99 veri seti trafik özellikleri [11]

Özellik ismi	**Açıklama**	**Tip**
count	Son 2 saniye içinde anlık bağlantı ile aynı host'a olan bağlantı sayısı	Sürekli
NOT: Aşağıdaki özelliklere aynı host bağlantıları için bakılır.		
serror_rate	Bağlantıların "SYN" hataları var	Sürekli
rerror_rate	Bağlantıların "REJ" hataları var	Sürekli
same_srv_rate	Aynı servis bağlantıları	Sürekli
diff_srv_rate	Farklı servis bağlantıları	Sürekli
srv_count	Son 2 saniye içinde anlık bağlantı ile aynı servise bağlantı sayısı	Sürekli
NOT: Aşağıdaki özelliklere aynı servis bağlantıları için bakılır.		
srv_serror_rate	Bağlantıların "SYN" hataları var	Sürekli
srv_rerror_rate	Bağlantıların "REJ" hataları var	Sürekli
srv_diff_host_rate	Bağlantılar, farklı host bağlantıları	Sürekli

2.2.2.3. İçerik özellikleri

Çoğu servisi reddetme (DoS) ve inceleme (Probe) saldırılarının aksine, yerele uzaktan saldırılar (R2L) ve root kullanım saldırılarında (U2R) sıkça tekrar eden örüntü dizileri yoktur. Bundan dolayı servisi reddetme (DoS) ve inceleme saldırıları (Probe), bazı host'lar için çok kısa zaman periyodlarında pek çok bağlantı içerir. Ancak yerele uzaktan saldırılar (R2L) ve root kullanım saldırıları (U2R), paketin veri bölümlerine gömülüdür ve normalde sadece tek bir bağlantı içerir. Bu çeşit saldırıları tespitte, veri parçalarındaki şüpheli davranışlara bakmak için bazı özelliklere ihtiyaç olur. Bu özellikler içerik özellikleri olarak adlandırılır. Başarısız oturum açma sıklığı gibi özellikler, içerik özelliğidir.

Tablo 2.6'da detaylandırılan KDD Cup'99 veri setinde yer alan bir bağlantının içerik özellikleri şöyledir:

- Yeni işaretlerin sayısı
- Başarısız oturum açma denemelerinin sayısı
- Başarılı oturum açılıp açılmadığı bilgisi
- Tehlikeli durumların sayısı
- Root shell erişim elde edilip edilmediği bilgisi
- Girişim komutunun "su root" olup olmadığı bilgisi
- "root"-"shell" işlemlerin sayısı
- Dosya oluşturma işlemlerinin sayısı
- Erişim kontrol dosyaları üzerindeki işlem sayısı
- Bir FTP oturumda giden komutların sayısı
- Girişin yeni işaret listesinde bulunma durumu bilgisi
- Girişin, misafir girişi olup olmadığı bilgisi

Ayrık veya sürekli veriler içerebilen özelliklerin alabileceği değer bilgileri, Tablo 2.3'te gösterildiği gibidir.

Tablo 2.6. KDD Cup'99 veri seti içerik özellikleri [11]

Özellik ismi	**Açıklama**	**Tip**
hot	Yeni (hot) işaretlerin sayısı	Sürekli
num_failed_logins	Başarısız oturum açma denemelerinin sayısı	Sürekli
logged_in	Başarılı otorum açıldıysa: 1 dir. Aksi halde: 0 dır.	Ayrık
num_compromised	Tehlikeli (compromised) durumların sayısı	Sürekli
root_shell	Root shell elde edildiyse :1 dir. Aksi halde: 0 dır.	Ayrık
su_attempted	Girişim komutu "su root" ise: 1 dir. Aksi halde: 0 dır.	Ayrık
num_root	"root" erişimlerin sayısı	Sürekli
num_file_creations	Dosya oluşturma işlemlerinin sayısı	Sürekli

Tablo 2.6. (Devam) KDD Cup'99 veri seti içerik özellikleri [11]

Özellik ismi	**Açıklama**	**Tip**
num_shells	"shell" istemlerinin sayısı	Sürekli
num_access_files	Erişim kontrol dosyaları üzerindeki işlem sayısı	Sürekli
num_outbound_cmds	Bir FTP oturumda (session), giden komutların sayısı	Sürekli
is_hot_login	Giriş, yeni işaret (hot) listesine aitse: 1 dir. Aksi halde: 0 dır.	Ayrık
is_guest_login	Giriş, bir misafir (guest) girişse: 1dir. Aksi halde: 0 dır.	Ayrık

2.2.3. Saldırı tipleriyle ilişkili özellikler

Deneysel işlemlerin gerçekçiliği açısından, araştırmacılar KDD Cup'99 veri seti içerisinde hangi saldırı tipinin hangi özelliklerle ilişkili olduğunun tespiti üzerinde çalışmaktadırlar.

Tablo 2.7'de görülen, bir ağ saldırısı tespit sistemi çalışması sonuçlarından refere edilen saldırı tipi özellik ilişkilendirmeleri şöyledir:

- Back: src_bytes, dst_bytes
- Land: land
- Neptune: dst_host_srv_serror_rate, service, flag, src_bytes, count, srv_serror_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, same_srv_rate
- Pod: wrong_fragment
- Smurf: protocol_type, dst_host_count, service, src_bytes, dst_bytes, same_srv_rate, logged_in, serror_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_srv_serror_rate, diff_srv_rate
- Teardrop: wrong_fragment
- Satan: rerror_rate
- IpSweep: dst_host_same_src_port_rate

- Nmap: src_bytes
- PortSweep: srv_rerror_rate
- Normal: service, dst_host_srv_serror_rate, dst_bytes, logged_in, count, serror_rate, dst_host_srv_diff_host_rate, srv_serror_rate, same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_count, dst_host_same_srv_rate, dst_host_serror_rate, diff_srv_rate
- Guess password: num_failed_logins, dst_bytes, service, flag
- Ftp-write: urgent, count
- IMAP: service, dst_host_srv_serror_rate
- Phf: dst_bytes, hot, lroot_shell, src_bytes
- Multihop: count
- Warezmaster: dst_bytes, duration length
- Warezclient: service, srv_count, srv_serror_rate
- Spy: dst_host_srv_serror_rate, duration length
- Buffer overflow: service, srv_count, lroot_shell, dst_bytes
- Loadmodule: dst_host_same_src_port_rate, srv_count, service
- Perl: lroot_shell, lnum_root, lnum_shells, src_bytes
- Rootkit: srv_count, count, service

Yukarıda saldırı tipleriyle ilişkili şekilde maddelendirilmiş özelliklere ait içerik bilgileri; Tablo 2.3, Tablo 2.4, Tablo 2.5 ve Tablo 2.6'da yer almaktadır.

Tablo 2.7. KDD Cup'99 veri seti bağlantı tipi-özellik ilişkilendirmesi [23]

Saldırı Tipi	**İlişkili Özellikler**	**Saldırı Tipi**	**İlişkili Özellikler**
back	5,6	ftp_write	9,23
land	7	imap	3,39
neptune	3,4,5,23,26,29,30,31,32,34,36,37,38,39	phf	6,10,14,5
pod	8	multihop	23
smurf	2,3,5,6,12,25,29,30,32,36,37,39	warezmaster	6,1
teardrop	8	warezclient	3,24,26

Tablo 2.7. (Devam) KDD Cup'99 veri seti bağlantı tipi-özellik ilişkilendirmesi [23]

Saldırı Tipi	**İlişkili Özellikler**	**Saldırı Tipi**	**İlişkili Özellikler**
satan	27	spy	39,1
ipsweep	36	buffer_overflow	3,24,14,6
nmap	5	loadmodule	36,24,3
portsweep	28	perl	14,16,18,5
normal	3,6,12,23,25,26,29,30,33,34,35,36,37,38,39	rootkit	24,23,3
guess_passwd	11,6,3,4		

2.3. Makine Öğrenmesi Algoritmaları

2.3.1. Lojistik Regresyon Algoritması (Logistic Regression Algorithm)

Bağımlı değişkeni kategorik olan problemler için kullanılan yöntemlerden birisidir. Amaç, bağımlı ve bağımsız değişkenler arasındaki ilişkinin modelini oluşturmaktır [24]. Modelin matematiksel yorumlama anlamında sağladığı kolaylık, bu yöntemi seçilebilir kılmaktadır.

Problem üzerinde kullanılması gereken lojistik regresyon tipini seçerken Tablo 2.8'de yer alan kıstaslar göz önünde bulundurulmalıdır.

Tablo 2.8. Lojistik Regresyon tipinin seçilmesi

Değişken Tipi	**Lojistik Regresyon Tipi**	
Bağımsız Değişken: x	x = 1 ise	x > 1 ise
	Lojistik Regresyon	Çoklu Lojistik Regresyon
Bağımlı Değişken: y	y kategori sayısı = 2 ise	y kategori sayısı > 2 ise
	İkili Lojistik Regresyon	Multinominal Lojistik Regresyon

2.3.1.1. İkili Lojistik Regresyon

Bağımsız değişken "x" ve bağımlı değişken "Y" olmak üzere, bağımlı değişkene ait ortalama değer " $E(Y \mid x)$ " ile gösterilir. Koşullu ortalama olarak da bilinen bu

ortalama değer, "Y" nin beklenen değeri olarak yorumlanır; Eşitlik (2.1) ile gösterilebilir [24].

$$E(Y \mid x) = \beta_0 + \beta_1 x \tag{2.1}$$

Y'ye ait kategori sayısı 2 ise koşullu ortalama değer aralığı " $0 \le E(Y \mid x) \le 1$ " şeklinde olacaktır.

" $\pi(x) = E(Y \mid x)$" olmak üzere Eşitlik (2.2) ve Eşitlik (2.3)'te gösterilen olasılıklar oluşturulabilir.

$$P(Y = 1 \mid x) = \pi(x) \tag{2.2}$$

$$P(Y = 0 \mid x) = 1 - \pi(x) \tag{2.3}$$

Lojistik regresyon modeli Eşitlik (2.4) ile gösterilir.

$$\pi(x) = E(Y \mid x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} \tag{2.4}$$

Bu durumda Eşitlik (2.4) kullanılarak Eşitlik (2.2) ve Eşitlik (2.3) düzenlenecek olursa aşağıdaki koşullu olasılık denklemleri elde edilecektir.

$$P(Y = 1 \mid x) = \pi(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} \tag{2.5}$$

$$P(Y = 0 \mid x) = 1 - \pi(x) = 1 - \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} = \frac{1}{1 + e^{\beta_0 + \beta_1 x}} \tag{2.6}$$

Elde edilen bu eşitliklerden sonra oluşturulması gereken iki önemli kavram denklemi vardır: bahis (odds) ve bahis değerleri oranı (odds ratio). Bahis denilen kavram; bağımlı değişkenin görülme olasılığının görülmeme olasılığına oranıdır. Değişkenler arası ilişkinin ölçüsü niteliğinde olan bahis oranı ise; iki değişkene ait bahis değerleri oranı olarak tanımlanabilir [25].

Bu bilgiler ışığında bağımlı değişkene ait bahis değeri hesaplanacak olursa, Eşitlik (2.7) elde edilecektir. Bahis oranı ise Eşitlik (2.8)'deki gibi olacaktır.

$$\text{odds}[Y] = \frac{p}{1-p} = \frac{P(Y=1|x)}{P(Y=0|x)} = \frac{\pi(x)}{1-\pi(x)} \tag{2.7}$$

$$OR = \frac{\text{odds}[x=1]}{\text{odds}[x=0]} = \frac{\pi(1)}{1-\pi(1)} \Big/ \frac{\pi(0)}{1-\pi(0)} \tag{2.8}$$

Eşitlik (2.7) ile gösterilen bahislerin logaritması alınacak olursa, Eşitlik (2.9) elde edilecektir. Gerçekleştirilen bu lojit dönüşüm, lojistik regresyonda "log-bahis" olarak tanımlanır.

$$g(x) = \ln[\text{odds}] = \ln\frac{\pi(x)}{1-\pi(x)} = \ln\frac{P(Y=1|x)}{P(Y=0|x)} = \beta_0 + \beta_1 x \tag{2.9}$$

Eşitlik (2.5) ve Eşitlik (2.6) "g(x)" ile yeniden düzenlenecek olursa, Eşitlik (2.10) ve Eşitlik (2.11) elde edilecektir.

$$P(Y=0|x) = \frac{1}{1+e^{g(x)}} \tag{2.10}$$

$$P(Y=1|x) = \frac{e^{g(x)}}{1+e^{g(x)}} \tag{2.11}$$

Lojit dönüşüm ile doğrusal bir forma dönüşen denklemde yer alan katsayılar şöyle yorumlanır [26]:

- β_0 : x sıfır olduğunda log-bahis oranındaki değişimi gösterir.
- β_1: x bir birim değiştiğinde, log-bahis oranındaki değişimi gösterir.

Parametre Tahmini:

Lojistik regresyonda parametre tahminlemesi, en çok olabilirlik yöntemi kullanılarak yapılır. Veri seti üzerinde bir nokta olan (x_i, y_i)'nin en çok olabilirlik fonksiyonu üzerindeki etkisi Eşitlik (2.12) ile ifade edilir.

$$\zeta(x_i) = \pi(x_i)^{y_i}[1-\pi(x_i)]^{(1-y_i)} \tag{2.12}$$

"β" parametresinin olabilirlik fonksiyonu olan "$l(\beta)$" Eşitlik (2.13) ile ifade edilir.

$$l(\beta) = \prod_{i=1}^{n} \zeta(x_i) \tag{2.13}$$

En çok olabilirlik tahminlemeleri için, " $l(\beta)$ " değerini maksimum yapan "β" değerleri elde edilmelidir. Buradaki "β" değerleri; " β_0, β_1 " değerlerinin koleksiyonudur. İlgili işlem için takip edilecek adımlar şöyledir:

Adım1: Eşitlik (2.14)'teki gibi "$l(\beta)$" nin logaritması alınır.

$$L(\beta) = \ln[l(\beta)] = \sum_{i=1}^{n} \{y_i \ln[\pi(x_i)] + (1 - y_i)\ln[1 - \pi(x_i)]\} \tag{2.14}$$

Adım2: " β_0 " ve " β_1 " e göre türev alınır, Eşitlik (2.15) ve Eşitlik (2.16)'da görüldüğü gibi türevler sıfıra eşitlenir [27].

$$\sum_{i=1}^{n} [y_i - \pi(x_i)] = 0 \tag{2.15}$$

$$\sum_{i=1}^{n} x_{ik} [y_i - \pi(x_i)] = 0 \quad , \quad k = 1 \tag{2.16}$$

2.3.1.2. Multinominal Lojistik Regresyon

Modelleme:

Tablo 2.8'de de görüldüğü üzere, bağımlı değişken kategori sayısının 2'den büyük olduğu lojistik regresyon problemlerinde multinominal yöntem kullanılır.

Elimizde bağımlı değişken kategori sayısı 3 olan bir problem olduğunu varsayacak olursak, bizi çözüme götürecek yöntem iki ayrı lojistik model oluşturmak olacaktır. "Y" kategorileri "{0, 1, 2}" olarak düşünülecek olursa; " $Y = 0$ " referans kategori olarak alındığında, oluşturulacak modeller şöyle olacaktır:

- $Y = 1$'e karşı $Y = 0$
- $Y = 2$'ye karşı $Y = 0$

Bu yöntem "bire-karşı-diğer (one-vs-all)" olarak bilinir.

Yukarıda ifade edilmiş olan karşılaştırmalara ait lojistik fonksiyonlar Eşitlik (2.17) ve Eşitlik (2.18)'de görüldüğü gibidir.

$$g_1(x) = \ln \frac{P(Y = 1 \mid x)}{P(Y = 0 \mid x)} = \beta_{10} + \beta_{11} x_1 + \beta_{12} x_2 + ... + \beta_{1p} x_p = x' \beta_1 \quad (2.17)$$

$$g_2(x) = \ln \frac{P(Y = 2 \mid x)}{P(Y = 0 \mid x)} = \beta_{20} + \beta_{21} x_1 + \beta_{22} x_2 + ... + \beta_{2p} x_p = x' \beta_2 \quad (2.18)$$

Eşitlik (2.10) ve Eşitlik (2.11) referans alınarak, bağımlı değişken kategorileri bazında koşullu olasılık denklemleri yazılacak olursa Eşitlik (2.19), Eşitlik (2.20) ve Eşitlik (2.21) elde edilecektir. Eşitlikler incelendiğinde, içlerinden herhangi biri kullanılarak bağımsız değişken vektörüne ulaşılabileceği açıkça görülür [28].

$$P(Y = 0 \mid x) = \frac{1}{1 + e^{g_1(x)} + e^{g_2(x)}} \quad (2.19)$$

$$P(Y = 1 \mid x) = \frac{e^{g_1(x)}}{1 + e^{g_1(x)} + e^{g_2(x)}} \quad (2.20)$$

$$P(Y = 2 \mid x) = \frac{e^{g_2(x)}}{1 + e^{g_1(x)} + e^{g_2(x)}} \quad (2.21)$$

Algoritma, bağımlı değişken kategori sayısı "3" olan bir problem üzerinden anlatılmıştır. "3" ten fazla bağımlı değişken içeren problemler için sistem, yukarıda anlatılanla aynı şekildedir.

Genelleştirme yapılacak olursa; bağımlı değişken kategori sayısı "K" olan bir problem için kullanılacak lojistik fonksiyon ve koşullu olasılık denklemleri Eşitlik (2.22), Eşitlik (2.23) ve Eşitlik (2.24)'te görüldüğü gibi olacaktır.

$$g_k(x) = \ln \frac{P(Y = k \mid x)}{P(Y = 0 \mid x)} = \beta_{k0} + \beta_{k1} x_1 + \beta_{k2} x_2 + ... + \beta_{kp} x_p = x' \beta_k, \quad (k = 1,2,3,..., K) \quad (2.22)$$

$$P(Y = 0 \mid x) = \frac{1}{1 + e^{g_1(x)} + e^{g_2(x)} + ... + e^{g_K(x)}}, \quad (k = 1,2,3,..., K) \quad (2.23)$$

$$P(Y = k \mid x) = \frac{e^{g_k(x)}}{1 + e^{g_1(x)} + e^{g_2(x)} + ... + e^{g_K(x)}}, \quad (k = 1,2,3,..., K) \quad (2.24)$$

Parametre Tahmini:

İkili analizde de yer verildiği üzere, lojistik regresyonda parametre tahminlemesi yapılırken en çok olabilirlik yöntemi kullanılır. Bağımlı değişken kategori sayısı "3" olan bir problem üzerinden devam edilecek olursa; multinominal analizdeki fark, veri seti üzerinden alınan bir verinin en çok olabilirlik fonksiyonu üzerindeki etkisini belirlemek için "{0, 1}" değerlerinden birini alabilen "3" değişkenden yararlanılmasıdır. Toplamları "1" olacak şekilde oluşturulan " Y_0, Y_1, Y_2 " değişkenleri, Eşitlik (2.25)'te görüldüğü gibidir.

$$Y_0 \Rightarrow \begin{matrix} Y_0 = 1 \\ Y_1 = 0 \\ Y_2 = 0 \end{matrix} \qquad Y_1 \Rightarrow \begin{matrix} Y_0 = 0 \\ Y_1 = 1 \\ Y_2 = 0 \end{matrix} \qquad Y_2 \Rightarrow \begin{matrix} Y_0 = 0 \\ Y_1 = 0 \\ Y_2 = 1 \end{matrix} \tag{2.25}$$

Elde edilecek olabilirlik fonksiyonu ise Eşitlik (2.26)'teki gibi olacaktır.

$$l(\beta) = \prod_{i=1}^{n} \pi_0(x_i)^{y_{0i}} \, \pi_1(x_i)^{y_{1i}} \, \pi_2(x_i)^{y_{2i}} \tag{2.26}$$

İkili analizde olduğu gibi en çok olabilirlik tahminlemeleri için, "$l(\beta)$" yi maksimum yapan "β" değerleri elde edilmelidir. Takip edilecek adımlar şöyledir:

Adım1: Eşitlik (2.27)'teki gibi "$l(\beta)$" nin logaritması alınır.

$$L(\beta) = \ln[l(\beta)] = \sum_{i=1}^{n} \left\{ y_{1i} g_1(x_i) + y_{2i} g_2(x_i) - \ln\left(1 + e^{g_1(x_i)} + e^{g_2(x_i)}\right) \right\} \tag{2.27}$$

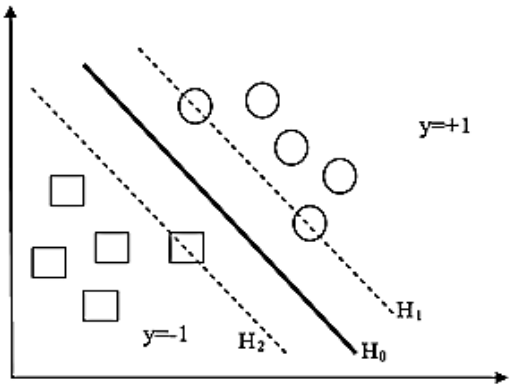
Adım2: "$L(\beta)$" denkleminin birinci kısmi türevleri alınarak genel gösterimi Eşitlik (2.28)'te gösterilen denklemler elde edilir [24]. Türevler sıfıra eşitlenerek "β" parametrelerine ulaşılır.

$$\frac{\partial L(\beta)}{\partial \beta_{jk}} = \sum_{i=1}^{n} x_{ki} \quad (y_{ji} - \pi_{ji}) \quad , \qquad \pi_{ji} = \pi_j(x_i) \tag{2.28}$$

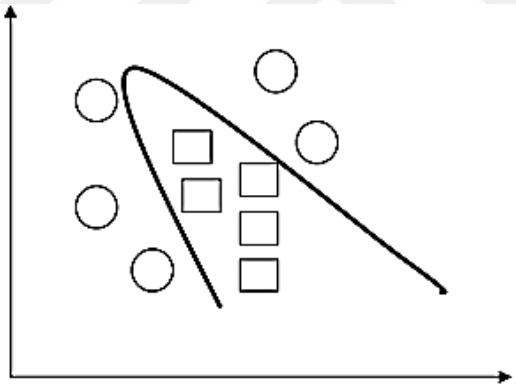
2.3.2. Destek Vektör Makineleri Algoritması (Support Vector Machine Algorithm)

Kompleks sınıflandırma problemlerinin çözümünde en etkili algoritmalardan biri olarak bilinen ve sıkça kullanılan makine öğrenmesi yöntemlerindendir. Destek

Vektör Makineleri (DVM) ilk olarak iki sınıflı doğrusal olarak ayrılabilen verilerin sınıflandırılması için tasarlanmış olup, günümüzde ikiden fazla sınıftan oluşan ve doğrusal olarak ayrılamayan verilerin sınıflandırılmasında kullanılmaktadır. DVM; iki sınıfı en iyi şekilde ayırabilen, sınıflama kararı verebilecek nitelikteki karar fonksiyonunu diğer bir ifadeyle hiper düzlemi tahminlemeye dayanır [29].



Şekil 2.1. Doğrusal ayrılabilen veriler [30]



Şekil 2.2. Doğrusal ayrılamayan veriler [30]

Doğrusal olarak ayrılabilen sınıflarda, " n " toplam sınıf sayısı olmak üzere veri kümesinin " $(x_1, y_1),(x_2, y_2),...,(x_n, y_n)$ " şeklinde ifade edildiği varsayılsın. " $i = 1,2,...,n$ " oluncaya kadar " $y_i \in \{+1, -1\}$ " şeklinde; " x_i " değerlerinin sınıf etiketini tutmaktadır. Doğrusal olarak iki sınıfa ayrılabilen " x_i " değerleri ve bunların iki sınıfa doğrusal olarak ayrıldığını gösteren " H_0 " düzlemi Şekil 2.1'de gösterilmiştir [30].

Bir veri kümesini ayırabilen pek çok hiper düzlem çizilebilir; ancak burada aranan iki sınıfı birbirinden ayıran kendine en yakın noktalar arasındaki uzaklığı maksimum yapan hiper düzlemi bulmaktır. Şekil 2.1'de görülen "H_0", aranan bu optimum hiper düzlemdir; "H_1" ve "H_2" ise destek vektörleri olarak bilinen, sınır genişliğinin belirleyicileri olan vektörlerdir.

"w" hiper düzlemin normali olan ağırlık vektörü ve "b" eğilim değeri olmak üzere destek vektörleri olan "H_1" ve "H_2" Eşitlik (2.29) ile; optimum hiper düzlem olan "H_0" ise Eşitlik (2.30) ve Eşitlik (2.31) ile ifade edilirler [31].

$$\left| w \cdot x_i + b \right| = 1 \tag{2.29}$$

$$w \cdot x_i + b \geq 1 \quad , \quad \forall x_i \in (y = +1) \tag{2.30}$$

$$w \cdot x_i + b \leq 1 \quad , \quad \forall x_i \in (y = -1) \tag{2.31}$$

Doğrusal olarak ayrılabilen veri kümesi içeren iki sınıflı problemler için, birtakım optimizasyon işlemleri sonucunda elde edilen DVM karar fonksiyonu Eşitlik (2.32)'da gösterilmiştir [31]. Burada "λ" Lagrange çarpanıdır; "$*$" ise optimum değerlerin ifadesidir.

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} y_i \lambda_i^* \ (x \cdot x_i) + b^*\right) \tag{2.32}$$

Eldeki veri kümesi, Şekil 2.2' de görüldüğü gibi doğrusal olarak ayrılamayabilir. Böyle sınıflama problemlerinde çözüm, veriyi daha büyük boyutlu bir uzaya taşımaktır. Özellik uzayı olarakta adlandırılan bu yeni uzayda, her bir "x" değişkeni "$\alpha_1\varphi_1(x), \ \alpha_2\varphi_2(x), \ \ldots \ , \alpha_n\varphi_n(x)$" ile ifade edilen "$\varphi(x)$" özellik vektörü halini alır.

Veri uzayı değiştirildikten sonra "$K(x, y) = \varphi(x).\varphi(y)$" ile ifade edilen çekirdek fonksiyonları kullanılarak karar fonksiyonu olan (2.33) elde edilir. Çekirdek fonksiyonları; doğrusal, polinom veya radyal tabanlı olabilir [31].

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} y_i \lambda_i^* \ K(x, x_i) + b^*\right) \tag{2.33}$$

Çok sınıflı DVM problemlerinde, birden fazla ikili DVM kullanılarak veri kümeleri sınıflandırılır. En önemlileri bire-karşı-bir (one-vs-one) ve bire-karşı-diğer (one-vs-

all) olmak üzere [32] çok sınıflı DVM problemleri için önerilen pek çok yöntem vardır. "k" sınıflı bir DVM problemi için:

- BKB' de, "$k(k-1)/2$" adet sınıflandırıcı oluşturulur. İki sınıflı gruplar halinde veri üzerinde sorgulama yapılır. Hangi sınıf çıktısı daha fazla olursa veri o sınıfa daha yakındır denerek eğitim gerçekleşir.
- BKD' de, "k" adet sınıflandırıcı oluşturulur. Her bir sınıf geriye kalan diğer tüm sınıflar ile ikili gruplanır. Tüm sınıflar bitene kadar işlem devam eder ve değerlendirmede olan veri için sonuçlar birleştirilerek karara varılır. Her örnek için geriye kalan tüm veri seti eğitildiğinden BKB' ye göre daha yavaştır ancak bazı araştırmacılar bu yöntemi tercih etmişlerdir [33].

2.3.3. Naive Bayes Algoritması (Naive Bayes Algorithm)

Her özelliğin bağımsız olduğu varsayımı ile çalışan; olasılık tabanlı, kullanımı çokça yaygın bir yöntemdir [34]. Model oluşturmak kolaydır ve büyük veri setleriyle daha iyi çalışır. Öğrenme modeli, veri örneklerinin hangi olasılıkla hangi sınıfa ait olduğuna dayanarak oluşturulur [35].

Naive Bayes, Eşitlik (2.34)'de gösterilen Bayes kuralı ile önsel olasılıklar olarak adlandırılan "$P(x \mid y)$" ve "$P(y)$" değerlerini kullanarak "$P(y \mid x)$" sonsal olasılığını üretir.

$$P(y \mid x) = \frac{P(x \mid y) \;.\; P(y)}{P(x)} \tag{2.34}$$

$$P(y \mid x) = P(x_1 \mid y) \;.\; P(x_2 \mid y) \;.\; P(x_3 \mid y) \;...\; P(x_n \mid y) \;.\; P(y) \tag{2.35}$$

Bayes ile meydana gelme olasılığı birbirinden bağımsız olaylar birlikte incelenir. Eşitlik (2.35)'deki "$x_1, x_2, x_3, ..., x_n$" birbirinden bağımsız olayları göstermektedir. Her bir "y" sınıfı için sonsal olasılık hesabı yapılır; maksimum değer üreten sınıf, verinin sınıf etiketi olarak belirlenir.

Algoritma için gerekli olasılıklar hesaplanırken kullanılan veri seti içeriği önemlidir. Sayısal değerler içeren bir veri setiyle yapılan işlemler, kategorik değerler içeren veri setiyle yapılanlardan biraz farklıdır. Fark şudur: gerekli önsel olasılıklar ilgili özellik değerlerinin ortalama ve standart sapma değerleriyle hesaplanır. Özelliklere ait

değerlerin ortalaması Eşitlik (2.36) ile, standart sapması ise Eşitlik (2.37) ile elde edilir. Kullanılan önsel olasılık formülüzasyonu ise Eşitlik (2.38)'te gösterilmiştir.

$$\mu = \frac{1}{n}\sum_{n=1}^{n} x_i \tag{2.36}$$

$$\sigma = \frac{1}{n-1}\sum_{n=1}^{n}(x_i - \mu)^2 \tag{2.37}$$

$$P(x \mid y) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{\sigma^2}} \tag{2.38}$$

2.3.4. Rastgele Orman Algoritması (Random Forest Algorithm)

Rasgele Orman algoritması, yaygın olarak kullanılan toplu sınıflandırma yöntemlerinden birisidir. Toplu sınıflandırma yöntemlerinde, tek bir sınıflandırıcı yerine birden çok sınıflandırıcı oluşturulur. Yeni gelen bir verinin sınıfı tahmin edilmek istendiğinde, eğitim aşamasında üretilen sınıflandırıcıların her birinden, yeni veri sınıfına dair tahminleme sonucu elde edilir. En çok oyu alan sınıf, öğrenme algoritmasının ilgili veri için ürettiği tahmini sınıf çıktısı olur [36].

Rastgele Orman algoritması için yukarıda bahsi geçen sınıflandırıcılar, ağaç tipi sınıflandırıcılardır. Algoritma tek bir karar ağacı üretmek yerine her biri farklı eğitim kümelerinde eğitilmiş olan çok sayıda çok değişkenli ağacın kararlarını birleştirir. Araştırmacılar tarafından *"ağaç tipi sınıflandırıcılar topluluğu"* [36, 37] olarak tanımlanmasının sebebi de budur.

Hem hızlı olması hem de yüksek doğruluk sağlaması yönünden diğer toplu sınıflandırma yöntemlerinin önüne geçmektedir. Yeterliliği ve doğruluğu ile çok kullanışlı bir sınıflandırıcı olarak bilinir [38].

Modelleme:

Rastgele Orman algoritmasında, model oluşturma adımları şöyledir:

Adım1: Eğitim veri setinden, yeni bir eğitim verisi üretilir. Bu üretim işlemi, eğitim verisi üzerinden gerçekleştirilen yer değiştirmeli seçimlerle olur.

"T" eğitim veri setinin tamamı, "T_k" ise oluşturulacak ağaç için üretilen yeni eğitim verileri olsun. "T_k" nın "$2/3$" ü geliştirilecek karar ağacı için önyükleme örnekleri olur. Kalan "$1/3$" lük kısım ise OOB verileri olarak kullanılacaktır. Ağaç oluşturulduğunda OOB verileri kullanılarak sınıflandırıcı test edilip hatalar hesaplanacaktır.

Adım2: Algoritma işletilmeye başlanmadan önce kullanıcı tarafından iki parametre tanımlanmalıdır [39]:

- "m", ağaç yapısında en iyi bölünmeyi gerçekleştirmek için her bir düğümde kullanılan değişken sayısıdır. Sınıflandırma işlemini etkileme yönünden önemli olan değişkenlerin sayısı anlamına gelen "m", algoritma hesabını sadeleştirmek ve ağaçların birbiriyle olan ilişkisini azaltmak amaçlı seçilen bir parametredir. Başlangıçta kullanıcı tarafından rastgele seçilen "m" değeri, "yüksek m, yüksek ilişki; düşük m düşük ilişki" kuralı esas alınarak, ortaya çıkan genelleştirilmiş hataya (OOB) göre artırılır veya azaltılır. Burada amaç, ağaçlar arası ilişkiyi en aza indirgeyen optimum "m" değerine ulaşmaktır. En uygun "m" en az hata, bu da en iyi sınıflandırıcı demektir. Araştırmacılar "m" değerini toplam değişken sayısının karekökü olarak seçtiklerinde genellikle en iyi sonuca ulaştıklarını söylemektedirler [40].
- "N", oluşturulacak ağaç sayısıdır.

Adım3: Üretilen yeni eğitim verisinden, rastgele özellik seçimi yöntemiyle bir karar ağacı oluşturulur. Ağaçlarda budama yapılmaz [41, 42]. Budama yapılmaması, algoritmaya diğer ağaç tipi sınıflandırıcılara göre daha performanslı bir sınıflandırıcı olma özelliği kazandırır.

Rastgele Orman algoritması, karar ağacı oluşturmak için CART algoritmasını kullanılır [42]. CART algoritması, ağaç düğümlerine karar verirken bazı bölünme algoritmalarından faydalanır. Rastgele Orman, bu bölünme algoritmalarından "Gini" yi kullanır [37].

Gini algoritması kullanıldığında, CART algoritmasının işleyişi şöyle olacaktır [30]:
1. Her bir özellik, içerdiği kategoriler bazında sağ ve sol olmak üzere iki gruba ayrılır. Oluşturulan her bir grup, sınıf değerleri bazında kendi içinde yeniden gruplanır.

2. Her bir özellik için oluşturulan sağ ve sol gruplarına ait Gini değerleri Eşitlik (2.39) ve Eşitlik (2.40) ile hesaplanır. Gini eşitliklerinde görülen parametrelerin karşılıkları şöyledir:

k : sınıf sayısı

R_i : sağ tarafta yer alan " i " kategorisindeki veri sayısı

L_i : sol tarafta yer alan " i " kategorisindeki veri sayısı

$T_{sağ}$: sağ taraftaki veri sayısı

T_{sol}: sol taraftaki veri sayısı

$$Gini_{sağ} = 1 - \sum_{i=1}^{k} \frac{R_i}{|T_{sağ}|}^2 \quad (2.39)$$

$$Gini_{sol} = 1 - \sum_{i=1}^{k} \frac{L_i}{|T_{sol}|}^2 \quad (2.40)$$

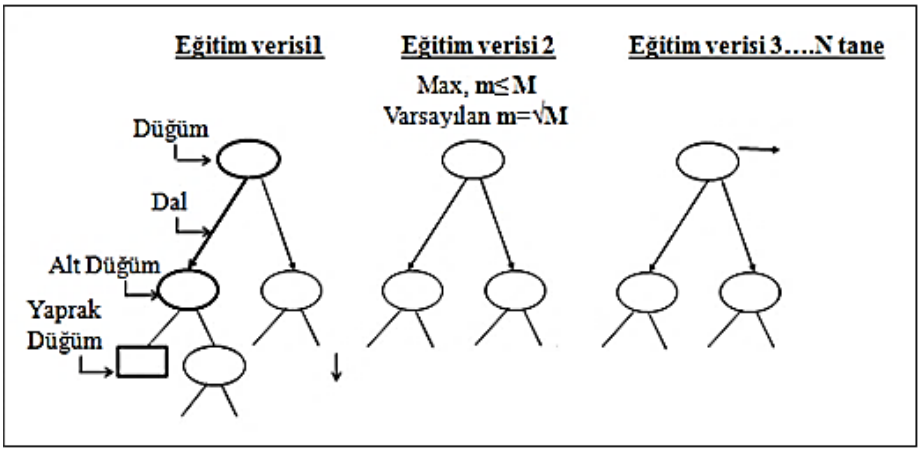
3. Her bir " j " özelliği için Eşitlik (2.41) kullanılarak "$Gini_j$" değerleri hesaplanır. Eşitlikte görülen " n ", kullanılan veri sayısıdır.

$$Gini_j = \frac{1}{n}\left(|T_{sağ}| \cdot Gini_{sağ} + |T_{sol}| \cdot Gini_{sol}\right) \quad (2.41)$$

4. "$Gini_j$" değerleri arasından en küçük olan hangi özelliğe aitse, düğüme o özellik yerleştirilir.

5. Durma koşulu, Gini değerinin " 0 " olmasıdır. Gini değeri " 0 " olan dallar sonlandırılır; olmayanlar için, yukarıdaki işlem adımları durma koşulu sağlanana kadar tekrarlanır.

Adım4: Rastgele Orman algoritması işlem adımları 1, 2 ve 3; kullanıcının tanımladığı "N" değeri kadar tekrar edilir. Bu döngüsel durum Şekil 2.3'te görülmektedir.

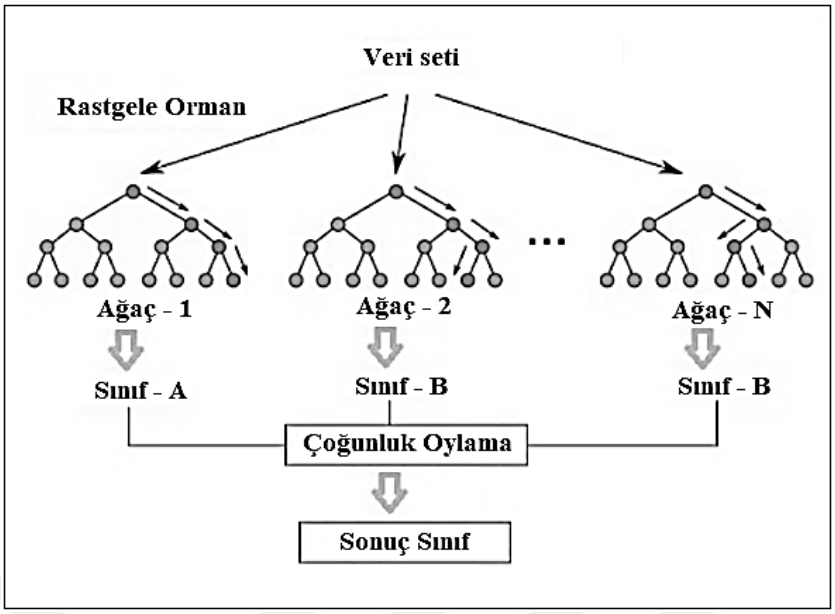


Şekil 2.3. Rastgele Orman algoritması ağaç yapısı [37]

Her işlem döngüsünün bitiminde, üretilen sınıflandırıcı test edilir. Bu test işlemi için, sınıflandırıcının üretiminde kullanılan veriler kullanılmaz, bunların dışındaki eğitim verileri kullanılır [42]. Adım1'de ayrılan OOB verileri ile tahminleme gerçekleştirilir. Yapılan tahminleme sonucunda genelleştirilmiş hata (OOB) elde edilir. Bu hata değeri, sınıflandırıcıda kullanılan değişkenlerin sınıflandırıcı için olan etkisinin de bir göstergesi niteliğindedir.

Sınıflandırma:

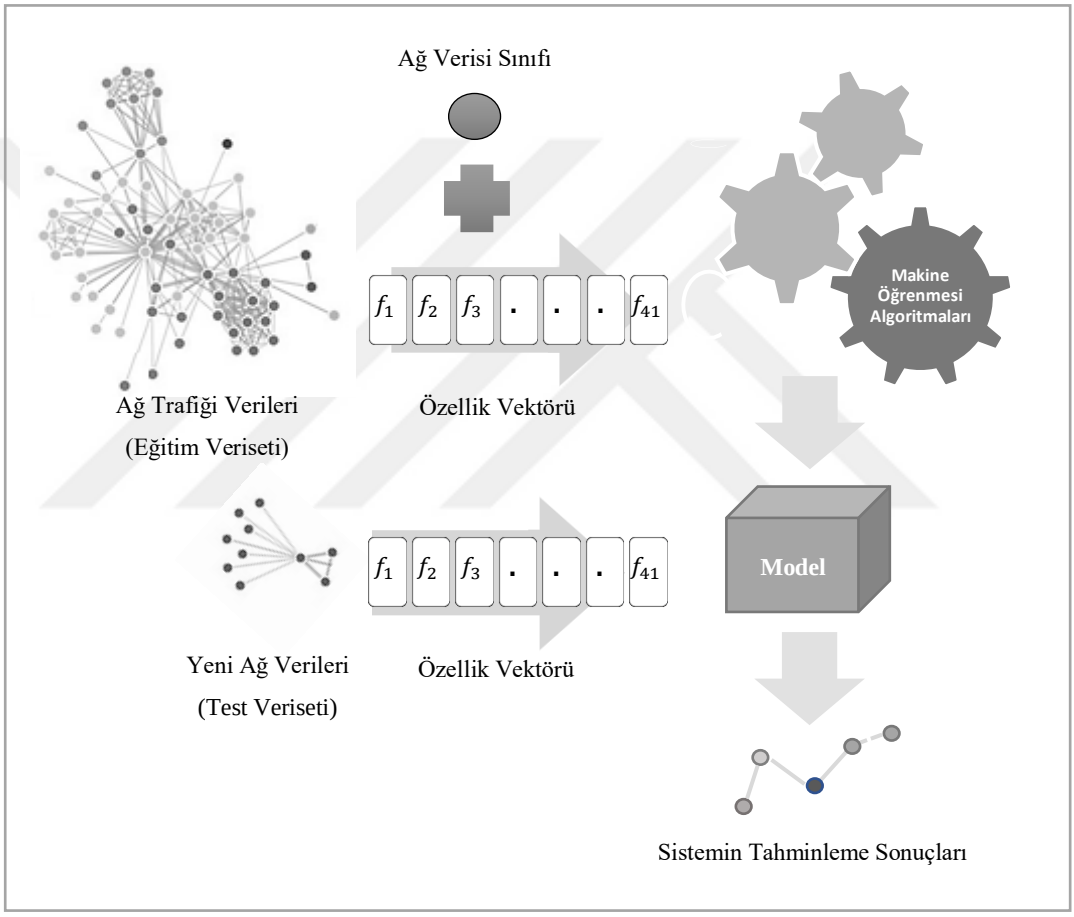
Rastgele Orman algoritmasının sınıflandırma yaparken kullandığı karar mekanizması Şekil 2.4'teki gibi özetlenebilir. İşleyiş şöyledir; yeni gelen bir veriye ait sınıf tahmini yapılırken öncelikle gelen veri, modelleme aşamasında oluşturulan ormandaki her bir ağaca verilir. Her bir ağaçta üretilen sınıf çıktıları alınır. En çok oyu alan sınıf, gelen verinin sınıfı olarak belirlenir.



Şekil 2.4. Rastgele Orman algoritması sınıflandırma karar mekanizması [43]

3. AĞ SALDIRISI TESPİT SİSTEMİ MİMARİSİ

Çalışmada geliştirilen ağ saldırısı tespit sistemine ait mimari, Şekil 3.1'de de görüldüğü gibi genel olarak eğitim ve tahminleme aşamalarından oluşmaktadır. Sistemden beklenen; öğrendiği saldırıları tespit etmesidir.



Şekil 3.1. Ağ saldırı tespit sistemi mimarisi

Sistem mimarisi şu şekilde özetlenebilir:

- Eğitim için belirlenen veriler makine öğrenmesi algoritmaları olan Lojistik Regresyon, Destek Vektör Makineleri, Naive Bayes ve Rastgele Orman algoritmalarına sürülür.
- Her bir algoritmaya ait model oluşur.
- Tahminleme için ayrılan sınıf etiketsiz veriler, oluşan modellere verilir ve her

algoritmanın tahminleme çıktıları elde edilir.

- Tahminleme çıktıları ile algoritmanın doğruluk oranı tespit edilir.

Sistem geliştirilirken gerçekleştirilen işlemler temel olarak 3 başlık altında incelenebilir: veri önişleme, eğitim, tahminleme.

3.1. Veri Önişleme

Adım1: Eldeki veriler; ilgili problemi tanımlamada etkili olacak özellikler çıkarılarak oluşturulan özellik vektörü ve yine veriden elde edilen sınıf etiketi şeklinde iki gruba ayrılır (tahminleme verilerinde sınıf etiketi yoktur).

Adım2: Kategorik özellikler sayısallaştırılır. Verideki özellikler tümüyle sayısal forma dönüştürülür.

Adım3: Eldeki veriler; %60'ı eğitim, %40'ı tahminleme verisi olacak şekilde ayrılır. Oluşturulan tahminleme verileri sınıf etiketine sahip değilken, eğitim verileri sahiptir.

3.2. Eğitim

Makinenin eğitildiği aşamadır. Bu aşamada yeterli miktarda ve nitelikte veri kullanılması ve probleme uygun algoritma seçimi son derece önemlidir.

Makine, sürülen verilerden elde ettiği bilgiler ve kullanması için belirlenen makine öğrenmesi algoritması kuralları ile kendisine bir model oluşturur. Başka bir ifadeyle; sınıf etiketi olmayan bir veri makineye verildiğinde, verinin sınıfı üzerinde tahminlemede bulunabileceği nitelikteki veri tipiyle ile ilgili öğrendiği bilgilerin tamamına model denilir.

Eğitim aşamasının çıktısı, algoritma bazında oluşan modellerdir.

3.3. Tahminleme

Makine, sınıf etiketi olmayan bir veri ile karşılaştığında; bu veri tipiyle ilgili belirlenen algoritma kurallarını kullanarak eğitim aşamasında oluşturduğu model üzerinde çalışmaya başlar ve önceden öğrendiği bilgilerle yeni verinin sınıfını tahminlemeye çalışır.

Tahminleme aşamasının çıktısı, makinenin yeni verinin sınıfı konusunda yaptığı tahminlemedir; yani tahmini sınıf etiketidir.

4. GELİŞTİRİLEN AĞ SALDIRISI TESPİT SİSTEMİ YAZILIMI DETAYLARI

Geliştirilen saldırı tespit sistemi yazılımı, Ubuntu işletim sistemli bir makine üzerinde koşturulmuştur. Geliştirmeler, Apache Spark Makine Öğrenmesi Kütüphanesi (Mllib)'ne ait DataFrame tabanlı API kullanılarak yapılmıştır.

4GB executer ve driver belleğiyle ile konfigüre edilen Apache Spark'a ait bilgiler şöyledir:

- Spark versiyonu: 2.2.1 (Dec 01 2017)
- Paket tipi: Pre-built for Apache Hadoop 2.7 and later

Deneysel veri seti olarak, ağ saldırısı tespit sistemlerinde çokça kullanılan KDD Cup'99 veri seti kullanılmıştır. KDD Cup'99, yaklaşık 5 milyon bağlantı verisi içermektedir. Veri setinde yer alan bağlantı verileri, %60-%40 oranında ayrılarak eğitim ve tahminleme verisi olarak kullanılmıştır.

Kullanılan makine öğrenmesi algoritmaları ve parametreleri ise şöyledir:

- Lojistik Regresyon, önerilen overfitting parametreleri ile bire-karşı-diğer (one-vs-all) yöntemi kullanılarak uygulanmıştır.
- Destek Vektör Makineleri, önerilen overfitting parametreleri ve doğrusal çekirdek fonksiyonu (linear kernel) ile bire-karşı-diğer (one-vs-all) yöntemi kullanılarak uygulanmıştır.
- Naive Bayes, önerilen parametreler kullanılarak uygulanmıştır.
- Rastgele Orman, rastgele orman ağaç sayısı (NumTrees) "300", ağaçlar oluşturulurken kullanılan homojenlik ölçütü (Impurity) "gini", her düğümdeki bölmeler için dikkate alınması gereken özellik sayısı (FeatureSubsetStrategy) "sqrt" olarak uygulanmıştır.

5. SONUÇLAR VE ÖNERİLER

Rastgele seçimlerle oluşturulan eğitim ve tahminleme verileri kullanılarak elde edilen algoritma bazlı yazılım çıktıları, ağırlıklı doğruluk ve performans metrikleri açısından Tablo 5.1'de; işlem süreleri açısından ise Tablo 5.6'da yer almaktadır. Tablo 5.2, Tablo 5.3, Tablo 5.4 ve Tablo 5.5'te ise kullanılan makine öğrenmesi algoritmalarının, performans metrikleri üzerinden sınıf bazlı değerlendirme sonuçları bulunmaktadır.

Kullanılan makine öğrenmesi algoritmalarından elde edilen sonuçların; ağırlıklı doğruluk ve performans metrik değerleri, eğitim süresi, tahminleme süresi, eğitim-tahminleme süreleri arasındaki fark açısından karşılaştırmalı değerlendirmeleri şöyledir:

- Doğruluk ve performans metrikleri:

Lojistik Regresyon 〉Rastgele Orman 〉 Destek Vektör Makineleri 〉 Naive Bayes

- Eğitim süresi:

Lojistik Regresyon 〉 Destek Vektör Makineleri 〉 Rastgele Orman 〉 Naive Bayes

- Tahminleme süresi:

Rastgele Orman 〉 Destek Vektör Makineleri 〉 Lojistik Regresyon 〉 Naive Bayes

- Eğitim-tahminleme süreleri arasındaki fark:

Lojistik Regresyon 〉 Destek Vektör Makineleri 〉 Rastgele Orman 〉 Naive Bayes

Tablo 5.1. Makine öğrenmesi algoritmaları bazında doğruluk ve performans metrikleri açısından karşılaştırmalı yazılım çıktısı analizi

Algoritma	Doğruluk	Precision	Recall	F-Measure
Lojistik Regresyon	0,991	0,989	0,995	0,992
Destek Vektör Makineleri	0,939	0,928	0,939	0,933
Naive Bayes	0,809	0,914	0,809	0,858
Rastgele Orman	0,980	0,972	0,985	0,978

Tablo 5.1'de yer alan ağırlıklı doğruluk ve performans metrikleri sonuçları değerlendirilecek olursa; en yüksek doğru tahminleme oranlarının sırasıyla Lojistik Regresyon, Rastgele Orman ve Destek Vektör Makineleri algoritmalarına ait olduğu görülecektir.

Çalışma sonuçlarının önceki çalışmalarla kıyaslanabilmesi açısından deneysel testler için, ağ saldırısı tespit sistemlerini doğrulamada oldukça sık kullanılan bir veri seti olan KDD Cup'99 veri seti kullanılmıştır. KDD Cup'99 veri setinin simülasyon yapısı oldukça iyi tasarlandığından, Tablo 5.1'de de görüldüğü üzere yazılım sonuçları çok yüksek doğruluk oranlarına sahip çıkmıştır. Çıktılar her ne kadar sonuçların gerçeğe yakınlığı anlamında düşündürücü olsa da, yapılan testler seçilen makine öğrenmesi algoritmalarının büyük ağ verileri üzerindeki davranışlarını oransal olarak görmeyi sağlamıştır. Diğer yandan, rastgele seçimlerle oluşturulan eğitim - test veri seti içeriklerinde yer alan sınıf tiplerinin değişiklik göstermesi ile bazı sınıfların çok fazla veri üzerinden öğrenilmesinin doğruluk oranını artırabileceği düşünülmüştür. Bu durum; makine tarafından az sayıda veriyle yetersiz şekilde öğrenilen veya eğitim veri setinde yer almadığı için hiç öğrenilemeyen, dolayısıyla da doğru tahmin edilemeyen sınıfları gözden kaçırmaya sebep olabileceğinden sonuçları daha yorumlanabilir hale getirmek amacıyla ağırlıklı ortalamaların yanı sıra, sınıf bazında filtrelemelerle de değerlendirmeler yapılmıştır. Sonuçları pozitif ve negatif yönde maksimum derecede etkileyen en uç değerlere sahip performans metrikleri, ait oldukları sınıflarla birlikte algoritma bazlı olarak Tablo 5.2, Tablo 5.3, Tablo 5.4 ve Tablo 5.5'te raporlanmıştır.

Tablo 5.2. Lojistik Regresyon algoritması sınıf bazlı performans metrikleri

Sınıf Etiketi	Precision	Recall	F-Measure
smurf	1,00000	0,99999	0,99999
teardrop - phf	1,00000	1,00000	1,00000
pod	1,00000	0,98214	0,99099
guess_passwd	0,87500	1,00000	0,93333
rootkit - loadmodule - ftp_write - multihop - perl	0,00000	0,00000	0,00000

Lojistik Regresyon algoritmasına ait sınıf bazlı performans metrikleri yorumlanacak olursa; Tablo 5.2'de de görüldüğü üzere makine özellikle Smurf, Teardrop, Phf, Pod, Guess password saldırılarını çok yüksek oranda doğru tahmin etmiştir. Bu değerler, algoritma tahminlemelerinin doğruluk ortalamasını yükseltme yönünde oldukça etkili olacaktır. Diğer yandan; Rootkit, Loadmodule, Ftp-write, Multihop, Perl saldırıları ise makine tarafından hiçbir şekilde tahminleme sonucu olarak ileri sürülmemiştir.

Tablo 5.3. Destek Vektör Makineleri algoritması sınıf bazlı performans metrikleri

Sınıf Etiketi	Precision	Recall	F-Measure
teardrop	1,00000	1,00000	1,00000
pod	1,00000	0,98214	0,99099
guess_passwd	0,87500	1,00000	0,93333
buffer_overflow - warezmaster - rootkit - loadmodule - ftp_write - multihop - phf	0,00000	0,00000	0,00000

Destek Vektör Makineleri algoritmasına ait sınıf bazlı performans metrikleri yorumlanacak olursa; Tablo 5.3'te de görüldüğü üzere makine özellikle Teardrop, Pod, Guess password saldırılarını çok yüksek oranda doğru tahmin etmiştir. Bu değerler, algoritma tahminlemelerinin doğruluk ortalamasını yükseltme yönünde oldukça etkili olacaktır. Diğer yandan; Buffer overflow, Warezmaster, Rootkit, Loadmodule, Ftp-write, Multihop, Phf saldırıları ise makine tarafından hiçbir şekilde tahminleme sonucu olarak ileri sürülmemiştir.

Tablo 5.4. Naive Bayes algoritması sınıf bazlı performans metrikleri

Sınıf Etiketi	Precision	Recall	F-Measure
portsweep - ftp_write - multihop	0,00000	0,00000	0,00000
guess_passwd	0,00013	1,00000	0,00026
rootkit	0,00086	1,00000	0,00171
phf	0,00001	1,00000	0,00003

Naive Bayes algoritmasına ait sınıf bazlı performans metrikleri yorumlanacak olursa; Tablo 5.4'te de görüldüğü üzere makine özellikle Guess password, Rootkit, Phf saldırılarını yüksek oranda doğru tahmin etmiştir. Bu değerler, algoritma tahminlemelerinin doğruluk ortalamasını yükseltme yönünde oldukça etkili olacaktır. Diğer yandan; PortSweep, Ftp-write, Multihop saldırıları ise makine tarafından hiçbir şekilde tahminleme sonucu olarak ileri sürülmemiştir.

Tablo 5.5. Rastgele Orman algoritması sınıf bazlı performans metrikleri

Sınıf Etiketi	Precision	Recall	F-Measure
smurf	1,00000	0,99998	0,99999
portsweep	1,00000	0,92938	0,96339
back - warezclient - teardrop - guess_passwd - buffer_overflow - land - warezmaster - rootkit - loadmodule - ftp_write - multihop - phf - perl	0,00000	0,00000	0,00000

Rastgele Orman algoritmasına ait sınıf bazlı performans metrikleri yorumlanacak olursa; Tablo 5.5'te de görüldüğü üzere makine özellikle Smurf, PortSweep saldırılarını çok yüksek oranda doğru tahmin etmiştir. Bu değerler, algoritma tahminlemelerinin doğruluk ortalamasını yükseltme yönünde oldukça etkili olacaktır. Diğer yandan; Back, Warezclient, Teardrop, Guess password, Buffer overflow, Land, Warezmaster, Rootkit, Loadmodule, Ftp-write, Multihop, Phf, Perl saldırıları ise makine tarafından hiçbir şekilde tahminleme sonucu olarak ileri sürülmemiştir.

Tablo 5.6. Makine öğrenmesi algoritmaları bazında eğitim ve tahminleme süreleri açısından karşılaştırmalı yazılım çıktısı analizi

Algoritma	Eğitim Süresi (h)	Tahminleme Süresi (h)
Lojistik Regresyon	4,041	0,089
Destek Vektör Makineleri	1,419	0,092
Naive Bayes	0,016	0,026
Rastgele Orman	0,412	0,297

Eğitim-tahminleme süreleri arasındaki farklar Tablo 5.6'da yer alan sonuçlar üzerinden değerlendirilecek olursa, diğer algoritmalarla karşılaştırıldığında Lojistik Regresyon algoritmasının en yüksek süre farkına sahip olduğu açıkça görülmektedir. Bu durum, Lojistik Regresyon algoritmasının ağa yapılan saldırıları zamanında tespit etme konusundaki uygunluğu üzerinde şüphe uyandırmaktadır.

Tablo 5.7. Önceki saldırı tespit sistemi ile geliştirilen sistem arasında performans karşılaştırması

Algoritma	Doğruluk	
	Geliştirilen Sistem	Önceki Sistem [3]
Lojistik Regresyon	0,991	0,916
Destek Vektör Makineleri	0,939	0,921
Naive Bayes	0,809	0,914
Rastgele Orman	0,980	0,921

Geliştirilen sistemle elde edilen sonuçlar, benzer teknolojiler kullanılarak tasarlanan diğer ağ saldırısı tespit sistemi çalışmalarıyla karşılaştırmalı olarak değerlendirildiğinde; büyük ağ verileri üzerindeki saldırıları tespit etmede Apache Spark'ın çıkan yeni sürümleriyle giderek daha etkili bir hale geldiği görülmektedir. Tablo 5.7'de daha önceden geliştirilen bir saldırı tespit sistemi çalışması [3] ile yapılan performans karşılaştırması yer almaktadır. Karşılaştırmadan da görüldüğü üzere; Naive Bayes algoritması haricindeki bahsi geçen makine öğrenmesi algoritmaları ile, kıyaslama yapılan önceki sistemden daha doğru tahminlemelerde bulunulmuştur. Naive Bayes algoritması olasılık tabanlı bir algoritma olduğundan, diğer algoritmalara kıyasla; eğitim-test veri seti seçimlerinin rastgele yapılmasından

ortaya çıkan olasılık dağılımı değişimlerinden daha fazla etkilenir. Rastgele seçilen verilerle gerçekleştirilen testlerde Naive Bayes algoritmasının doğru tahminleme oranının, diğer algoritmalara kıyasla daha yüksek iniş-çıkış oranına sahip olduğu görülmüştür. Bu durumun sonucu olarak; Tablo 5.7'de görülen doğru tahminleme oranı düşüşünün, algoritma yapısından kaynaklı olduğu kanısına varılmıştır.

Çalışmada yapılanlar özetlenecek olursa; kıyaslama yapılabilmesi açısından literatürdeki benzer çalışmalarda kullanılan algoritmalar ve veriler ile çalışılmıştır.

Hem çalışmadan elde edilen sonuçlarla makine öğrenmesi algoritmalarının ağ verileri üzerindeki saldırıları tespit etme performansları çeşitli açılardan değerlendirilmiş; hem de literatür araştırmalarından elde edilen sonuçlar gerçekleştirilen çalışmanın sonuçlarıyla karşılaştırılmıştır. Bu karşılaştırma, büyük veri teknolojisi olan Apache Spark'ın büyük ağ verileri üzerindeki saldırıları tespit etmede giderek daha da etkili bir hale geldiğini göstermiştir.

Çalışma yapılabilecek geliştirmeler açısından değerlendirilecek olursa; geliştirilen yazılım, farklı yapay zekâ yöntemleri (Derin Öğrenme vb.) üzerinde farklı parametrelerle koşturulup karşılaştırmalı analizler yapılabilir. Özellikle eğiticisiz sınıflandırma algoritmaları ile daha önceden karşılaşmadığı saldırıları da tespit edebilecek ağ saldırısı tespit sistemleri geliştirilebilir.

Deneysel testlerde kullanılan veri seti, oldukça iyi tasarlanmış bir simülasyon yapısına sahiptir ve eski bir veri seti olduğu için güncel saldırı tiplerini kapsamamaktadır. Bu sebeplerden ötürü, geliştirilen sistemin gerçek bir ortamda aynı sonuçları verememe durumu söz konusudur. Bu durumun çözümü olarak güncel saldırıları da kapsayan, yapısal olarak gerçeğe daha yakın veri setleri hazırlanabilir. Algoritma bazında yapılan karşılaştırmalı analizler, geliştirilen yeni veri setleriyle de gerçekleştirilebilir. Ancak tam anlamıyla stabil yeni bir veri seti üretme işlemi, oldukça maliyetli bir işlemdir. Bunun yanı sıra, veri seti açısından literatürdeki benzer çalışmalarla aynı deneysel koşullara sahip olunamayacağından geçmişteki çalışmalarla tam anlamıyla gerçekçi bir kıyaslama yapılamayacaktır.

Hız iyileştirmesi için, veri setinde yer alan bağlantılara ait özellikler (41 tane) analiz edilip sınıflandırmada etkisi az olanlar çıkartılabilir.

Gelecekte yapılması düşünülen çalışmalar vardır. Geliştirilen sistem, ağ saldırısı tespit sistemlerine giriş niteliğindedir. Ağ saldırısı ve büyük veri araştırmacıları için referans kaynak sağlamaktadır. Çalışmanın devamı olarak planlanan bir sonraki adım; veri setindeki sınıflamaya etkisi az olan özelliklerin farklı yöntemlerle tespit edilerek çıkarılması ve sonuçların mevcut geliştirme sonuçları ile karşılaştırılmasıdır.

KAYNAKLAR

[1] Çevik M., Intrusion Detection with Pattern Classification, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2005, 166493.

[2] Becerikli Y., İleri Örüntü Tanıma Teknikleri ve Uygulamaları Ders Notu, *Kocaeli Üniversitesi*, Lecture1, 19-25, 2016.

[3] Gupta G. P., Kulariya M., A Framework for Fast and Efficient Cyber Security Network Intrusion Detection Using Apache Spark, *Procedia Computer Science*, 2016, **93**, 824-831.

[4] Siddique K., Akhtar Z., Lee H. G., Kim W., Toward Bulk Synchronous Parallel-Based Machine Learning Techniques for Anomaly Detection in High-Speed Big Data Networks, *Symmetry*, 2017, **9**(9), 197.

[5] Harifi S., Byagowi E., Khalilian M., Comparative Study of Apache Spark MLlib Clustering Algorithms, *Second International Conference on Data Mining and Big Data*, Fukuoka, Japan, 27 July-1 August 2017.

[6] Jeong H. D. J., Jeong G. S., Kim W. J., Kim J., Song H., Ryu M. U., Lee J. R., A Search for Computationally Efficient Supervised Learning Algorithms of Anomalous Traffic, *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Torino, Italy, 10-12 July 2017.

[7] Oh S. W., Kim H. S., Lee H. S., Kim S. J., Park H., You W., Study on the Multi-Modal Data Preprocessing for Knowledge-Converged Super Brain, *International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea, 19-21 October 2016.

[8] Orhan M., İnternet Nedir ?, Web Sitesi, https://www.mustafaorhan.com/genel/internet-nedir/, (Ziyaret tarihi: 27 Aralık 2018).

[9] Aksu A., Ağ Çeşitleri, ODTÜ Bilgisayar Topluluğu Elektronik Dergisi (e-bergi), http://e-bergi.com/y/ag-esitleri (Ziyaret tarihi: 27 Kasım 2018).

[10] Apache Spark, Lightning-Fast Unified Analytics Engine, Web Sitesi, https://spark.apache.org/ (Ziyaret tarihi: 27 Kasım 2018).

[11] Sathya S. S., Ramani R. G., Sivaselvi K., Discriminant Analysis Based Feature Selection in KDD Intrusion Dataset, *International Journal of Computer Applications*, 2011, **31**(11), 1-7.

[12] Tavallaee M., Bagheri E., Lu W., Ghorbani A. A., A Detailed Analysis of the KDD Cup 99 Dataset, *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, Ottawa, Canada, 8-10 July 2009.

[13] Çel Ö., Dos Attack Mitigation in IOT, Yüksek Lisans Tezi, Boğaziçi Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2018, 523080.

[14] Özçelik İ., DoS Attack Detection and Mitigation, PhD Thesis, Clemson University, Electrical and Computer Engineering, Clemson, South Carolina, USA, 2015.

[15] Carl G., Kesidis G., Brooks R. R., Rai S., Denial-of-Service Attack-Detection Techniques, *IEEE Internet Computing*, 2006, **10**(1), 82-89.

[16] CERT Advisory CA-98.01, CERT Advisory CA-98.01 "smurf" IP Denial-of-Service Attacks, Web Sitesi, http://www.aoc.nrao.edu/~krowe/TCC/sysadmin/topics/security/CA-98.01.smurf.html (Ziyaret tarihi: 24 Aralık 2018).

[17] Kendall K. K. R., A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems, PhD Thesis, Massachusetts Institute of Technology, Cambridge, USA, 1999.

[18] Polat O., Yazılım Tanımlı Ağlarda DoS Saldırılarının Tespiti ve Engellenmesi, Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2017, 472809.

[19] Denial of Service Attacks, Web Sitesi, https://www.cs.utexas.edu/users/chuang/dos.html (Ziyaret tarihi: 24 Aralık 2018).

[20] Ghorbani A. A., Lu W., Tavallaee M., *Network Intrusion Detection and Prevention: Concepts and Techniques*, 1st ed., Springer Science & Business Media, New York, USA, 2010.

[21] Sabhnani M., Serpen G., KDD Feature Set Complaint Heuristic Rules for R2L Attack Detection, *The 2003 International Conference on Security and Management (SAM '03)*, Las Vegas, Nevada, USA, 23-26 June 2003.

[22] Knowledge Extraction Based on Evolutionary Learning (KEEL), KDD Cup (Imbalanced: Rootkit-Imap vs Back) Data Set, Web Sitesi, https://sci2s.ugr.es/keel/dataset.php?cod=1320, (Ziyaret tarihi: 27 Aralık 2018).

[23] Olusola A. A., Oladele A. S., Abosede D. O., Analysis of KDD'99 Intrusion Detection Dataset for Selection of Relevance Features, *Proceedings of the World Congress on Engineering and Computer Science (WCECS)*, San Francisco, USA, 20-22 October 2010.

[24] Hosmer D. W., Lemeshov S., *Applied Logistic Regression*, 2nd ed., John Wiley & Sons., New York, 2000.

[25] Şenel S., Alatlı B., Lojistik Regresyon Analizinin Kullanıldığı Makaleler Üzerine Bir İnceleme, *Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi*, 2014, **5**(1).

[26] Everitt B. S., Dunn G., *Applied Multivariate Data Analysis*, 2nd ed., Wiley, Chichester, 2010.

[27] Rush S., Logistic Regression: The Standard Method of Analysis in Medical Research, *Trinity University*, Mathematics Technical Report #S3, 2001.

[28] Powers D., Xie Y., *Statistical Methods for Categorical Data Analysis*, 1st ed., Emerald Group Publishing, United Kingdom, 2008.

[29] Vapnik V., *The Nature of Statistical Learning Theory*, 2nd ed., Springer-Verlag, New York, 2000.

[30] Özkan Y., *Veri Madenciliği Yöntemleri*, 3.Basım, Papatya Bilim Yayınevi, İstanbul, 2016.

[31] Osuna E. E., Support Vector Machines: Training and Applications, PhD Thesis, Massachusetts Institute of Technology, Cambridge, USA, 1998.

[32] Pöyhönen S., Support Vector Machine Based Classification in Condition Monitoring of Induction Motors, PhD Thesis, Helsinki University of Technology, Espoo, Finlandiya, 2004.

[33] Hasan M. A. M., Nasser M., Pal B., Ahmad S., Support Vector Machine and Random Forest Modeling for Intrusion Detection System (IDS), *Journal of Intelligent Learning Systems and Applications*, 2014, **6**(01), 45.

[34] Trevor H., Robert T., JH F., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer-Verlag, New York, 2009.

[35] İlhan Omurca S., Makine Öğrenmesi Ders Notu, *Kocaeli Üniversitesi*, Bayesian Learning, 7-43, 2016.

[36] Akar Ö., Güngör G., Rastgele Orman Algoritması Kullanılarak Çok Bantlı Görüntülerin Sınıflandırılması, *Jeodezi ve Jeoinformasyon Dergisi*, 2012, **1**(2), 139-146.

[37] Ok A., Akar Ö., Güngör O., Rastgele Orman Sınıflandırma Yöntemi Yardımıyla Tarım Alanlarındaki Ürün Çeşitliliğinin Sınıflandırılması, *TUFUAB 2011 VI. Teknik Sempozyumu*, Antalya, 23-26 Şubat 2011.

[38] Gislason P. O., Benediktsson J. A., Sveinsson J. R., Random Forests for Land Cover Classification, *Pattern Recognition Letters*, 2006, **27**(4), 294-300.

[39] Pal M., Random Forest Classifier for Remote Sensing Classification, *International Journal of Remote Sensing*, 2005, **26**(1), 217-222.

[40] Breiman, L., Manual on Setting up, Using, and Understanding Random Forests v3. 1, Statistics Department University of California, Berkeley (2002), https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf (Ziyaret tarihi: 27 Kasım 2018).

[41] Archer K. J., Kimes R. V., Empirical Characterization of Random Forest Variable Importance Measures, *Computational Statistics & Data Analysis*, 2008, **52**(4), 2249-2260.

[42] Breiman L., Random Forests, *Machine Learning*, 2001, **45**(1), 5-32.

[43] Mishra 2.3A. K., Ramteke S. V., Sen P., Verma A. K., Random Forest Tree Based Approach for Blast Design in Surface Mine, *Geotechnical and Geological Engineering*, 2018, **36**(3), 1647-1664.

KİŞİSEL YAYIN VE ESERLER

Kurt E. M., Becerikli Y., Network Intrusion Detection on Apache Spark with Machine Learning Algorithms, *19th International Conference on Engineering Applications of Neural Networks (EANN 2018)*, Bristol, United Kingdom, 3-5 September 2018.

ÖZGEÇMİŞ

Elif Merve Kurt 1991'de Ankara'da doğdu. Lise öğrenimini Tokat Anadolu Öğretmen Lisesi'nde tamamladı. 2011 yılında girdiği Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2015 yılında onur öğrencisi olarak mezun oldu. Aynı yıl içinde Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine başladı. Yüksek lisans eğitimi süresince makine öğrenmesi algoritmaları ve büyük veri üzerinde çeşitli çalışmalar yaptı. Yine aynı yıl içinde çalışmaya başladığı CTech Bilişim Teknolojileri şirketindeki yazılım mühendisliği görevinden 2018 yılı sonunda ayrıldı. 2019 yılı başında Architecht Bilişim Sistemleri şirketinde başladığı yazılım mühendisliği görevini halen sürdürmektedir.