KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

DOKTORA TEZİ

MAKİNE ÖĞRENMESİ VE DERİN ÖĞRENME YÖNTEMLERİ KULLANILARAK SALDIRI TESPİT VE ÖNLEME SİSTEMİ GELİŞTİRİLMESİ

MEHMET ALİ ALTUNCU

KOCAELİ 2021

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

DOKTORA TEZİ

MAKİNE ÖĞRENMESİ VE DERİN ÖĞRENME YÖNTEMLERİ KULLANILARAK SALDIRI TESPİT VE ÖNLEME SİSTEMİ GELİŞTİRİLMESİ

MEHMET ALİ ALTUNCU

Doç. Dr. Suhap ŞAHİN		
Danışman,	**Kocaeli Üniversitesi**	**.....**
Prof. Dr. Kerem KÜÇÜK		
Jüri Üyesi,	**Kocaeli Üniversitesi**	**.....**
Dr. Öğr. Üyesi Adem TUNCER		
Jüri Üyesi,	**Yalova Üniversitesi**	**.....**
Prof. Dr. Cihan KARAKUZU		
Jüri Üyesi,	**Bilecik Şeyh Edebali Üniversitesi**	**.....**
Dr. Öğr. Üyesi Alpaslan Burak İNNER		
Jüri Üyesi,	**Kocaeli Üniversitesi**	**.....**

Tezin Savunulduğu Tarih: 10.08.2021

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması, ağlara veya sistemlere karşı yapılan kötü niyetli saldırıların tespiti ve önlenmesi için iki yeni yöntem geliştirmek amacıyla gerçekleştirilmiştir.

Doktora eğitimim süresince desteğini esirgemeyen, tezimin her aşamasında sorunlarımı dinleyerek, görüşleri ile çalışmalarıma katkıda bulunan ve yoğun akademik yaşamında değerli zamanını her türlü problemimi çözmeye ayıran tez danışmanım saygı değer hocam Doç. Dr. Suhap ŞAHİN'e içtenlikle teşekkür ederim.

Tez çalışmama bilgi ve tavsiyeleri ile katkıda bulunan tez ilerleme jürim Prof. Dr. Kerem KÜÇÜK'e ve Dr. Öğr. Üyesi Adem TUNCER'e,

Çalışmalarım boyunca desteğini ve yardımlarını esirgemeyen, aynı laboratuvarı paylaştığım çalışma arkadaşlarım Dr. Öğr. Üyesi Fidan KAYA GÜLAĞIZ'a, Arş. Gör. Hikmetcan ÖZCAN'a ve Gömülü ve Algılayıcı Sistemler Araştırma Laboratuvarı çalışanları Tuğrul Hakan GENÇTÜRK, Enes KESEN, Ata NİYAZOV, Ömer Faruk BAYIR, Alperen Polat GEZGİN'e,

Akademik çalışmalarım sırasında, birçok aşamada beni destekleyen Mehmet Ali ÇAVUŞLU'ya ve Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'ndeki çalışma arkadaşlarıma,

Maddi ve manevi desteklerini tüm hayatı boyunca esirgemeyen anneme, babama, kardeşlerime, akrabalarıma ve tez çalışmam boyunca gösterdiği destek ve anlayış için eşime teşekkürü borç bilirim.

Ağustos – 2021 Mehmet Ali ALTUNCU

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

γ, β	:	Mini-yığın yöntemindeki öğrenme parametreleri
σ	:	Varyans
A	:	XGBoost algoritmasında amaç fonksiyon
b	:	Bias vektörü
Bİ	:	Beklenen iyileştirme
c	:	Ceza parametresi
d	:	Uzaklık
E	:	Derin öğrenmede uygunluk fonksiyonu
ε	:	Hata Değeri
f	:	Aktivasyon fonksiyonu
H	:	Entropi değeri
$h_j(x)$	:	RF algoritmasında j. ağaç kullanılarak x'teki çıkış değişkeninin tahmini
i	:	İterasyon sayısı
k	:	Ağaç sayısı
l	:	Gerçek etiket ile tahmin edilen değer arasındaki farkı ölçen hata fonksiyonu
L	:	Kayıp değer
μ	:	Ortalama
n	:	Öznitelik sayısı
p	:	Olasılık değeri
Q	:	Değişkenlerdeki gürültüyü azaltmak için kullanılan düzenleme terimi
r_{it}	:	GB algoritmasında negatif gradyanın bileşenleri
S	:	Bölünmüş veri kümesi
t	:	Yineleme sayısı
w	:	Ağırlık vektörü
x	:	Veri noktaları
y	:	Çıktı parametresi
z	:	Normalize edilmiş veri

Kısaltmalar

ANN	:	Artificial Neural Networks (Yapay Sinir Ağları)
Bİ	:	Beklenen İyileştirme
BLSTM	:	Bidirectional Long Short-Term Memory (İki Yönlü Uzun-Kısa Vadeli Bellek)
CNN	:	Convolutional Neural Network (Evrişimli Sinir Ağı)
DBN	:	Deep Belief Network (Derin İnanç Ağı)
DDoS	:	Distributed Denial of Service (Dağıtılmış Hizmet Reddi)
DFNN	:	Deep Feedforward Neural Network (Derin İleri Beslemeli Sinir Ağı)
DNS	:	Domain Name System (Alan Adı Sistemi)
DoS	:	Denial of Service (Hizmet Reddi)
DT	:	Decision Tree (Karar Ağacı)
ELM	:	Extreme Learning Machine (Aşırı Öğrenme Makinesi)

FAR	:	False Alarm Rate (Yanlış Alarm Oranı)
GBT	:	Gradient Boosted Trees (Gradyan Güçlendirilmiş Ağaçlar)
GLM	:	Generalized Linear Model (Genelleştirilmiş Doğrusal Model)
HLSTM	:	Hierarchical Long Short-Term Memory (Hiyerarşik Uzun-Kısa Vadeli Bellek)
HTTPS	:	Secure Hyper Text Transfer Protocol (Güvenli Hiper Metin Transfer Protokolü)
KNN	:	K-Nearest-Neighbors (K-En Yakın Komşu)
LLM	:	Logic Learning Machine (Mantık Öğrenme Makinesi)
LR	:	Linear Regression (Doğrusal Regresyon)
LSTM	:	Long Short-Term Memory (Uzun Kısa Süreli Bellek)
MDPCA	:	Modified Density Peak Clustering Algorithm (Modifiye Yoğunluk Tepe Kümeleme Algoritması)
MLP	:	Multi Layer Perceptron (Çok Katmanlı Algılayıcı)
NB	:	Naive Bayes (Saf Bayes)
NSL-KDD	:	NSL-Knowledge Discovery and Data mining (NSL-Bilgi Keşfi ve Veri madenciliği)
OSI	:	Open Systems Interconnection (Açık Sistem Arabağlantısı)
OSS	:	One-Side Selection (Tek Taraflı Seçim)
RF	:	Random Forest (Rastgele Orman)
RNN	:	Recurrent Neural Network (Tekrarlayan Sinir Ağı)
R2L	:	Remote to Local (Uzaktan Yerele)
SBMO	:	Sequential Bayesian Model-based Optimization (Sıralı Bayes Modeline Dayalı Optimizasyon)
SMOTE	:	Synthetic Minority Over-Sampling Technique (Sentetik Azınlık Aşırı Örnekleme Tekniği)
SSFCM	:	Semi-supervised Fuzzy C Means (Yarı denetimli Bulanık C Means)
STL	:	Self-Taught Learning (Kendi Kendine Öğretilen Öğrenme)
STS	:	Saldırı Tespit Sistemleri
SVM	:	Support Vector Machine (Destek Vektör Makinesi)
TLD	:	Top Level Domain (Üst Düzey Alan)
TPE	:	Tree-Structured Parzen Estimator (Ağaç Yapılı Parzen Tahmincisi)
U2R	:	User to Root (Kullanıcıdan Köke)

MAKİNE ÖĞRENMESİ VE DERİN ÖĞRENME YÖNTEMLERİ KULLANILARAK SALDIRI TESPİT VE ÖNLEME SİSTEMİ GELİŞTİRİLMESİ

ÖZET

İnternet kullanımının yaygınlaşması ile birlikte, siber saldırıların sayısı ve karmaşıklığı da artmaktadır. Bu durum saldırı tespit sistemi oluşturmada önemli bir rol oynayan geleneksel makine öğrenmesi tekniklerinin yetersiz kalmasına sebep olmaktadır. Derin öğrenme temelli yaklaşımlar bu sorunu belirli oranda çözmüştür ancak bu yöntemler, ağ trafiği verilerindeki dengesizlik sebebiyle düşük miktarda veri içeren atakların tespitinde başarısız olmaktadır. Bu tez çalışmasında farklı atak türlerin tespiti için iki farklı yöntem önerilmiştir.

Tez çalışmasının ilk aşamasında Probe, DoS, U2R ve R2L ataklarının tespiti için hem makine öğrenmesi hem de derin öğrenme temelli yöntemlerin avantajlarını kullanan hibrit bir saldırı tespit sistemi geliştirilmiştir. Geliştirilen sistemde ilk aşamada veriler atak ve normal olarak ikili sınıflandırma işlemi gerçekleştirilmiştir. İkinci aşamada da atak olarak tespit edilen verilerin türlerinin tespiti için özgün bir DFNN modeli önerilmiştir. Önerilen sistemin başarımını değerlendirmek için NSL-KDD veri seti kullanılmıştır. Önerilen hibrit model, geleneksel yöntemlere ve son zamanlarda önerilen tekniklere kıyasla daha iyi bir sınıflandırma başarımı göstermiştir.

Tez çalışmasının ikinci aşamasında DNS üzerinden gerçekleştirilen ve yaygın olarak kullanılan DNS tünelleme tehditlerini önlemek için canlı ağlarda gerçek zamanlı çalışan derin öğrenme tabanlı bir sistem geliştirilmiştir. DNS tünelleme tespiti, ağ içerisinde gerçek zamanlı olarak çıkarılan özellikler kullanılarak derin ağ tabanlı karar mekanizmaları ile gerçekleştirilmiştir. Testler esnasında, sistem gerçek zamanlı olarak gelen tehditleri önleyecek şekilde ağa entegre edilmiştir. Test sonuçları, DNS protokolü üzerinden yapılan tünel saldırılarının neredeyse tamamının, sistemin gerçek zamanlı olarak çalışmasında herhangi bir gecikemeye sebep olmadan engellendiğini göstermiştir.

Anahtar Kelimeler: Derin Öğrenme, DNS Tünelleme, Hiperparametre Optimizasyonu, Makine Öğrenmesi, Saldırı Tespit Sistemleri.

DEVELOPING AN INTRUSION DETECTION AND PREVENTION SYSTEM USING MACHINE LEARNING AND DEEP LEARNING METHODS

ABSTRACT

The number and complexity of cyber-attacks grow in parallel with the increasing use of the Internet. As a result, traditional machine learning techniques which play an important role in intrusion detection systems become inadequate. Deep Learning-based approaches resolve this problem to a certain extent; however, these methods fail to detect the attacks with a small amount of data due to the unbalanced network traffic data. In this thesis, two different methods have been proposed for the detection of different attack types.

In the first phase of the thesis, a hybrid intrusion detection system which uses the advantages of both machine learning and deep learning-based methods is developed to detect Probe, DoS, U2R and R2L attacks. In this system, data are classified in binary as attack and normal, in the first phase. In the second phase, in order to classify the types of data that are detected as attack, a unique DFNN model is proposed. NSL-KDD dataset is used to evaluate the performance of the proposed system. The proposed hybrid model demonstrates a better classification performance compared to traditional methods and other recently proposed techniques.

In the second phase of the thesis, a real-time deep learning-based system is developed on live networks to prevent common DNS tunneling attacks which use over DNS. DNS tunneling detection is been carried out by deep network-based decision mechanisms, using real-time extracted features within the network. During tests, the system is integrated into the network to prevent incoming attacks in real-time. Test results are shown that almost all tunnel attacks over the DNS protocol are blocked without causing any delay in the operation of the system in real-time.

Keywords: Deep Learning, DNS Tunneling, Hyperparameter Optimization, Machine Learning, Intrusion Detection Systems.

GİRİŞ

Teknolojinin hızla gelişmesiyle birlikte internet insan hayatının vazgeçilmez bir parçası haline gelmiştir. 2019 yılı istatistiklerine göre dünya çapında 4,5 milyardan fazla insan internet kullanmaktadır. Bu rakam dünya nüfusunun yüzde 60'tan fazlasına karşılık gelmektedir (Kemp, 2019). İnternet kullanımının bu oranda yaygınlaşmasıyla, internet üzerinden aktarılan verinin miktarı da büyük oranda artış göstermiştir. Bu gelişmelerle birlikte bilgi güvenliği ve gizliliği gibi problemlerde de ciddi oranlarda artış olmuştur. Böylece daha güvenli bir veri iletim ortamı sağlamak için kullanılan saldırı tespit sistemlerinin önemi artmıştır.

Saldırı tespit sistemleri (STS) ağ kaynaklarındaki etkinlikleri izlemek için çeşitli teknikler kullanan güvenlik çözümlerinden biridir (Tama ve Rhee, 2017). Bu sistemler yaygın olarak iki farklı şekilde kategorize edilmektedir. Bunlardan ilki sistemin konumuna göre ana bilgisayar tabanlı, ağ tabanlı ve hibrit saldırı tespit sistemleridir. Diğeri ise saldırı tespit yaklaşımına göre imza ve anomali tabanlı tespit sistemleridir (Meng ve diğ., 2018, Kasongo ve Sun, 2019). STS'lerin en temel görevi ilk aşamada ağ verisinin normal ve saldırı verisi olarak ayrılmasını sağlamaktır. İkinci aşamada ise anormal olarak tespit edilen verinin DoS (Denial of Service), R2L (Remote to Local), U2R (User to Root), Probe vb. saldırıları şeklinde kategorize edilmesi gerekmektedir. Bir STS, sınıflandırma doğruluğu ne kadar yüksek ve FAR (False Alarm Rate) değeri ne kadar düşükse o kadar etkili ve başarılı kabul edilir (Kasongo ve Sun, 2019). Aynı zamanda STS'lerin internet ortamındaki hızlı değişimlerle birlikte ortaya çıkan yeni saldırı türlerine karşı da duyarlı olması gerekmektedir (Liu ve Lang, 2019). Bu gereksinimleri karşılayacak STS sistemlerinin geliştirilmesi için makine öğrenmesi teknikleri ve derin öğrenme temelli yöntemler yaygın olarak kullanılmaktadır.

Makine öğrenmesi, büyük veri kümeleri üzerinden performansı arttırmak veya tahmin yapmak için hesaplama tekniklerinin kullanılması olarak tanımlanabilir (Mohri ve diğ., 2018). STS'ler de geçmiş verileri göz önünde bulundurarak saldırı tespiti yapmayı hedeflemektedir ve bu en temelde bir makine öğrenmesi alt dalı olan sınıflandırma

problemidir. Saldırı tespiti alanında yaygın olarak kullanılan makine öğrenmesi yöntemleri K-Nearest-Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), Naive Bayes (NB), Artificial Neural Network (ANN) vb. olarak sıralanabilir. Yeterli eğitim verisi elde edildiğinde makine öğrenmesi temelli yöntemler saldırı tespiti probleminde yüksek doğruluklara ulaşabilirler. Makine öğrenmesi yöntemlerinin tasarımı ve kodlanması kolaydır (Liu ve Lang, 2019) ancak bu yöntemlerin ilk adımı olan özellik çıkarma işlemi tam olarak gerçekleştirilemezse yöntemin başarısı düşecektir (Jiang ve diğ., 2020). Son yıllarda kullanımı yaygınlaşan ve bir makine öğrenmesi alt dalı olan derin öğrenme yöntemleri de saldırı tespiti alanında kullanılmaktadır. Bu yöntemlerde özellik çıkarma ihtiyacı otomatize edilmiştir. Aynı zamanda bu yöntemler büyük veriler üzerinde makine öğrenmesi yöntemlerine göre daha başarılı sonuçlar üretmektedirler. Ancak verilerdeki dengesiz dağılım derin öğrenme temelli yöntemlerin küçük boyutlu saldırı verisi içeren saldırıların tespitinde FAR değerinin artışına sebep olmaktadır (Jiang ve diğ., 2020, Wu ve diğ., 2018). Bu nedenlerle STS'lerde kullanılan veriler gibi dengesiz dağılıma sahip veri setleri üzerinde hem makine öğrenmesi yöntemlerinin hem de derin öğrenme yöntemlerini avantajlarını bir araya getirecek hibrit yöntemlere ihtiyaç vardır.

STS ağ güvenliği açısından çok stratejik bir konudur. Uzun yıllardır bu konuda pek çok farklı çalışma gerçekleştirilmiştir ve konu günümüzde de güncelliğini korumaktadır. Bu konuda gerçekleştirilen çalışmalar genel olarak iki farklı alana yoğunlaşmışlardır. Bu alanlardan bir tanesi klasik makine öğrenmesi, diğeri ise derin öğrenmedir. NSL-KDD veri seti, bu alanda en yaygın olarak kullanılan veri setlerinden bir tanesidir. Literatürde bu veri seti üzerinden gerçekleştirilmiş pek çok STS sistemi bulunmaktadır. Ancak NSL-KDD veri seti içerdiği sınıflar açısından hem eğitim kümesinde hem de test kümelerinde dengesiz bir dağılım içermektedir. Bu nedenle gerçekleştirilen çalışmaların pek çoğunda veri seti üzerinde indirgeme, arttırma, yumuşatma vb. işlemler uygulanmıştır. Bu şekilde gerçekleştirilen çalışmaların başarısı veri setine müdahale olmadan gerçekleştirilen çalışmalara göre yüksek olmaktadır. Ancak bu işlem veri setinin doğasını bozmaktadır ve geliştirilen STS'lerin gerçek ağ verilerine uygunluğunu düşürmektedir. Bu nedenle tezin ilk aşamasında yapılan çalışmada NSL-KDD veri seti hiçbir değişikliğe uğratılmadan kullanılmıştır.

Aşağıda anlatılan ilgili çalışmalar bölümünde de yoğunlukla veri setini benzer şekilde kullanan çalışmalara yer verilmiştir.

Kasongo ve Sun (2019), saldırı tespiti için derin öğrenme tabanlı bir sistem önermişlerdir. Önerilen sistemde özelliklerin çıkarımı için entropiye dayalı bir algoritma kullanılmıştır. Özellik çıkarımı işleminden sonra geleneksel makine öğrenme yöntemleri ile önerilen ileri beslemeli derin sinir ağı yöntemi karşılaştırılmıştır. Çalışmada eğitim aşamasında KDDTrain veri setinin %75'i seçilmiştir. Test aşamasında KDDTest+ verisinde %87,74 alındığı görülmüştür.

Wu ve diğ. (2018), derin öğrenme yöntemlerinden biri olan Convolutional Neural Network (CNN) yöntemini önermişlerdir. Çalışmada CNN'in performansını arttırmak için veri ön işleme aşamasında normalleştirilmiş veriler görüntü veri formatına dönüştürülmüştür. Eğitim aşamasında ise önerilen modelin hangi parametrelerde daha iyi sonuçlar verdiğinin belirlenmesi işlemi otomatik olarak yapılmıştır. Sonuçlar incelendiğinde KDDTest+ verisinde %81,29, KDDTest-21 verisinde ise %64,67 başarı oranı elde edildiği görülmüştür.

Hu ve diğ. (2020), saldırı tespitini önlemek için yine derin öğrenme tabanlı hibrit bir yöntem önermişlerdir. Çalışmada NSL-KDD veri setindeki saldırı türlerinin dağılımını dengelemek için Adasyn algoritmasını kullanmışlardır. Veri setindeki ön işleme aşamasının ardından geliştirilmiş CNN modeli tasarlanmıştır. Önerilen model, kanallar arası bilgi fazlalığının model eğitimi üzerindeki etkisini ortadan kaldırabilen bölünmüş evrişim modülüne dayanmaktadır. KDDTest+ verisinde %84,08, KDDTest-21 verisinde ise %72,54 başarı oranı elde edildiği belirtilmiştir.

Latah ve Toker (2020), KNN, aşırı öğrenme makinesi ve hiyerarşik aşırı öğrenme makinesi sınıflandırma yöntemlerine dayalı çok katmanlı bir yaklaşım önermişlerdir. Önerilen yöntemde her katmanda eğitim veri seti belirli bir saldırı türü (DoS, Probe vb.) ve normal bir trafiği ya da sonraki katman tarafından tespit edilmesi gereken bir saldırıyı temsil eden "diğer" kategorisi olmak üzere iki kategoriye ayrılmıştır. Modelde her bir katman başına bir sınıflandırıcı kullanılmıştır. Sonuçlar incelendiğinde KDDTest+ verisinde %84,29 başarı oranı elde edildiği görülmüştür.

Wang ve diğ. (2020), saldırı tespit sistemi için makine öğrenmesi temelli bir çerçeve önermişlerdir. Önerilen yöntem veri setindeki önemli özellikleri çıkararak, özelliklerin değerleri ile belirli saldırı türleri arasındaki ilişkileri keşfetme temeline dayanmaktadır. Önerilen yöntem ile KDDTest+ verisinde %80,60 başarı oranı elde edildiği belirtilmiştir.

Su ve diğ. (2020), iki aşamalı derin öğrenme tabanlı bir sistem önermişlerdir. Birinci aşamada trafik verilerinin yerel özelliklerini yakalamak için evrişimli katman kullanılmıştır. İkinci aşamada ise, veri paketindeki zaman serisi özelliğini öğrenmek için Bidirectional Long Short-Term Memory (BLSTM) modeli önerilmiştir. Çalışma geleneksel derin öğrenme yöntemleri ile karşılaştırılmış olup, önerilen sistemin diğer yöntemlere göre KDDTest+ verisinde %84,25, KDDTest-21 verisinde %69,42 başarı oranı ile daha iyi sonuçlar elde edildiği belirtilmiştir.

Al-Qatf ve diğ. (2018), geleneksel makine öğrenmesi ve derin öğrenme yöntemlerini birleştiren hibrit bir yaklaşım önermişlerdir. Önerilen yaklaşımda özellik öğrenme ve boyutluluk azaltma işlemleri için Self-Taught Learning (STL) çerçevesi, sınıflandırma aşaması için SVM algoritması kullanılmıştır. Sonuçlar incelendiğinde KDDTest+ verisinde %84,96 başarı oranı elde edildiği görülmüştür.

Hou ve diğ. (2020), Hierarchical Long Short-Term Memory (HLSTM) ağına dayanan derin öğrenme mimarisi önermişlerdir. Çalışmada eğitim veri setindeki örnekleri dengelemek için normal ve DoS kategorilerinin kayıp ağırlığı düşürülürken, R2L ve U2R kategorilerinin kayıp ağırlığı arttırılmıştır ve veriler bu haliyle görüntü veri formatına dönüştürülmüştür. Böylece önerilen mimarinin özellikle düşük frekanslı saldırılarla ilgili olanları hızlı ve doğru bir şekilde çalışabildiği belirtilmiştir. Sonuç olarak KDDTest+ verisinde %83,85, KDDTest-21 verisinde ise %69,73 başarı oranı elde edilmiştir.

Haggag ve diğ. (2020), derin öğrenme algoritmalarına dayalı bir STS sistemi önermişlerdir. Bu çalışmada da eğitim veri setindeki örnekleri dengelemek için Synthetic Minority Over-Sampling Technique (SMOTE) kullanılmıştır. Önerilen sistemde model eğitimi Apache Spark üzerinde yapılmıştır. Model eğitiminde Multi Layer Perceptron (MLP), Recurrent Neural Network (RNN) ve Long Short-Term Memory (LSTM) yöntemleri ayrı ayrı değerlendirilmiştir. LSTM yöntemiyle

KDDTest+ verisinde %85,44, KDDTest-21 verisinde ise %70,59 başarı oranı elde edildiği görülmüştür.

Wu ve diğ. (2020), Deep Belief Network (DBN) ve ağırlıklı SVM yöntemlerinin beraber kullanıldığı hibrit bir sistem önermişlerdir. Önerilen yöntemde özellik çıkarma işlemi DBN yöntemiyle, özelliklerin iyileştirilmesi ve sınıflandırma işlemi ağırlıklı SVM yöntemiyle gerçekleştirilmiştir. Çalışma sonucunda KDDTest+ verisinde %85,73, KDDTest-21 verisinde ise %70,25 başarı oranı elde edilmiştir.

Liu ve diğ. (2020), CNN tabanlı bir STS önermişlerdir. Çalışmada modelin daha iyi sınıflandırma yapabilmesi için eğitim veri setinde az bulunan saldırı türlerine veri arttırma yapılmıştır. Ayrıca CNN'in etkinliğini arttırmak için özellik vektörleri 2 boyutlu (2D) piksel tabanlı görüntülere dönüştürülmüştür. Sonuç olarak KDDTest+ verisinde %81,30, KDDTest-21 verisinde ise %64,70 başarı oranı elde edilmiştir.

Jiang ve diğ. (2020), CNN ile BLSTM'yi birleştiren derin hiyerarşik ağ modelini önermişlerdir. Bu çalışmada da veri setindeki saldırı türleri arasındaki örnek sayısı dengelenmiştir. Veri setindeki çoğunluk örneklerini azaltmak için One-Side Selection (OSS) ve azınlık örneklerini artırmak için SMOTE tekniği kullanılmıştır. Önerilen yöntem ile KDDTest+ verisinde %83,58 başarı oranı elde edilmiştir.

Tama ve diğ. (2019), makine öğrenmesi temelli bir STS önermişlerdir. Çalışmada ilk olarak eğitim veri setinin özellik boyutunu azaltmak için, parçacık sürüsü optimizasyonu, karınca kolonisi algoritması ve genetik algoritma gibi üç yöntemden oluşan bir hibrit özellik seçim tekniği kullanılmıştır. İkinci aşamada ise RF ve torbalama olan iki seviyeli bir sınıflandırıcı önerilmiştir. Önerilen modelle KDDTest+ verisinde %85,79, KDDTest-21 verisinde ise %72,52 başarı oranı elde edilmiştir.

Ieracitano ve diğ. (2020), istatistiksel analiz ve otomatik kodlayıcı tabanlı bir derin öğrenme mimarisi önermişlerdir. Önerilen model eğitilmeden önce veri setindeki aykırı değerler kaldırılmıştır. İkili sınıflandırmada normal ve anormal kategorilerin, çoklu sınıflandırmada ise DoS, R2L ve probe saldırı türlerinin tespiti yapılmıştır. KDDTest+ verisinde ikili sınıflandırmada için %84,21, çoklu sınıflandırma için %87 başarı oranı elde edilmiştir.

Tchakoucht ve Ezziyyani (2018), çok katmanlı yankı durum makinesi tabanlı RNN mimarisi önermişlerdir. Önerilen mimari NB, J48, RF, KNN, MLP ve SVM gibi geleneksel makine öğrenmesi yöntemleri ile karşılaştırılmış olup, KDDTest+ verisinde %83, KDDTest-21 verisinde ise %66,20 başarı oranı elde edilmiştir.

Gao ve diğ. (2018), regresyon ağacı ve belirsizlik yöntemlerinin birlikte kullanıldığı hibrit bir yöntem önermişlerdir. Çalışmada etiketli verilerin eğitimi için regresyon ağacı, etiketlenmemiş veriler için belirsizlik tabanlı bir yöntem kullanılmıştır. Önerilen yöntem KDDTest+ verisinde %84,54, KDDTest-21 verisinde ise %71,29 başarı oranı elde edilmiştir.

Yang ve diğ. (2019), Modified Density Peak Clustering Algorithm (MDPCA) ve DBN yöntemlerinin kullanıldığı hibrit bir model önermişlerdir. MDPCA yöntemi eğitim veri setinde az sayıda olan U2R ve R2L türündeki saldırıları diğerleriyle dengelemek için, DBN yöntemi ise sınıflandırma için kullanılmıştır. Önerilen model KDDTest+ verisinde %82,08, KDDTest-21 verisinde ise %66,18 başarı elde etmiştir.

Li ve diğ. (2020), çoklu-CNN tabanlı bir derin öğrenme yaklaşımı önermişlerdir. Önerilen model J48, NB, RF, MLP ve SVM yöntemleri ile karşılaştırılmış olup, KDDTest+ verisinde %86,95, KDDTest-21 verisinde ise %76,67 başarı oranı elde edilmiştir.

Rathore ve Park (2018), Semi-supervised Fuzzy C Means (SSFCM) ile Extreme Learning Machine (ELM) sınıflandırıcı yöntemini birleştiren bir model önermişlerdir. Önerilen modelde SSFCM ile kümeleme, ileri beslemeli sinir ağı tabanlı ELM yöntemi ile sınıflandırma işlemi yapılmıştır ve KDDTest+ verisinde %86,53, KDDTest-21 verisinde ise %75,77 başarı oranı elde edilmiştir.

Tez kapsamının ilk aşamasında hem makine öğrenmesi hem de derin öğrenme temelli yöntemlerin avantajları göz önünde bulundurularak hibrit bir STS geliştirilmiştir. Geliştirilen sistemde sınıflandırma işlemi iki aşamalı olarak gerçekleştirilmektedir. İlk aşamada XGBoost algoritması kullanılarak saldırı ve normal verilerin sınıflandırılması ikinci aşamada ise probleme özgü olarak geliştirilen Deep Feedforward Neural Network (DFNN) modeli ile saldırı türlerinin sınıflandırılması yapılmıştır. Önerilen

hibrit yöntem TXG-DFNN olarak isimlendirilmiştir. Tez kapsamının ilk aşamasında ana katkılar aşağıda maddeler halinde verilmiştir:

- Ağ veri trafiğindeki dengesiz dağılım modellerin başarımını ciddi oranda etkilemektedir. Çalışma kapsamında veri setine müdahale etmeden, kullanılan iki aşamalı model ile veri dengesizliğinin sınıflandırma üzerindeki etkisi indirgenmiştir. Bu amaçla modelin ilk aşamasında dengesiz veriler üzerinde başarısı bilinen XGBoost algoritması kullanılmıştır. Böylece hem model oluşturma hem de test aşamalarında ihtiyaç duyulan zaman klasik modellere göre indirgenmiştir. XGBoost algoritması özellikle hızlı ve paralel veri işleme amacıyla geliştirilmiş bir algoritmadır. Doğru parametre seçimiyle hem performans hem de doğruluk anlamında daha iyi sonuçlar üretmektedir. Bu amaçla XGBoost algoritmasına ait ideal hiperparametrelerin seçimi için Tree-Structured Parzen Estimator (TPE) yöntemi kullanılmıştır.
- İkinci aşamada XGBoost algoritmasının saldırı olarak sınıflandırdığı verilerin probleme özgü olarak geliştirilmiş olan DFNN modeline sokulmasıyla hem çoklu sınıflandırma işlemine ait hem de eğitim veri setinde bulunmayan saldırı türleri için FAR değerinin düşürülmesi sağlanmıştır. Bu aşamada oluşturulan DFNN modeli, yalnızca saldırı türlerinin tespiti için geliştirildiğinden daha spesifik ve başarılı olmuştur.
- Geliştirilen hibrit modelin etkinliği eğitim veri kümesinde bulunmayan saldırı tiplerini içeren hem KDDTest-21, hem de KDDTest+ veri setleri üzerinden gösterilmiştir. Buna ek olarak geliştirilen model literatürdeki benzer çalışmalar, bu alanda yaygın kullanılan yöntemlerle de (RF, KNN, NB, DFNN vb.) karşılaştırılmıştır. Sınıflandırma başarımı sırasıyla KDDTest+ ve KDDTest-21 veri setleri için ikili sınıflandırmada %96,20, %93,01 ve çoklu sınıflandırmada %89,68, %80,62 olarak elde edilmiştir. Böylece veri setine müdahale etmeden, veri dengesizliğinin sınıflandırma üzerindeki etkisi indirgenmiştir.

Literatürde dengesiz veriler kullanılarak geliştirilmiş topluluk modelleri ve derin öğrenme tekniklerini hibrit olarak kullanan iki aşamalı bir yöntem bulunmamaktadır. Ayrıca çalışma kapsamında geliştirilen DFNN modeli yalnızca atak olarak sınıflandırılan verilen türlerine ayrılmasını sağladığından ve ilk aşamada indirgenmiş olan veriler üzerinden model ürettiğinden literatürde yer alan DFNN modellerinden farklı ve özgündür.

Domain Name System (DNS) internette kullanılan önemli protokol ve hizmetlerden birisidir. Birden fazla kullanım amacı olmasına rağmen, DNS'in en önemli görevi alan adları ile Internet Protocol (IP) adreslerini çift taraflı dönüştürme işlemini gerçekleştirmektir. DNS alan adı ve kullanıcı sayısının artmasından etkilenmeyerek, hızlı bir şekilde sorgulara yanıt vermeyi sağlayan ve dünyanın her yerinden erişilebilen bir sunucu topluluğundan oluşmaktadır (Skow, 2016).

Tipik bir DNS sorgusu UDP/53 portunu kullanmasına rağmen cevabı 512 byte'dan büyük DNS sorguları TCP/53 portunu kullanmaktadır. DNS'te asıl amaç veri taşıma işlemi olmadığından DNS'in kötü amaçlı iletişim veya veri sızması için bir tehdit olarak algılanması çoğu durumda akıllara gelmemektedir. Bu yüzden güvenlik duvarlarında çoğu zaman bu portlar (53 portu) açık olmaktadır. Bu portların açık olması, DNS protokolü içerisindeki dosya aktarım sistemi gibi çeşitli saldırı türlerini gerçekleştirmek için güvenlik açığı oluşturmaktadır. Güvenlik açığından faydalanarak yapılan bu saldırılarda genel olarak tünelleme yöntemi kullanılmaktadır. Bir protokole ait verilerin başka bir protokol üzerinden taşınması işlemine tünelleme, taşıma işleminin DNS paketleri aracılığıyla yapılması ise DNS tünellemesi olarak adlandırılır (Sammour ve diğ., 2017).

DNS Tünelleme tespit edilirken genel olarak ağ paketlerine ait iki özellik kullanılmaktadır. Bunlar yük ve trafik analizleridir. Yük analizinde her bir DNS istek ve yanıtın, etki alanı uzunluğu, bayt sayısı ve içerik gibi özellikleri tek tek değerlendirilerek DNS tünelleme tespiti yapılmaktadır. Trafik analizinde ise genel trafiğe ait istatiksel özellikler (DNS trafiğinin hacmi, her bir alana düşen bilgisayar adlarının sayısı, coğrafi konum vb.) ve alan geçmişi kullanılarak DNS tünelleme tespiti yapılmaktadır (Farnham ve Atlasis, 2013).

Ağ paketlerine ait özellikler belirlendikten sonra DNS tünelleme tespiti için kullanılacak yöntemin belirlenmesi gerekmektedir. Literatürde yapılan çalışmalar incelendiğinde DNS Tünelleme tespitinde genel olarak sınıflandırma temelli makine öğrenmesi teknikleri kullanılmıştır.

Aiello ve diğ. (2013), denetimli öğrenme tekniklerinden Bayes sınıflandırma yöntemi kullanarak gerçek zamanlı DNS tünelleme tespitini gerçekleştirmişlerdir. Çalışmada DNS sorgularının ve cevaplarının istatistiksel özellikleri kullanılmıştır.

Aiello ve diğ. (2015), yaptıkları başka bir çalışmada DNS verisine ait paketler arası varış zamanları ve paket büyüklükleri gibi istatiksel özellikleri kullanarak bir önceki çalışmadan farklı olarak gerçek zamanlı DNS tünelleme tespitini yapmışlardır. Çalışmada DNS tünelleme tespiti için makine öğrenmesi teknikleri karşılaştırılmış olup, özellikle sınıflandırma temelli yöntemlerin daha başarılı sonuçlar verdiği belirtilmiştir.

Sammour ve diğ. (2017), DNS Tünelleme tespiti için sınıflandırma temelli makine öğrenmesi tekniklerinden faydalanmışlardır. Çalışmada her bir DNS paketine ait DNS istek uzunluğu, IP paket gönderen uzunluğu, IP paket yanıt uzunluğu, kodlanmış DNS sorgu adı uzunluğu, istek uygulama katmanı entropisi, IP paket entropisi ve sorgu adı özellikleri kullanılmıştır. Yöntem olarak SVM, NB ve J48 algoritmaları kullanılmış olup, bu algoritmaların karşılaştırmalı analiz sonuçları verilmiştir. Sonuçlar incelendiğinde, SVM'in %83 f-measure oranıyla diğer yöntemlere kıyasla daha başarılı olduğu görülmüştür.

Almusawi ve Amintoosi (2018), DNS tünelini tespitiyle birlikte, tünel türlerini ayırt etmek için çok etiketli SVM önermişlerdir. Çalışmada 4 farklı (FTP, HTTP, HTTPS, POP3) protokolden ve 530 örnekten oluşan tünel veri seti kullanılmıştır. Tünel türlerinin ayırt edilmesi için DNS uzunluğu, IP uzunluğu, entropi bilgilerini içeren 7 farklı özellikten faydalanılmıştır. Ayrıca önerilen çok etiketli SVM yöntemi çok etiketli Bayesian sınıflandırıcı ile karşılaştırılmış olup, önerilen yöntemin %80 f-measure oranıyla daha başarılı sonuçlar verdiği görülmüştür.

Liu ve diğ. (2017), yine bir sınıflandırma temelli DNS tüneli tespit mekanizması gerçekleştirmişlerdir. Gerçekleştirilen sistem, normal veri ile tünelli veri arasındaki karakter dağılım farklarıyla birlikte, zaman aralıkları, DNS kayıt türleri ve DNS sorgu uzunluklarını dikkate alarak tespit işlemini gerçekleştirmektedir. Çalışmada sınıflandırma yöntemlerinden SVM, DT ve Lojistik Regresyon kullanılmış olup, SVM'in daha iyi sonuçlar verdiği görülmüştür.

Buczak ve diğ. (2016), DNS Tünelleme tespitinde yine sınıflandırıcı temelli bir yöntem önermişlerdir. Çalışmada Iodine, DNSCat2 ve Cobalt Strike araçlarıyla toplanan tünel veri seti ve legal DNS verisinden oluşmaktadır. Daha sonra ağ verisine

ait 16 özellik belirlenip, RF yöntemine tabi tutulmuştur. Çalışmada her bir araçla toplanan tünel türlerinin %95 oranla tespit edildiği belirtilmiştir.

Cambiaso ve diğ. (2016), DNS tünelleme sisteminin içyapısını incelemek için tespitin aksine, yeni bir DNS tünel açma sistemi önermişlerdir. Önerdikleri yaklaşım ağ trafiğine ait ortalama, standart sapma, DNS sorgularının ve cevaplarının boyutlarının varış zamanları gibi istatiksel özelliklerin çıkarılmasına dayanmaktadır. Bunun için de temel bileşenler analizi ve karşılıklı bilgi yöntemlerini kullanmışlardır.

Homem ve diğ. (2017), DNS trafiği ile tünellenmiş verinin protokolünü tahmin etmeyi amaçlayan bir sistem tasarlamışlardır. Çalışmada DNS tünelleme tekniklerinin içyapısı incelenmiş ve paket baytlarının entropisi kullanılarak protokol türü tahmin edilmeye çalışılmıştır. Çalışmada veri seti Iodine aracılığıyla toplanmış olup, 20 örnekten oluşmaktadır. Çalışma sonucunda %75 başarı oranı elde edilmiştir.

Nadler ve diğ. (2019), DNS Tünelleme tespitiyle birlikte düşük verimli veri sızıntısını tespit etmek amacıyla makine öğrenmesi temelli bir yöntem önermişlerdir. Çalışmada ilk olarak toplanan DNS trafiğinden etki alanına karşılık gelenler özellik vektörüne dönüştürülmüştür. Ardından DNS Tünelleme tespiti yapan alanların tespit edilmesi için önceden eğitilmiş olan sınıf sınıflandırıcısı kullanılmaktadır. Çalışmada ayrıca tünel tespiti yapan alan adları kara listeye alınmaktadır. Çalışma, iki DNS tünelleme aracı (Iodine ve Dns2tcp) kullanılarak değerlendirilmiş ve doğruluğu test edilmiştir.

Aiello ve diğ. (2019), DNS tünelleme tespitiyle ilgili yaptıkları başka bir çalışmada K-Means ve Logic Learning Machine (LLM) yöntemlerini kullanmışlardır. Çalışmada anomali yoğunluğunu azaltmak için ilk olarak kümeleme yöntemi (K-Means) kullanılmıştır. Daha sonra kümeleme sonucu elde edilen veri, karar ağacından kural çıkarma ve sınıflandırma yöntemi olan LLM algoritması kullanılarak DNS Tünelleme tespiti gerçekleştirilmiştir. Çalışmada kullanılan test verisinin eğitim verisinden farklı davranış göstermesine rağmen başarılı sonuçlar elde edildiği belirtilmiştir.

Bubnov (2018), sınıflandırıcı temelli yapay sinir ağı kullanarak DNS Tünelleme tespitini gerçekleştirmiştir. Çalışmada paketlere ait tür, sayı, ad uzunluğu, ad entropisi, veri uzunluğu, veri entropisi özellikleri analiz edilmiş, üç katmanlı ileri beslemeli sinir

ağı modeliyle de DNS Tünelleme tespiti gerçekleştirilmiştir. Çalışma sonucunda %83 başarı oranı elde edilmiştir.

Engelstad ve diğ. (2017), mobil ağlardaki DNS tünelleme tespiti için tek sınıflı SVM ve K-Means algoritmalarını önermişlerdir. Önerilen makine öğrenmesi tabanlı yöntemler zaman, hedef, protokol ve DNS sorgusunun uzunluğu gibi özellikleri kullanarak mobil ağ içinde DNS tünellemeyi %96 oranında tespit edebilmektedir.

Ahmed ve diğ. (2019), DNS üzerinden verilerin dışarı sızmasını ve tünellenmesini tespit etmek için gerçek zamanlı bir sistem geliştirmişlerdir. Çalışmada iki kuruluştan toplanan DNS verilerinden öznitelik çıkarımı yapılmış, daha sonra makine öğrenmesi yöntemi kullanılarak, canlı bir ağ üzerinde DNS sorgularındaki anormallikler tespit edilmiştir.

Literatürde gerçek zamanlı olarak DNS tünelleme tespiti yapan çalışmalar olmasına rağmen, gerçek zamanlı olarak DNS tünellemeyi engelleyen bir sistemin tasarlanmadığı açıkça görülmektedir. Ahmed ve diğ. (2019) tarafından yapılan çalışmada bahsedilen, 2014 yılında Sally Beauty firması müşterilerine ait 25K ve Home Depot firması müşterilerine ait 56M kredi kartı bilgilerinin çalınması gibi durumların önüne geçebilmek için sadece DNS tünelleme tespiti değil, aynı zamanda gerçek zamanlı olarak DNS tünelleme saldırılarının engellenmesi büyük önem arz etmektedir.

Tez kapsamının ikinci aşamasındaki motivasyonumuz DNS trafiği üzerinden yapılan saldırılarıları DFNN tabanlı karar mekanizması kullanarak ağ üzerinde gerçek zamanlı olarak engelleyen sistem geliştirilmesidir. Geliştirilen bu sistem ile DNS üzerinden yapılacak tünel saldırılarını engellemek için ağ trafiğinin operatör tarafından sürekli olarak takip edilmesine gerek kalmayacaktır ve bu süreç otomatize hale gelecektir. Sistem karar mekanizmasında kullanılan özgün DFNN yöntemi ile literatürde önerilen çalışmalara göre tehditlerin engellenmesine yönelik daha yüksek başarı elde edilmiştir. Ayrıca önerilen sistemin canlı ağlar üzerinde tehdit anında gerçek zamanlı olarak karar verme özelliği özgün bir değere sahiptir.

Çalışmanın bundan sonraki bölümü şu şekilde organize edilmiştir. Birinci bölümde siber saldırı yöntemleri açıklanmıştır. İkinci bölümde STS'ler hakkında bilgi

verilmiştir. Üçüncü bölümde tez kapsamında kullanılan geleneksel makine öğrenmesi yöntemleri, dördüncü bölümde ise derin öğrenme yöntemleri hakkında bilgi verilmiştir. Beşinci bölüm, farklı saldırı türleri için önerilen iki farklı mimarinin detaylı olarak anlatıldığı kısımdır. Altıncı bölümde yapılan çalışmalardan elde edilen sonuçlar ve literatürdeki benzer çalışmalarla karşılaştırılması verilmiştir. Sonuçlar ve öneriler bölümünde ise çalışmanın özeti, bilime ve günümüz teknolojisine sağlayabileceği katkıları anlatılmıştır.

1. SİBER SALDIRI YÖNTEMLERİ

STS Anderson tarafından 1980'de ortaya atıldığında, bir siber saldırı girişimi veya tehdidi; bilgiye erişmek, bilgiyi değiştirmek veya bir sistemi güvenilmez veya kullanılamaz hale getirmek şeklinde tanımlamıştır (Sundaram, 1996). Siber saldırılar amacına göre genel olarak ikiye ayrılmaktadır. Birinci türde yapılan saldırıda, sistemin kaynakları değiştirilmeye çalışabilir. Buna aktif saldırı denilmektedir. İkinci türde ise sistem kaynaklarını etkilemeyecek şekilde sistemden bilgi öğrenilmesi ve bu bilginin kullanılması için saldırı yapılmaktadır (ör: telefon dinleme). Bu tür saldırılara ise pasif saldırı denilmektedir. Ayrıca siber saldırılar yapıldığı yere göre de ikiye ayrılmaktadır. Güvenlik çevresi içindeki bir varlık tarafından yapılan saldırılar iç saldırı olarak tanımlanmaktadır. Sistemin yetkisiz veya gayri meşru bir kullanıcısı tarafından yapılan saldırılara ise dış saldırı denilmektedir (URL-1, 2021). Siber saldırılar birçok farklı teknik kullanılarak yapılmaktadır. Uluslararası alanda en yaygın siber saldırı yöntemleri ise alt bölümlerde açıklanmıştır.

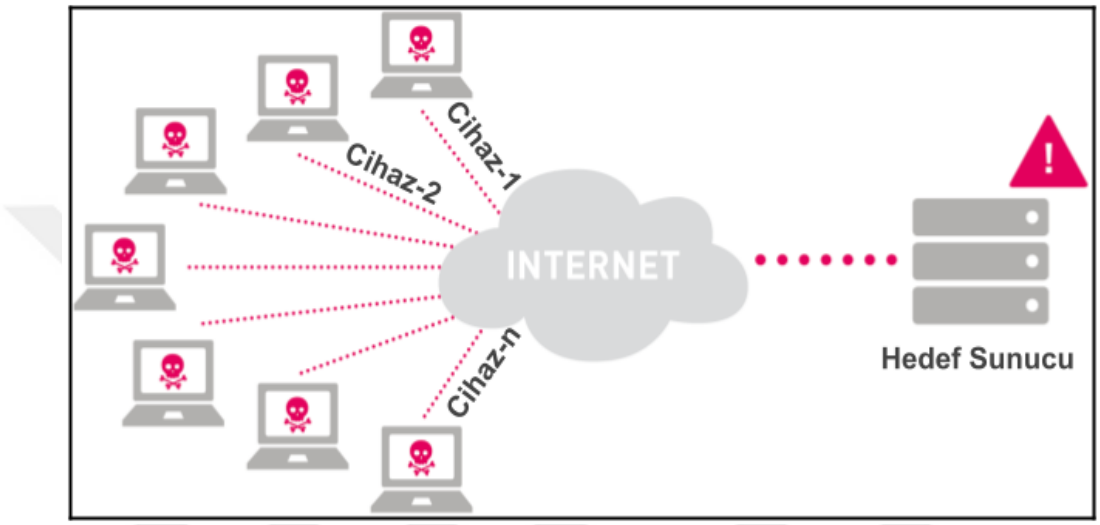
1.1. DoS-DDoS Saldırıları

DoS saldırıları, internet ve ağın gelişmesiyle birlikte ağ güvenliğindeki en popüler saldırı türüdür. DoS saldırısında sistem kaynaklarını aşarak hizmet talebine yanıt veremez. Bir saldırgan tarafından kontrol edilen kötü amaçlı yazılımlardan etkilenen ana makine, DoS saldırısı başlatır. Bu tür siber saldırıda, internete bağlı hostun hizmetini bozarak makina veya ağ kaynakları amaçlanan kullanıcı için kullanılamaz hale getirilir (Biju ve diğ., 2019). Tek bir internet bağlantısıyla tek bir cihazdan saldırı yapılmasına DoS, internetin farklı bölümlerinden aynı anda birden fazla bağlı cihazdan saldırı yapılmasına DDoS (Distributed Denial of Service) saldırısı denilmektedir. DDoS saldırısının çalışma prensibi Şekil 1.1'de gösterilmiştir.

DoS-DDoS saldırısı yapılırken birbiriyle ilişkili bazı adımlar içerir. Bu adımlar aşağıda maddeler halinde listelenmiştir:

- Öncelikle, saldırgan yanlış adresle çok sayıda hizmet isteği gönderir.

- Sunucu, gönderene bir yanıt mesajı gönderir ve istemciden yanıt bilgilerini bekler.
- Adresler sahte olduğu için sunucu herhangi bir bilgi alamaz ve uzun süre beklemesi gerekir ve zaman aşımı nedeniyle bağlantı kesilir.
- Bu istek için ayrılan kaynak serbest bırakılamaz. İstek sayısı fazla olduğu için sunucu kaynakları tükenene kadar kullanılır. Böylece yeni kullanıcı hizmeti alamaz ve saldırı başarıyla gerçekleştirilmiş olur (Liu, 2009).



Şekil 1.1. DDoS saldırısı çalışma prensibi

DDoS saldırıları hacim tabanlı, protokol ve uygulama katmanı saldırılar olmak üzere genel olarak 3 kategoriye ayrılmaktadır. Hacim tabanlı saldırılarda amaç, saldırıya uğrayan sitedeki bant genişliğini doyurmaktır. Protokol saldırıları, sunucunun geçerli isteklere yanıt veremeyeceği kadar kaynağı tüketmek amacıyla yapılan saldırılardır. Yedinci katman saldırıları olarak da isimlendirilen uygulama katmanı saldırıları, çok sayıda istek göndererek sunucuda yoğun kaynak kullanımı ve işlem gerektirerek aşırı yüklemeye sebep olmaktadır (Singh ve diğ., 2017). DDoS saldırılarının en yaygın türleri Tablo 1.1'de gösterilmiştir.

Tablo 1.1. Yaygın kullanılan DDoS saldırı türleri

Saldırı Türü	Kategori	Yöntem
UDP flooding	Hacim Tabanlı	UDP protokolü kullanılarak yapılır.
ICMP flooding	Hacim Tabanlı	Sürekli ICMP paket istekleri göndermesiyle yapılır.

Tablo 1.1. (Devam) Yaygın kullanılan DDoS saldırı türleri

Saldırı Türü	Kategori	Yöntem
Packet-spoofing	Hacim Tabanlı	IP paketlerinin sahte kaynak IP adresi ile oluşturulmasıyla yapılır.
SYN flooding	Protokol	Sisteme ardışık senkronizasyon isteği gönderilerek yapılır.
Smurf	Protokol	IP ve ICMP protokollerinin açıkları kullanılarak yapılır.
Fraggle	Protokol	Yönlendiricinin adresine sürekli sahte UDP trafiği gönderilerek yapılır.
HTTP flooding	Uygulama Katmanı	HTTP get ve post istekleri kullanılarak yapılır.
DNS query flooding	Uygulama Katmanı	Sunuculardaki DNS çözümleme işleminin bozulmaya çalışılmasıyla yapılır.

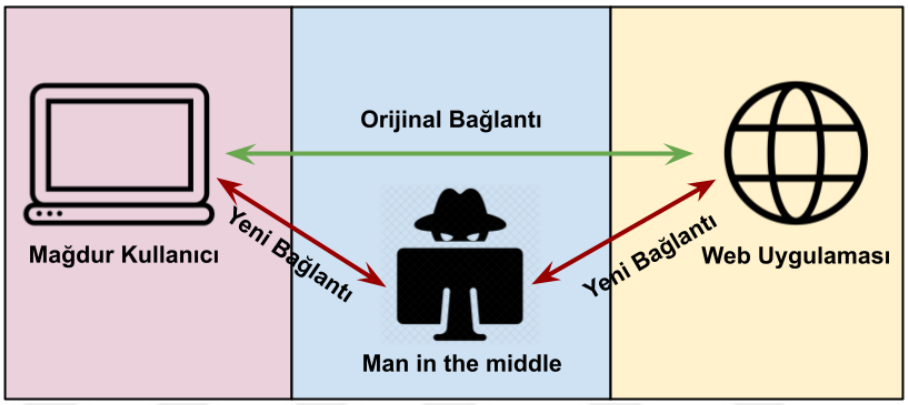
1.2.Phishing (Oltalama) Saldırıları

Pishing (oltalama) saldırıları parolalar, hesap ayrıntıları veya kredi kartı numaraları gibi hassas bilgileri elde etmek için kötü amaçlı bir web sitesinin meşru bir web sitesi kimliğine büründüğünde ortaya çıkan saldırı türüdür. Saldırganlar genellikle kurbanlarına ekleri açmaya veya bağlantıları tıklamaya zorlayarak, korku, merak, aciliyet ve açgözlülük gibi duyguları kullanır (Basnet ve diğ., 2008). Son zamanlarda en yaygın kullanılan saldırı türlerinden biridir.

1.3. Man in the Middle Saldırıları

Man in the middle saldırısı, HTTPS (Secure Hyper Text Transfer Protocol) sunucusunun web tarayıcısına ortak anahtarı olan bir sertifika göndermesiyle yapılmaktadır. Bu saldırı türünde HTTPS sunucusunu doğrulayan orijinal sertifika güvenilir olmayan bir sertifika ile değiştirilir. Böylelikle tüm iletişim yolu savunmasız bırakılır. Kullanıcı, tarayıcı bir uyarı bildirimi gönderdiğinde sertifikayı tekrar kontrol etmeyi ihmal ederse saldırı başarıyla gerçekleştirilmiş olur (Callegati, 2009).

Man in the middle saldırısı, bir mektubun içeriğini okuyan veya hatta içeriğini değiştiren postacı tarafından ele geçirilmesi olarak benzetilebilir. Örneğin; içinde kötü amaçlı yazılım yüklü bir kablosuz yönlendirici bulunan ücretsiz Wi-Fi bağlantısı sağlayan alışveriş merkezi gibi halka açık bir yerde man in the middle saldırısı gerçekleştirilebilir. Bir kullanıcı o sırada bir bankanın web sitesini telefondan veya dizüstü bilgisayardan ziyaret ederse, kullanıcının banka kimlik bilgilerini saldırganlar tarafından kaydedilebilir (Gangan, 2015). Man-in-the-middle saldırılarının genel olarak çalışma prensibi Şekil 1.2'de gösterilmiştir.



Şekil 1.2. Man in the middle saldırısı çalışma prensibi (URL-2, 2021)

1.4. SQL Injection Saldırıları

SQL injection günümüzde web destekli veritabanlarına karşı en yaygın saldırı tekniklerinden biridir. Bu saldırılarda saldırgan SQL dili özelliklerinden faydalanarak, web uygulamalarının gizliliği, bütünlüğü ve kullanılabilirliğine etkin bir şekilde zarar vermeyi amaçlamaktadır. SQL injection, kullanıcılardan gelen girdileri okuyan ve bunu temel bir veritabanına SQL sorguları oluşturmak ve yürütmek için kullanan herhangi bir web uygulaması için ciddi bir tehdittir. Saldırganlar bu şekilde çalışan web uygulamalarına veri girişini işleyerek bir SQL sorgusuna kötü amaçlı SQL kodu ekleyip, rastgele SQL sorguları çalıştırabilir. Böylece hassas müşteri ve sipariş bilgilerini e-ticaret uygulamalarından çıkarabilir veya arka uç veritabanlarını ve veri sunucusu dosya sistemini tehlikeye atan güçlü güvenlik mekanizmalarına ciddi zararlar verebilir (Kemalis ve Tzouramanis, 2008). SQL injection saldırıları OSI referans modelinin uygulama katmanında gerçekleşmektedir.

1.5. DNS Tünelleme Saldırıları

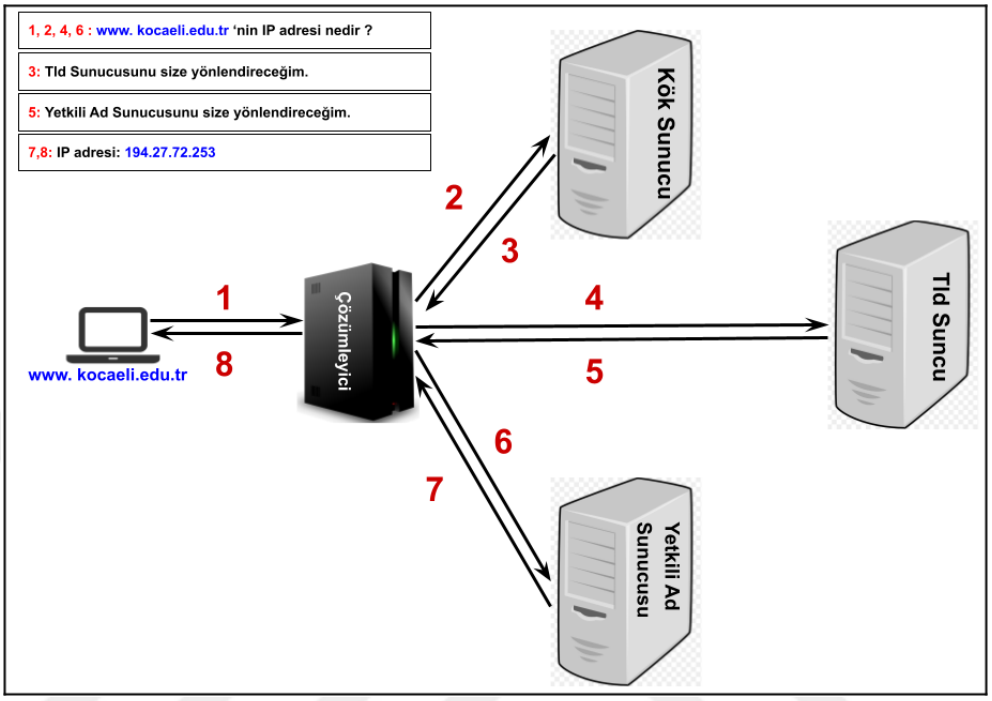
DNS'in en önemli görevi alan adları ile IP adreslerini çift taraflı dönüştürme işlemini gerçekleştirmektir. Örneğin kullanıcı tarayıcıya www.kocaeli.edu.tr adresini girdiğinde, bu adrese karşılık gelen IP adresi aşağıdaki adımlarla bulunmaktadır (Şekil 1.3).

1) Çözümleyici adrese karşılık gelen IP adresinin kendi önbelleğinde olup olmadığına bakar. Eğer önbelleğinde kayıtlıysa sorguya yanıt verir ve ilgili web sayfasına erişim sağlanır.
2) IP adresi çözümleyicinin önbelleğinde kayıtlı değilse, çözümleyici kök sunuculara erişmek istenilen adresi sorar. Kök sunucular gelen istekleri, adreslerini bildiği hiyerarşinin sonraki seviyesi olan Top Level Domain (TLD) sunucularına gönderir. Bu adımda yapılan işlemler sırasıyla Şekil 1.3'te 1, 2, 3 ve 4 numaralı oklarla gösterilmiştir.
3) TLD sunucusu üst düzey alan adlarının (.com,.org., .net gibi) adres bilgilerinin tutulduğu ve görev dağılımının ilk olarak yapıldığı yerdir. Örneğin www.kocaeli.edu.tr adresindeki, edu eğitim kuruluşu, tr ise Türkiye için ülke alan kodunu ifade eder. TLD sunucusundan sonra sorgu hiyerarşide son adım olan yetkili ad sunucusuna aktarılır. Bu adımda yapılan işlemler sırasıyla Şekil 1.3'te 5 ve 6 numaralı oklarla gösterilmiştir.
4) Ad çözümlemesini verimli olarak yapılmasını sağlamak için her bir coğrafi bölgeye yetkili ad sunucusu ilişkilendirilmiştir. Yetkili ad sunucusu isminden de anlaşılacağı gibi, DNS sorgularına ait IP adreslerini çözümleyiciye gönderen sunuculardır. Çözümleyiciye gelen IP adresi DNS istek talebinde bulunan bilgisayara gönderilerek ilgi web sayfasına erişim sağlanmış olur (Chandramouli ve Rose, 2006). Bu adımda yapılan işlemler sırasıyla Şekil 1.3'te 7 ve 8 numaralı oklarla gösterilmiştir.

DNS tünelleme ile neler yapılabileceği, bir senaryo üzerinde açıklanması daha anlaşılır olacaktır (Şekil 1.4). Örneğin, yerel ağdaki bir makineden abc.tunnel.mali.com adresi sorgulandığında sırayla şu adımlar yürütülür.

1) Yerel DNS sunucusuna iletilen sorgu ön belleğindeki kayıtlara bakar ve kayıt varsa kullanıcıya cevap döner.
2) Eğer kendi üzerinde kayıt yoksa öncelikle mali.com'dan sorumlu DNS sunucuyu bulur.

3) mali.com'dan sorumlu DNS sunucuyu bulduktan sonra tunnel.mali.com alt domaininden kimin sorumlu olduğunu sorar ve alacağı cevaba abc.tunnel.mali.com adresini sorar.

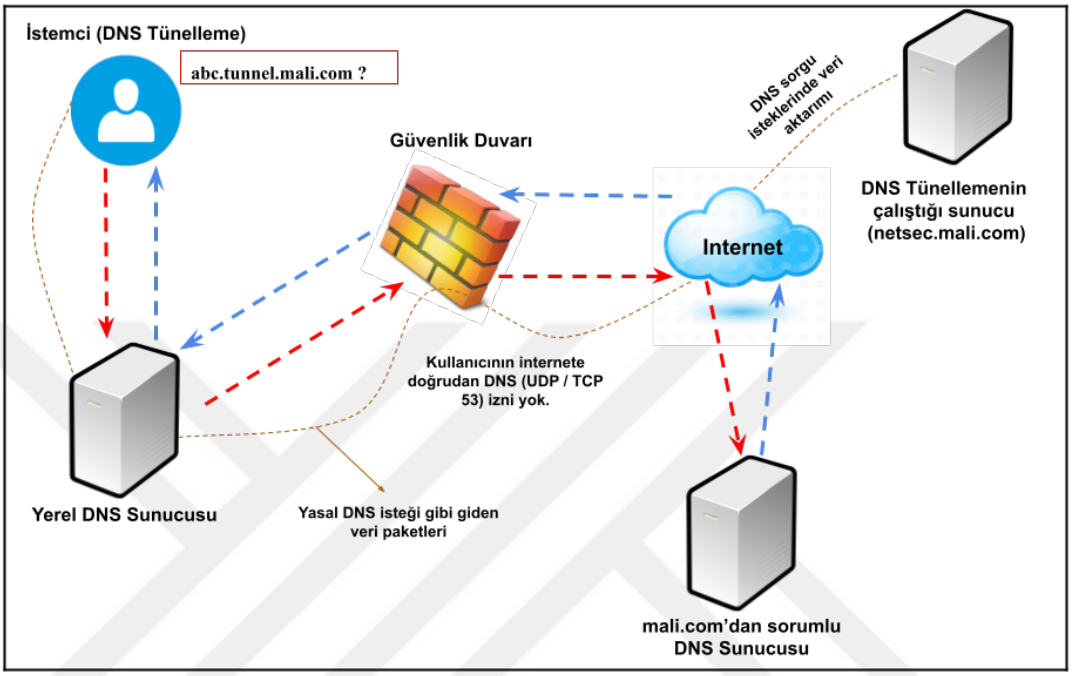


Şekil 1.3. DNS protokolü çalışma prensibi

Böylelikle yerel ağdan yapılan herhangi bir DNS isteği abc.tunnel.mali.com'dan sorumlu DNS sunucusuna kadar gelmektedir (netsec.mali.com olduğunu varsayalım). Kötü niyetli bir kullanıcının, özel bir DNS isteği oluşturduğu ve sorgulama kısmının haricinde kalan alana istediği verileri (abc yerine xyz byte) yerleştirip gönderdiği düşünülürse, bu istek hiçbir değişikliğe uğramadan netsec.mali.com adresinin UDP/53 portuna kadar gelecektir. netsec.mali.com adresinde de özel bir uygulama çalıştırılarak yine özel istemciden gelen veriler yorumlanabilir ve böylece DNS tünelleme yoluyla veri sızdırma işlemi gerçekleştirilir (Önal, 2016).

DNS sorgu ve yanıtlarına veri enjekte edilmesine izin vermesini sağlayan çeşitli DNS tünelleme araçları bulunmaktadır. Kurulumlarının kolay olması veya platform bağımsızlığı sebebiyle bu araçlardan popüler olanları Iodine (Hangal ve diğ., 2005), DNSCAT2 (Yassine ve diğ., 2018) ve DNS2TCP (Al-Kasassbeh ve diğ., 2019)'dir. DNS Tünelleme aracı, tünel istemcisi ve sunucu arasında DNS sorgu ve yanıtları gönderirken daha önceden DNS paketlerinin uygun alanına yerleştirdiği bilgilerin

alışverişini sağlamakla yükümlüdür. Böylece DNS tünelleme sunucusu alınan veriyi hedef bir istemciye ileterek veri sızdırma işlemini gerçekleştirebilir (Merlo ve diğ., 2011). Amaçları aynı olmasına rağmen her bir aracın DNS tünelleme istemcisiyle sunucusu arasında tünel oluşturmak için kendine özgü mimarileri bulunmaktadır.



Şekil 1.4. DNS tünelleme saldırısı çalışma prensibi (Önal, 2016)

1.6. R2L, U2R ve Probe Saldırıları

R2L saldırıları, hedef makinede kullanıcı hesabı olmayan bir saldırganın, uzak bağlantı yoluyla o makineye yetkisiz erişim elde ederek paketler gönderdiği saldırılardır. U2R saldırıları, saldırganın sistemde normal bir kullanıcı hesabıyla başladığı ve süper kullanıcı ayrıcalıkları elde etmek için sistemdeki güvenlik açıklarını kötüye kullanma girişiminde bulunduğu saldırılardır. Probe saldırıları ise saldırganın ileride yapabileceği saldırıda sistemin zayıflıkları veya güvenlik açıklarını belirlemek için bir makine veya bir ağ aygıtı hakkında bilgi edinmeye çalıştığı saldırılardır (Paliwal ve Gupta, 2012).

2. SALDIRI TESPİT SİSTEMLERİ

Bilgisayar veya ağ sistemlerine karşı uygun olmayan veya şüpheli etkinlikleri tespit eden sistemlere ise saldırı tespit sistemleri denilmektedir. Günümüzde her gün çok sayıda ihlal yapıldığı için bilgisayar sistemlerini veya ağ cihazlarını güncel tutmak zordur. STS, bu tür sistemlerin kullanımını yazılım veya donanım aracılığıyla otomatik olarak izleyerek güvenli olmayan durumların ortaya çıkmasını tespit etmektedir. Bu durumlar, iç kullanıcıların yetkilerini kötüye kullanma girişimi ya da dış kullanıcıların güvenlik açıklarından yararlanma girişimi olabilir (Rights, 2003).

STS, tehditleri tespit etmek ve paketleri izlemek için stratejik olarak bir ağa yerleştirilir. Bu aşamadan sonra STS, farklı sistemlerden ve ağ kaynaklarından veri toplayarak ve verileri olası tehditler için analiz ederek, tehditler hakkında bilgi sunma, tehditleri tespit ettiğinde düzeltici adımlar atma ve bir ağdaki tüm önemli olayları kaydetme görevini üstlenmektedir (Hodo ve diğ., 2017).

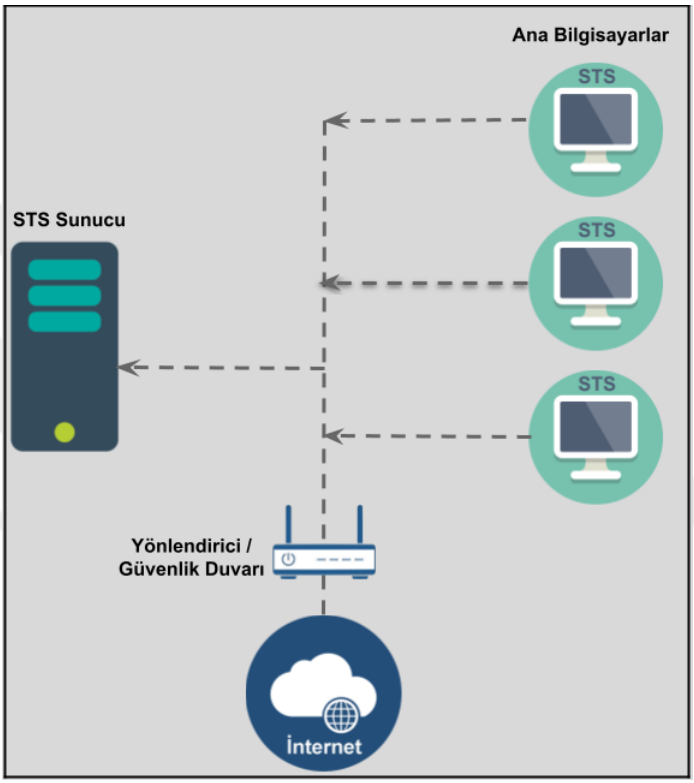
2.1. STS'lerin Sınıflandırılması

2.1.1. Algılamanın gerçekleştiği yere göre STS'ler

STS'ler algılamanın gerçekleştiği yere göre ana bilgisayar tabanlı, ağ tabanlı ve hibrit olmak üzere üçe ayrılmaktadır.

İlk geliştirilen STS türü olan ana bilgisayar tabanlı sistemler, yerel ana bilgisayar sisteminde oturum açma, uygunsuz dosya erişimi, onaylanmamış ayrıcalık yükseltme veya sistem ayrıcalıklarında değişiklikler gibi yerel ana bilgisayarla sınırlı olağandışı etkinlikleri kontrol ederler. Kontrol işleminde bilgi kaynağı olarak sıklıkla ana sistemin denetim ve günlükleme mekanizmasından faydalanılır. Elde edilen bilgi kaynağını analiz etmek için kural tabanlı motorlar kullanılmaktadır. Örneğin; "Süper kullanıcı ayrıcalığı yalnızca su komutu ile elde edilebilir." Bu nedenle, kök hesaba yapılan art arda oturum açma girişimleri bir saldırı olarak değerlendirilebilir (URL-3, 2021).

Ana bilgisayar tabanlı STS'ler bütünlük denetimi, olay korelasyonu, günlük analiz, politika uygulama, kök kullanıcı takımı algılama, işlemci, bellek, sabit disk ve pil kullanımı ve uyarı gibi fonksiyonları yerine getirme becerisine sahiptir. Bu sistemlerde herhangi bir kötü amaçlı etkinlik olup olmadığını gösteren raporlar oluşturulur ve bu raporlar analiz edilir. Bunun için sistemlere program veya araçların yüklenmesi gereklidir (Othman ve diğ., 2018). Ana bilgisayar tabanlı STS mimarisi Şekil 2.1'de gösterilmiştir.



Şekil 2.1. Ana bilgisayar tabanlı STS mimarisi

Ana bilgisayar tabanlı STS'ler, sistem tarafından üretilen günlüklere ve diğer uygulamalara bağlı olduğundan, trojan programları gibi bazı saldırı türlerini algılamasını sağlayabilir. Ayrıca bu sistemlerde şifrelenmiş bilgilerin analiz edilmesi mümkündür. Bu durumlar ağ tabanlı saldırı tespit sistemlerine göre önemli avantajlar sağlamaktadır. Ana bilgisayar tabanlı STS'lerin dezavantajları ise şu şekilde sıralanabilir (Saxena ve diğ., 2017):

1) Ana bilgisayar tabanlı STS'ler, izlenen her ana bilgisayarda yapılandırılıp yönetildikleri için bu sistemlerin kurulumu, yapılandırılması ve çalıştırılması için

karşılaştırılabilir boyuttaki bir ağ tabanlı STS çözümünden daha fazla yönetim çabası gerektirir.

2) Ana bilgisayar tabanlı STS'ler, hem doğrudan saldırılara hem de ana bilgisayar işletim sistemine yönelik saldırılara açık olduğu için sistemin işlevselliğinin bozulmasına veya kaybına neden olabilir.

3) Ana bilgisayar tabanlı STS'ler, çoklu ana bilgisayar taramasını algılayacak şekilde optimize edilmediği için yönlendiriciler veya anahtarlar gibi ağ cihazlarının taramasını algılayamaz. Bu yüzden ağdaki birden fazla cihazı kapsayan saldırıların tespit edilmesi mümkün değildir.

4) Ana bilgisayar tabanlı STS'ler, sistem tarafından üretilen günlükleri tutmak için büyük miktarda disk alanına ihtiyaç duyar. Ayrıca büyük miktardaki bu verilerin işlenmesi sırasında ciddi performans ek yüküne neden olabilir.

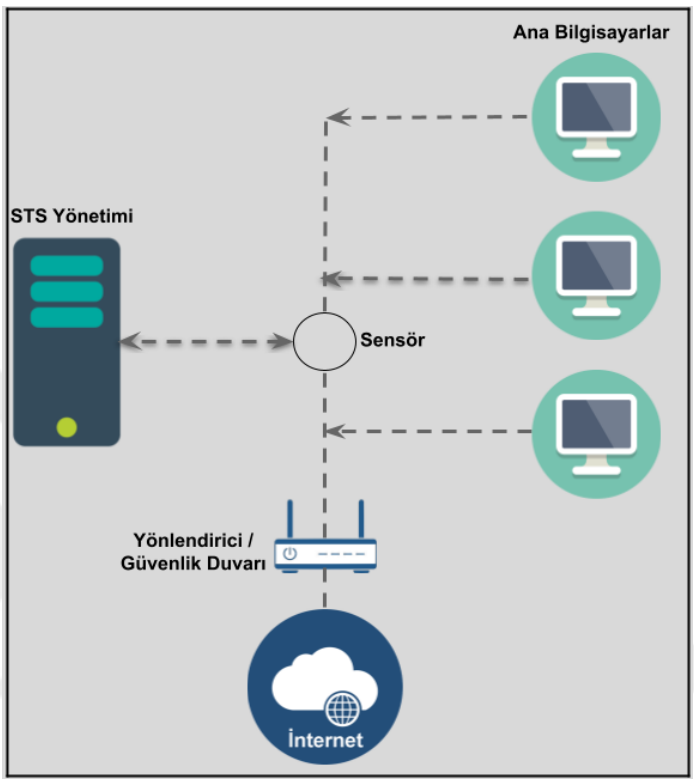
Ağ tabanlı STS'ler ana bilgisayar tabanlı sistemlerden farklı bir yaklaşım ile çalışmaktadır. Bu sistemler bilgilerini ana bilgisayarlardan değil, ağ segmenti üzerinde dolaşırken ağ trafik akışından toplanmaktadır. Bu yüzden çalışma mantıkları telefon dinleme konseptine benzerdir. Ağ tabanlı sistemler ağda hareket eden tüm paketlerin içeriklerini ve başlık bilgilerini inceleyerek saldırıları veya anormal davranışları kontrol ederler. Kontrol işlemi ağ sensörleri sayesinde gerçekleşir. Ağ sensörleri neyin bir saldırı oluşturacağına ilişkin kurallar olan saldırı imzaları ile donatılmıştır. Ağ tabanlı sistemlerin çoğu, gelişmiş kullanıcıların kendi saldırı imzalarını tanımlamasına izin verir. Böylece her bir ağın ihtiyaçlarına ve kullanım türlerine göre özelleştirmenin bir yolu sağlanmış olur. Sensörler daha sonra bu imzaları yakaladıkları trafikle karşılaştırır ve sistemin saldırı trafiğini tanımlamasını sağlamış olur (Rights, 2003). Ağ tabanlı STS mimarisi Şekil 2.2'de gösterilmiştir.

Ağ tabanlı STS'lerin iyi yerleştirilmesi durumunda büyük bir ağı izleyebilir. Ayrıca kurulan sistem saldırganlara görünmez hale getirilebilir. Ağ tabanlı sistemlerin bu özellikleri ana bilgisayar tabanlı STS'lere göre önemli avantajlar sağlamaktadır. Ağ tabanlı STS'lerin dezavantajları ise şu şekilde sıralanabilir (Gaddam ve Nandhini, 2017):

1) Ağ tabanlı sistemlerde şifrelenmiş bilgiler analiz edilemez.

2) Yüksek trafik sırasında paketleri işleyemediği için saldırı olup olmadığını tespit edemeyebilir.

3) Bir saldırının başlatılıp başlatılmadığını ayırt edilebilir, saldırının başarılı olup olmadığı ayırt edilemez.



Şekil 2.2. Ağ tabanlı STS mimarisi

Ana bilgisayar tabanlı ve ağ tabanlı sistemlerin iç ve dış ağlara göre güçlü ve zayıf yönleri Tablo 2.1'de gösterilmiştir.

Tablo 2.1. Ana bilgisayar tabanlı ve ağ tabanlı STS'lerin karşılaştırılması (URL-4, 2021)

Davranış	Ana bilgisayar tabanlı STS	Ağ tabanlı STS
İzinsiz girişlerin tespiti	İç ağda güçlü	Dış ağda güçlü
İzinsiz girişleri önleme	İç ağda güçlü	Dış ağda güçlü
Saldırılara yanıt	Gerçek zamanlı saldırılarda zayıf	Gerçek zamanlı saldırılarda güçlü
Hasar tespiti	Güçlü	Zayıf
Saldırıları tahmin etme	Güçlü	Zayıf

Ana bilgisayar ile ağ tabanlı STS'lerin birlikte kullanıldığı sistemlere hibrit sistemler denilmektedir. Hibrit sistemlerde her iki sistemden bilgi toplanarak bir analiz

metodolojisi uygulanır. Bu sistemler, her iki sistemin dezavantajlarının üstesinden gelebilecek şekilde tasarlanmıştır. Bu yüzden günümüzde en çok hibrit sistemler tercih edilmektedir (Chauhan ve Chandra, 2013).

2.1.2. Saldırı tespit yaklaşımına göre STS'ler

Saldırı tespit yaklaşımına göre STS'ler imza ve anomali tabanlı olmak üzere ikiye ayrılmaktadır.

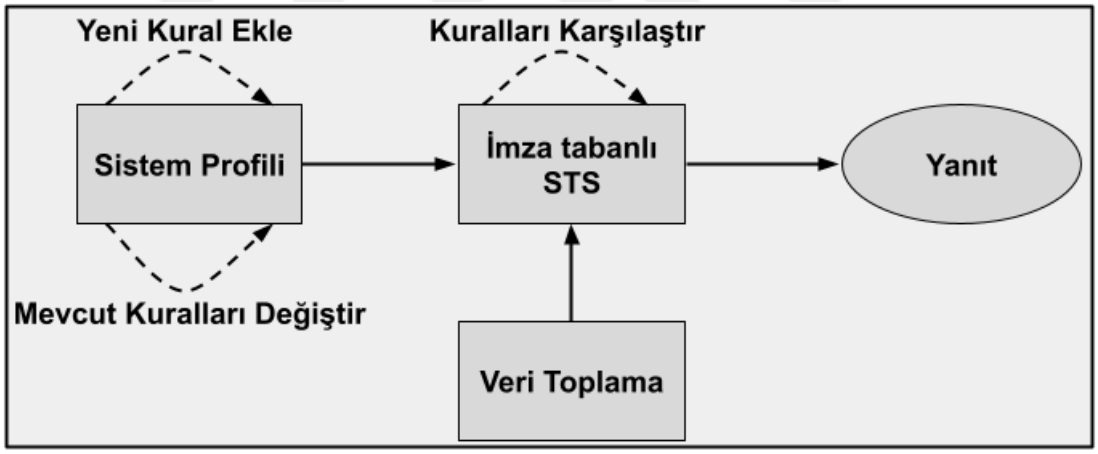
İmza tespiti ya da başka bir deyişle kötüye kullanım tespiti, saldırı modellerinin, yetkisiz ve şüpheli davranışların geçmiş faaliyetlere dayalı olarak öğrenildiği yaklaşımlardır. Öğrenilen modeller hakkındaki bilgi, bir ağda daha sonraki benzer saldırıları tespit etmek veya tahmin etmek için kullanılmaktadır. Bu modeller ağa ve bilgi işlem ana bilgisayarlarına yönelik tehdit olarak tanımlanan günlük dosyası veya veri paketi modellerini içermektedir. Her günlük dosyası, 0 ve 1 ikili bitlerinden oluşan kendine özgü bir model sergileyen kendi imzasından oluşmaktadır (Sen ve Mehtab, 2020).

İmza tabanlı algılama sisteminin nasıl çalıştığı Şekil 2.3'te gösterilmektedir. Bu algılama sistemleri, herhangi bir olası saldırıyı veya kötü niyetli etkinliği tespit etmek için geçmiş saldırılardan öğrenilen kalıpları veya imzaları bir ağdaki etkinliklerle eşleştirmeye çalışan algoritmaları çalıştırır. Ağdaki herhangi bir etkinliğin imzası, saldırı imzası veritabanındaki herhangi bir etkinliğin imzasıyla eşleşirse, algoritma sisteme bir uyarı vermektedir. Algılama sistemindeki modül, saldırı için daha fazla inceleme başlatır ve bu tür saldırılara karşı savunma için uygun güvenlik modüllerini çağırmaya başlar. Saldırı tespit sistemi tarafından gerçek bir saldırı olarak bulunursa ve bu saldırı yanlış bir alarm değilse, saldırı imzalarının mevcut veritabanı yeni saldırının imzasıyla güncellenir (Sen ve Mehtab, 2020).

İmza tabanlı tespit sisteminin etkinliği, sistemin saldırı imza veritabanında yakalanan saldırı kalıpları, imzaların bilgisinin tamlığı ve yeterliliğine bağlıdır. Bir siber altyapıdaki veya bir bilgi işlem makineleri ağındaki saldırı ve sistem güvenlik açıklarının bilgisini yakalamak büyük ölçüde bu alanda çalışan uzman kişilere bağlıdır. Alan uzmanlarının bilgi ve becerileri kişiden kişiye önemli ölçüde farklılık gösterdiği için, imza algılama sistemlerinin tasarımı çoğu zaman yetersiz kalabilmektedir. Ayrıca

saldırı imzasında hafif bir değişiklik veya önceden bilinen saldırıların bir kombinasyonu imza tabanlı STS'lerde algılama işlemini imkansız bir görev haline getirebilmektedir. Bu durum, imza tabanlı saldırı tespit sistemi gibi benzerliğe dayalı çalışan öğrenme sistemlerinde büyük bir sorundur. Bu sorunun üstesinden gelebilmek ve başarı oranını yükseltmek için imza tabanlı sistemlerde saldırı bilgilerinin düzenli olarak güncellenmesi gerekmektedir (Herrero ve Corchado, 2014). Bu dezavantajlara rağmen imza tabanlı sistemlerin avantajları şu şekilde sıralanabilir (Bace ve Mell, 2001):

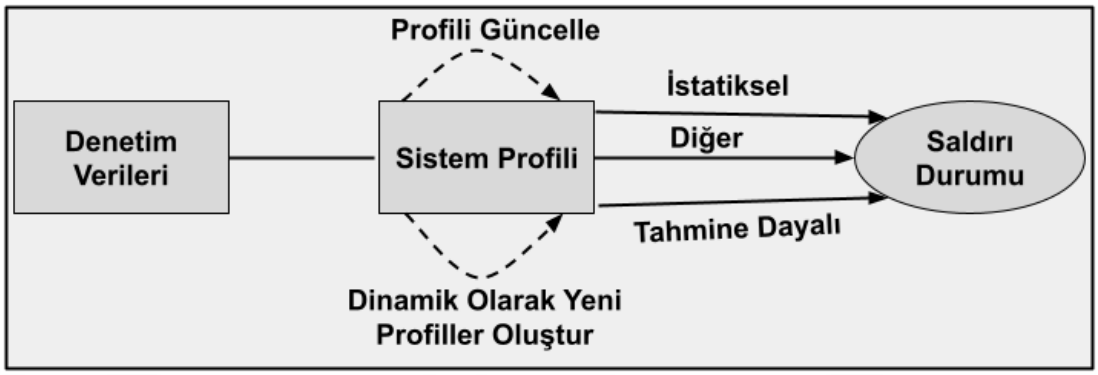
1) İmza tabanlı sistemler çok fazla sayıda yanlış alarm oluşturmadan saldırıları tespit etmede çok etkilidir.
2) Belirli bir saldırı aracı veya tekniğinin kullanımını hızlı ve güvenilir bir şekilde teşhis edilebilir.
3) Sistem yöneticilerinin sistemlerindeki güvenlik sorunlarını takip etmelerine ve olay işleme prosedürlerini başlatmalarına olanak sağlayabilir.



Şekil 2.3. İmza tabanlı STS çalışma prensibi

Saldırı tespit yaklaşımında bir başka yöntem olan anomali tespiti bir ana bilgisayar veya ağdaki olağandışı davranışları tanımlamayı amaçlamaktadırlar. Bu sistemler saldırıların normal faaliyetten farklı olduğu varsayımına göre çalışmaktadırlar. Bu nedenle saldırılar bu farklılıkları tanımlayan sistemler tarafından tespit edilebilirler. Anomali tespitini yapan sistemler kullanıcıların, ana bilgisayarların veya ağ bağlantılarının normal davranışını temsil eden profiller oluşturur. Bu profiller, sistemin normal davranış gösterdiği çalışma süresi boyunca toplanan geçmiş verilerden oluşturulmuştur. Anomali tespiti yapan detektörler bu verileri toplar ve

izlenen faaliyetin normdan ne zaman saptığını belirlemek için çeşitli önlemler kullanır (Bace ve Mell, 2001). Tipik bir anomali tabanlı algılama sistemi Şekil 2.4'te gösterilmektedir.



Şekil 2.4. Anomali tabanlı STS çalışma prensibi

Bir ağda yeni bir saldırı başlatıldığında, saldırı imzası mevcut saldırı imzaları veritabanında bulunmadığından imza tabanlı sistemler saldırıyı algılayamaz. Anomali tabanlı sistemlerin en önemli avantajlarından biri, yeni ve görünmeyen saldırıları tespit etme ve önceden uyarı verme yeteneğine sahip olmasıdır. Bu yeteneği sayesinde anomali tabanlı STS'ler, imza tabanlı STS'lerde imzaları tanımlamak için gerekli bilgileri üretmeye katkı verebilmektedir. Buna rağmen anomali tespitinde, tüm müdahaleci faaliyetlerin zorunlu olarak anormal olduğu varsayılır. Bu varsayım iki temel soruna neden olmaktadır. Birincisi, müdahaleci olmayan anormal etkinliklerin, müdahaleci olarak tespit edilmesidir. İkincisi ise anormal olmayan müdahaleci faaliyetlerin tespit edilememesidir. Bu sorunların ortadan kaldırılması için anomali tabanlı STS'lerinde eşik seviyelerinin seçimi ve trafiği izlemek için seçilen özelliklerin iyi belirlenmesi önem arz etmektedir (Herrero ve Corchado, 2014).

Anomali tabanlı tespit sistemlerinde birçok yaklaşım türü vardır. Bu yaklaşımlardan en çok tercih edilenleri istatiksel, tahmine dayalı ve yapay sinir ağları kullanımına dayalı yaklaşımlardır. İstatiksel yaklaşımlarda saldırı türleri için davranış profilleri oluşturulur. Sistemin çalışması sırasında anomali detektörü sürekli olarak mevcut profilin orijinal profilden varyansını oluşturur. Bu durumda, davranış profilini etkileyen etkinlik ölçütleri, kullanılan işlemci süresi, bir zaman dilimindeki ağ bağlantılarının sayısı gibi çeşitli istatiksel özeliklerin bilgisi kayıt altına alınır. Böylece kullanıcıların davranışlarının uyarlamalı olarak öğrenmeleri sağlanmış olur. İstatiksel

yaklaşımlarla ilgili en büyük sorun, izlenecek önlemlerin seçimidir. Müdahaleci etkinlikleri doğru bir şekilde öngören tüm olası önlemlerin alt kümesinin ne olduğu tam olarak bilinememesi, sistemin korunması için oluşturulan statik yöntemlerin yetersiz kalmasına sebep olabilir. Tahmine dayalı yaklaşımda, önceden meydana gelen olaylara bağlı olarak gelecekteki olaylar tahmin edilmeye çalışılır. Tahmin etme işlemi kural tabanlı kalıpların belirlenmesiyle yapılmaktadır. Bu yaklaşımda geleneksel yöntemlerle zor olan anormal etkinlikleri tespit edebilir. Ayrıca, bu model kullanılarak oluşturulan sistemler değişikliklere oldukça uyumludur. Tahmine dayalı yaklaşımdaki en büyük sorun kurallar tarafından tanımlanmayan bazı saldırı senaryolarının tespit edilememesidir. Yapay sinir ağlarının kullanımına dayalı yaklaşımda ise amaç bir kullanıcının bir sonraki eylemini veya komutunu tahmin etmek için sinir ağını eğitmektir. Eğitim işleminden sonra ağ, gerçek komutları ağda mevcut olan gerçek kullanıcı profiliyle eşleştirmeye çalışır. Bu yaklaşımda başarı herhangi bir istatistiksel varsayıma bağlı olmadığı için yeni kullanıcı toplulukları için değiştirilmeleri daha kolaydır. Fakat eğitim ve test verilerinde bilinmeyen dağılımların gürültüsü olması normal ve anormal davranışların sürekli olarak değişmesine sebep olmaktadır. Bu sorun, bir ağda anomali tespitini zor bir görev haline getirmektedir (Sundaram, 1996).

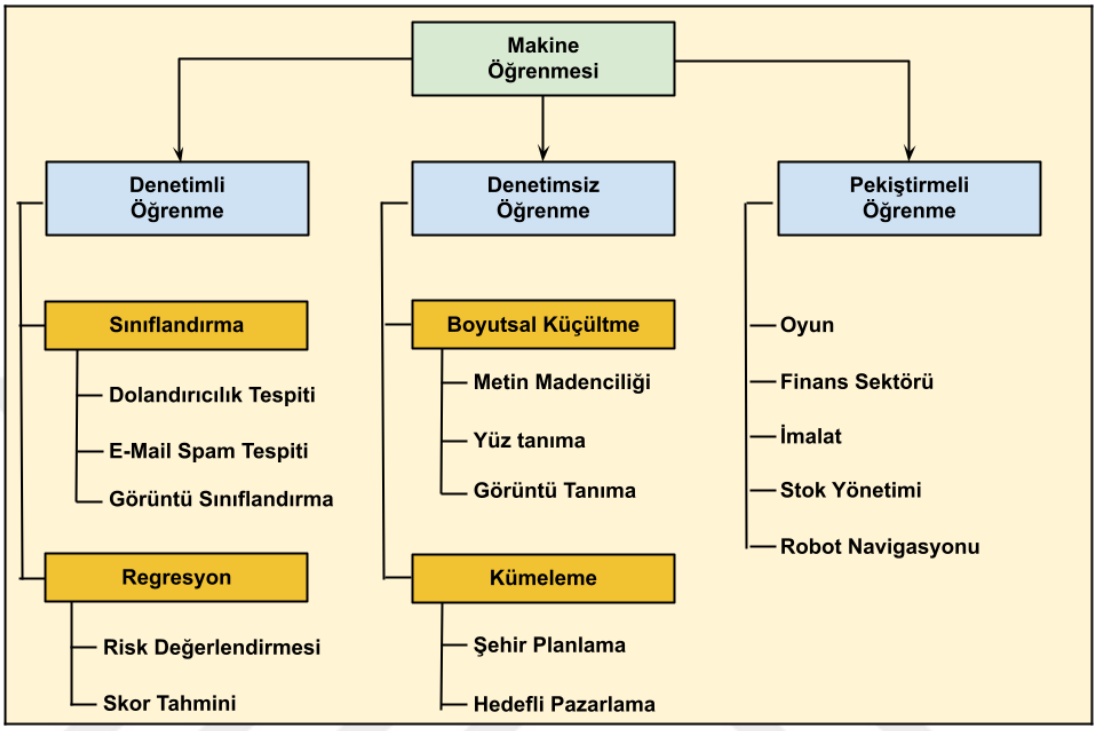
3. MAKİNE ÖĞRENMESİ

Makine öğrenmesi istatistik, yapay zeka ve bilgisayar biliminin kesiştiği araştırma alanıdır. Verilerden bilgi çıkarmakla ilgilidir ve tahmine dayalı analitik veya istatistiksel öğrenme olarak da bilinir. Makine öğrenimi yöntemlerinin uygulanması, son yıllarda günlük yaşamda her alanda popüler hale gelmiştir. Hangi filmlerin izleneceği, hangi yiyeceklerin sipariş edileceği veya hangi ürünlerin satın alınacağı gibi otomatik önerilerden kişisel fotoğraflarda arkadaş tanımaya kadar birçok modern web sitesi veya cihaz, özünde makine öğrenimi algoritmalarına sahiptir. Ticari uygulamaların dışında, makine öğreniminin günümüzde veriye dayalı araştırmanın yapılma şekli üzerinde önemli bir etkiye sahiptir (Müller ve Guido, 2016).

Makine öğrenmesinde amaç, örnek veriler veya geçmiş deneyimler kullanılarak bir başarım kriterini optimize etmek için bilgisayarları programlamaktır. Bunun için probleme özgü matematiksel modeller oluşturulmaktadır. Makine öğrenmesinde temel görev bir örnekten çıkarımda bulunmak olduğu için, matematiksel modeller oluşturmada istatistik teorisi kullanılmaktadır. Burada bilgisayar biliminin rolü iki yönlüdür: Birincisi, eğitimde, optimizasyon problemini çözmek ve sahip olunan büyük miktardaki veriyi depolamak ve işlemek için gerekli olan algoritmanın varlığıdır. İkincisi ise bir model öğrenildikten sonra, gösterimi ve çıkarımı için algoritmik çözümünün verimli olmasıdır (Alpaydın, 2004).

Makine öğrenmesi algoritmalarını eğitmenin farklı yöntemleri vardır. Bu yöntemlerin her birinin kendine özgü avantajları ve dezavantajları mevcuttur. Uygun makine öğrenmesi yönteminin belirlenmesinden önce hangi tür veri aldığına bakmak gerekir. Makine öğrenmesinde etiketli ve etiketlenmemiş veriler olmak üzere iki tür veri vardır. Etiketli veriler hem girdi hem de çıktı parametrelerinin tamamen makine tarafından okunabilir bir düzende olduğu verilerdir. Etiketli verilerin algoritmalarda işlenmesi kolaydır fakat başlangıçta verileri etiketlemek için çok fazla insan emeğine ihtiyaç duymaktadır. Etiketlenmemiş verilerde, makine tarafından okunabilir bir biçimde parametrelerden yalnızca biri vardır veya hiç yoktur. Bu durum, insan emeğine olan ihtiyacı ortadan kaldırmasına rağmen, başarı elde edilmesi için daha karmaşık

çözümler gerektirir. Makine öğrenmesi için birçok metadoloji ve algoritma olmasına rağmen, öğrenme türlerine göre denetimli, denetimsiz ve takviyeli öğrenme olmak üzere üçe ayrılır (Şekil 3.1).



Şekil 3.1. Makine öğrenmesi türleri

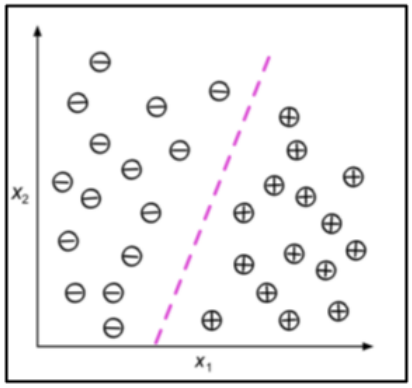
3.1. Denetimli Öğrenme

Denetimli öğrenme en yaygın kullanılan ve başarılı makine öğrenmesi türlerinden biridir. Belirli bir girdiden belirli bir sonucu tahmin edilmek istendiğinde denetimli öğrenme teknikleri kullanılmaktadır. Bu teknikler eğitim veri setindeki giriş ve çıkış çiftlerinden bir makine öğrenimi modeli oluşturularak uygulanır. Denetimli öğrenmede amaç, daha önce hiç görülmemiş veriler için doğru tahminlerde bulunmaktır. Denetimli öğrenme yöntemine e-posta spam filtreleme işlemini örnek olarak verilebilir. Yeni bir e-postanın spam olup olmadığını tahmin etmek için spam ve spam olmayan doğru şekilde etiketlenen e-postalardan oluşan bir topluluk üzerinde denetimli bir makine öğrenimi algoritması kullanılarak bir model eğitilebilir (Müller ve Guido, 2016).

Denetimli öğrenmede eğitim veri seti oluşturma işlemi etiketli verilerle yapıldığı için genellikle insan çabasına ihtiyaç duyulmaktadır. Ancak denetimli öğrenme sayesinde,

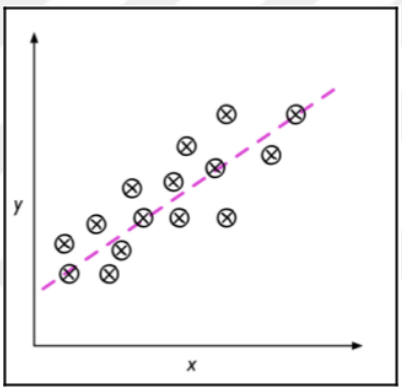
yüz ve ses tanıma, ürün önerileri ve satış tahmini gibi günlük uygulamalarda zor veya gerçekleştirilemez bir görev otomatikleştirilir ve genellikle karar verme süresi hızlandırılır.

Denetimli öğrenme, çıkışa bağlı olarak sınıflandırma ve regresyon problemleri olmak üzere iki şekilde gruplandırılabilir. Veri sınıflarını veya kavramlarını tanımlayan ve ayıran bir model bulma sürecine sınıflandırma denilmektedir. Sınıflandırmada model, sınıf etiketlerinin bilindiği veri nesnelerinin analizine dayalı olarak üretilmektedir. Modelin amacı sınıf etiketinin bilinmediği nesnelerin sınıf etiketini tahmin etmektir. Bu sınıf etiketleri, örneklerin grup üyelikleri olarak varsayıldığı ayrık değerlerdir (Han ve diğ., 2011). Şekil 3.2, 30 eğitim örneği verilen bir ikili sınıflandırma kavramını göstermektedir. Şekilde 30 eğitim örneğinin 15 eğitim numunesi negatif sınıf (eksi işaretler) ve 15 eğitim numunesi pozitif sınıf (artı işaretler) olarak etiketlenir. Bu senaryoda, veri setindeki her örneğin kendisiyle ilişkili x1 ve x2 olmak üzere iki değeri olduğu için veri seti iki boyutludur. Sınıflandırma işleminde, yeni verileri x1 ve x2 değerlerine göre bu iki kategorinin her birine sınıflandırabilen bir kuralı, başka bir deyişle kesikli çizgi olarak temsil edilen karar sınırını öğrenmek için denetimli bir makine öğrenimi algoritması kullanılmaktadır. Ancak, sınıf etiketleri kümesinin ikili yapıya sahip olması gerekmez. Denetimli öğrenme algoritması tarafından öğrenilen tahmine dayalı model, eğitim veri kümesinde sunulan herhangi bir sınıf etiketini etiketlenmemiş bir örneğe atayabilir. Bu modellere çok sınıflı bir sınıflandırma denilmektedir. Çok sınıflı sınıflandırmaya elle yazılmış karakter tanıma örnek olarak gösterilebilir. Karakter tanımada, alfabedeki her harfin birden çok el yazısıyla yazılmış örneğinden oluşan bir eğitim veri kümesi toplanabilmektedir (Raschka ve Mirjalili, 2017).



Şekil 3.2. İkili sınıflandırma (Raschka ve Mirjalili, 2017)

Denetimli öğrenmenin ikinci türü olan regresyon, sürekli sonuçları tahmin etme yöntemidir. Başka bir deyişle regresyon analizinde, sınıflandırmadaki ayrık sınıf etiketleri yerine eksik veya mevcut olmayan sayısal veri değerler tahmin edilmektedir. Regresyon problemi, girdi olarak etiketlenmiş örneklerin bir koleksiyonunu alan ve giriş olarak etiketlenmemiş bir örneği alıp bir hedef çıktısı alabilen bir model üretilerek çözülmektedir. Örneğin alan, yatak odası sayısı, konum gibi ev özelliklerine dayalı olarak konut fiyatının tahmin edilmesi bir regresyon problemidir. Şekil 3.3 doğrusal regresyon kavramını göstermektedir. Şekilde yordayıcı değişken (x) ve bir yanıt değişkeni (y) verildiğinde, bu verilere örnek noktalar ile uydurulan çizgi arasındaki mesafeyi en aza indiren düz bir çizgi sığdırılır. Böylece yeni verilerin sonuç değişkeni, bu verilerden öğrenilen kesişme, eğim vb. bilgiler kullanılarak tahmin edilir (Raschka ve Mirjalili, 2017).



Şekil 3.3. Doğrusal regresyon (Raschka ve Mirjalili, 2017)

Sınıflandırma algoritmasından en yaygın olanları KNN, DT, SVM, NB iken, en yaygın regresyon algoritmaları RF, lineer regresyon ve gradyan arttırmadır.

3.1.1. KNN

KNN denetimli makine öğreniminde en basit sınıflandırıcılardan biridir. KNN genellikle tembel bir öğrenciye benzetilir. Bunun sebebi, tahmin yapmak için teknik olarak bir model eğitmemesidir. KNN'de bir gözlemin en yakın k gözlemlerin en büyük oranının sınıfı olduğu tahmin edilmektedir (Albon, 2018). KNN tüm eğitim örneklerini bellekte tutar. Yeni, daha önce görülmemiş bir örnek geldiğinde, KNN x'e en yakın k eğitim örneğini bulur ve çoğunluk etiketini geri döndürür.

KNN'de iki nokta arasındaki mesafe bir uzaklık fonksiyonu ile verilir. Mesafe ölçüsü seçimi ve k değeri, algoritmayı çalıştırmadan önce analist tarafından belirlenmelidir. Mesafe ölçümü genellikle Öklid uzaklığı ile hesaplanır. Öklid uzaklığı Eşitlik (3.1)'deki formülle hesaplanmaktadır.

$$d(x_i,x_k)=\sqrt{\sum_{j=1}^{D}(x_i^{(j)}-x_k^{(j)})^2} \tag{3.1}$$

Eşitlik (3.1)'de d uzaklığı, x_i ve x_k veri noktalarını temsil etmektedir. KNN algoritmasına ait işlem adımları sırasıyla aşağıda verilmiştir.

1) K sayısı ve uzaklık fonksiyonu seçilir.
2) Sınıflandırmak istenen örneğin k-en yakın komşuları bulunur.
3) Sınıf etiketi oy çokluğu ile belirlenir.

3.1.2. DT

DT, böl ve yönet stratejisini uygulayan hem sınıflandırma hem de regresyon için kullanılabilen parametrik olmayan bir yöntemdir. Parametrik tahminde, tüm girdi alanı üzerinden bir model tanımlanır ve modelin parametreleri eğitim verilerinin tamamından öğrenilmektedir. Daha sonra herhangi bir test girişi için aynı model ve aynı parametre seti kullanılmaktadır. Parametrik olmayan tahminde ise, girdi alanı Öklid uzaklığı gibi bir mesafe ölçüsü ile tanımlanan yerel bölgelere bölünür. Her girdi için, o bölgedeki eğitim verilerinden hesaplanan değerlere karşılık gelen yerel model kullanılır. DT algoritmaları tıpkı geleneksel bir ağaç gibi dallara ve düğümlere sahiptir. Her karar düğümü, dalları etiketleyen farklı sonuçlara sahip bir test fonksiyonu uygular. Bir girdi verildiğinde, her düğümde bir test uygulanır ve sonuca bağlı olarak dallardan biri alınır. Bu süreç kökte başlar ve bir yaprak düğümüne ulaşılıncaya kadar yinelemeli olarak tekrarlanır ve yaprakta yazılan değer çıktı değerini oluşturur (Alpaydın, 2020).

DT algoritmaları sınıflandırmayı üç farklı adımda gerçekleştirir. Birinci adımda, algoritma hem ağaç büyümesini hem de ağaç budama işlevlerini tetikler. İkinci adımda, yineleme durumunda en yaygın olan hedef değişkenin değerine bağlı olarak her veri değeri bir sınıfa atanarak ağaç büyütülür. Son adımda ise, ortaya çıkan modelin

performansını optimize etmek ve aşırı uygunluğu önlemek için büyümüş ağacın budanması gerçekleştirilir (Berry ve Mohamed, 2019).

Karar ağaçlarında kullanılan birçok algoritma mevcuttur. En yaygın olanları ID3, C4.5, C5.0 algoritmalarıdır. Tez kapsamında C4.5 algoritması tercih edilmiştir. C4.5 algoritmasının adımları yinelemeli olarak şu şekildedir (URL-5):

1) Tüm öznitelikler için entropi ve kazanç değeri hesaplanır.
2) Hesaplanan kazanç değerine göre en iyi öznitelik seçilir ve karar düğümü oluşturulur.
3) Bir önceki adımda oluşturulan karar düğümüne göre veri seti bölünür.
4) Bölünen veri kümesi için bir önceki adımlar tekrarlanır ve yeni bir ağaç oluşturulur.
5) 4. adımda elde edilen ağaç 2. Adımdaki karar düğümüne eklenir.
6) Oluşturulan ağacın doğruluğunu test etmek için geri kalan eğitim örnekleri kullanılır.
7) Tüm örnekler doğru şekilde sınıflandırılmışsa algoritma sonlanır.

C4.5 algoritmasındaki öznitelikler için hesaplanan entropi ve kazanç değeri Eşitlik (3.2) ve Eşitlik (3.3)'teki formüllerle hesaplanmaktadır.

$$\text{Kazanç Değeri (S, A)} = \text{Entropi (S)} - \sum_{i\text{-}1}^{n} \frac{|S_i|}{|S|} \text{Entropi } (S_i) \tag{3.2}$$

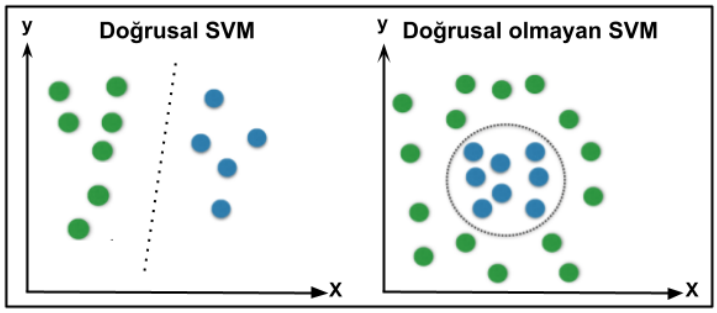
$$\text{Entropi (S)} = \sum_{i=1}^{n} (-p_i \log_2 p_i) \tag{3.3}$$

Eşitlik (3.2)'deki S bölünmüş veri kümesini, A öznitelikleri, n öznitelik sayısını, i iterasyon sayısını ve Eşitlik (3.3)'deki p, S_i'nin S'ye oranını ifade etmektedir.

3.1.3. SVM

SVM yapısal risk minimizasyonu ilkesine dayanan, küçük örneklem ve doğrusal olmayan sınıflandırma problemini çözebilen yaygın olarak kullanılan bir makine öğrenme yöntemidir. SVM iki türde olabilir. Bir veri kümesi tek bir düz çizgi kullanılarak iki sınıfa ayrılabiliyorsa bu tür veriler doğrusal olarak ayrılabilir demektir. Bu tür sınıflandırıcılara doğrusal SVM denilmektedir. Bir veri kümesinin düz bir çizgi

kullanılarak sınıflandırılamaması durumunda kullanılan sınıflandırıcılara ise doğrusal olmayan SVM denilmektedir (Şekil 3.4).



Şekil 3.4. Doğrusal ve doğrusal olmayan SVM

SVM'nin amacı, sınıflandırma probleminin doğrusal olarak ayrılabilir hale gelmesi için optimal ayırıcı alt düzlem aramasıdır. x_i'nin örnek veri ve y_i'nin örnek kategori olduğu $S=\{(x_i,y_i)_{i=1}^{n} x_i \in R^N, y_i \in \{-1,1\}\}$ bir veri seti varsayalım. Optimum ayırma düzlemi için çözüm, Eşitlik (3.4) ve Eşitlik (3.5)'teki nesnel fonksiyonlara ve kısıtlamalara dönüştürülür.

$$\min \frac{1}{2}\|w\|^2 \tag{3.4}$$

$$s.t. y_i (w x_i + b) \geq 1 \tag{3.5}$$

Eşitlik (3.4)'teki w ağırlık vektörünü ve Eşitlik (3.5)'teki b ise bias vektörünü ifade etmektedir. Sınıflandırmanın doğruluğunu arttırmak için ceza parametresi eklenir ve doğrusal ayrılma durumundaki dönüşümler yapılarak Eşitlik (3.6) ve Eşitlik (3.7) elde edilir.

$$\max W(a) \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} a_i a_i y_i y_i K(x_i x_j) \tag{3.6}$$

$$s.t. \sum_{i=1}^{n} a_i y_i \quad (0 \leq a_i \leq c;\ i=1,2,...,n) \tag{3.7}$$

Eşitlik (3.6)'daki $K(x_i x_j)$ çekirdek fonksiyonunu, Eşitlik (3.7)'deki c ceza parametresini ifade etmektedir. Eşitlik (3.7)'deki iç çarpım işlemiyle birlikte son

sınıflandırma sonucu Eşitlik (3.8)'deki karar fonksiyonu (f(x)) ile değerlendirilir (Yan ve Jia, 2018).

$$f(x)=\text{sign}\left(\sum_{i=1}^{n} a_i y_i < K(x_i x_j) > +b\right) \tag{3.8}$$

3.1.4. NB

NB, Bayes olasılık modeli tabanlı bir algoritmadır. NB algoritmasında sınıflandırma işlemi güçlü bir bağımsızlık varsayımı üzerinde çalışır. Bu durum bir özelliğin olasılığının diğerinin olasılığını etkilemediği anlamına gelir. NB algoritmasının başarısı eğitim verisindeki gürültü, sapma ve varyansa bağlı değişmektedir (Mukherjee ve Sharma, 2012).

NB algoritması, önceden tanımlanmış bir kategoriye ait yeni bir gözlemin olasılığını Bayes teorisine göre tanımlanan bir olasılık modeli kullanarak tahmin etmeyi amaçlar. Algoritmada her bir kategorinin önceki olasılığı, bir dizi değişkenle tanımlanan geniş bir eğitim verileri kümesine dayanarak değerlendirilir. Sınıflandırma, koşullu olasılık yoğunluk fonksiyonu ve posteriori olasılığı hesaplanarak tahmin edilmektedir (Tsangaratos ve Ilia, 2018). Sınıflandırılacak veri setinin $x=x_1, x_2 \ldots, x_n$ olduğu varsayıldığında NB algoritmasının işlem adımları aşağıda maddeler halinde sıralanmıştır (Kubat, 2017).

1) Her x_i ve her c_j sınıfı için, c_j'ye ait eğitim örnekleri arasında x_i'nin göreli frekansı olarak koşullu olasılık değeri, $P(x_i|c_j)$ hesaplanır.
2) Her sınıf için eğitim setinde bu sınıfın göreceli sıklığı olarak $P(c_j)$ değeri hesaplanır.
3) Karşılıklı bağımsız özniteliklerin varsayımı kullanılarak, $P(x|c_j)$ koşullu olasılık değeri hesaplanır. Koşullu olasılık değeri Eşitlik (3.9)'daki formülle hesaplanmaktadır.
4) Eşitlik (3.10)'daki en yüksek değere sahip sınıf seçilir.

$$P(x|c_j) = \prod_{i=1}^{n} P(x_i|c_j) \tag{3.9}$$

$$P(c_j).\prod_{i=1}^{n} P(x_i|c_j) \tag{3.10}$$

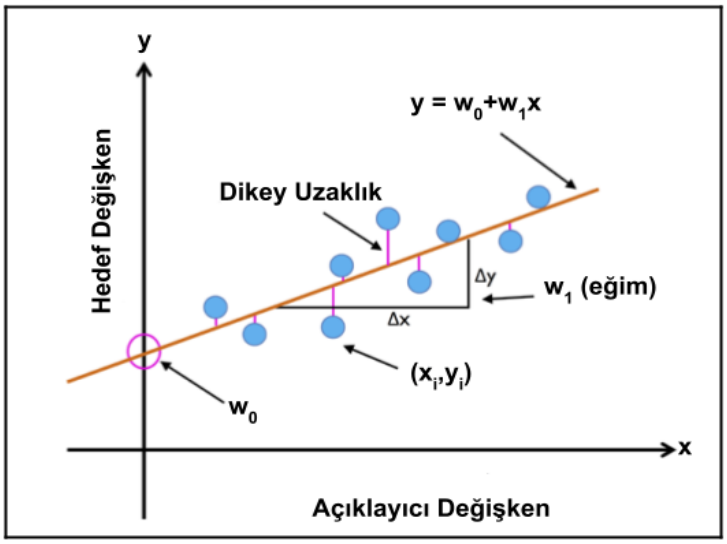
3.1.5. LR

LR, bir veya daha fazla özellik ile sürekli bir hedef değişken arasındaki ilişkiyi modellemeyi amaçlayan regresyon yöntemlerinden biridir. LR yönteminde çıktılar, kategorik sınıf etiketleri yerine sürekli bir ölçekte tahmin edilmektedir. Basit ve çoklu LR olmak üzere ikiye ayrılmaktadır.

Basit LR yönteminde, tek bir özellik ile sürekli değerli bir yanıt arasındaki ilişki modellenmektedir. Şekil 3.5'te gösterildiği gibi örnek noktalar aracılığıyla en uygun düz çizgi Eşitlik (3.11)'deki denklemle bulunmaya çalışılır.

$$y=w_0+w_1x \tag{3.11}$$

Eşitlik (3.11)'de y hedef değişkeni, x açıklayıcı değişkeni, w_0 y eksenine kesiştiği nokta, w_1 x değişkeninin ağırlık katsayısıdır. Basit LR modelinde amaç, x ile y değişkeni arasındaki ilişkinin tanımlanması için doğrusal denklemin ağırlıkların öğrenilmesidir. Böylece eğitim veri setinde olmayan yeni açıklayıcı değişkenlerin yanıtları tahmin edilir (Raschka ve Mirjalili, 2017).



Şekil 3.5. Basit LR modeli (Raschka ve Mirjalili, 2017)

Çoklu LR yönteminde ise, birden çok açıklayıcı değişken ile sürekli değerli bir yanıt arasındaki ilişki Eşitlik (3.12)'deki denklemle modellenmektedir.

$$y=w_0x_0+w_1x_1+...+w_mx_m=\sum_{i=0}^{m} w_ix_i \tag{3.12}$$

3.1.6. Topluluk öğrenme algoritmaları

Tek bir süper doğru modeli öğrenmeye çalışmak yerine, çok sayıda düşük doğruluklu modeli eğitmeye ve ardından yüksek doğruluklu bir meta-model elde etmek için bu düşük doğruluklu modeller tarafından verilen tahminleri birleştirmeye odaklanan öğrenme yöntemine topluluk öğrenme denilmektedir. Düşük doğruluklu modeller genellikle karmaşık modelleri öğrenemeyen ve bu nedenle genellikle eğitimde ve tahmin zamanında hızlı olan öğrenme algoritmalarıdır. Bu modellerde elde edilen ağaçlar sığdır. Ancak toplu öğrenmenin arkasındaki fikir, eğer ağaçlar aynı değilse ve her ağaç rastgele tahmin etmekten en azından biraz daha iyiyse, bu tür çok sayıda ağacı birleştirerek yüksek doğruluk elde edilebilmesidir. Topluluk öğrenmede torbalama ve güçlendirme olmak üzere iki yöntem vardır. Torbalama, eğitim verilerinin birbirinden farklı olan birçok kopyasını oluşturmaktan ve ardından düşük doğruluklu modelleri her kopyaya uygulayarak bunları birleştirmekten oluşur. Birleştirme, her düşük doğruluklu modele bir çeşit ağırlıklı oylama verilerek yapılmaktadır (Burkov, 2019). Güçlendirme yönteminin torbalama yönteminden farkı, öğrenme işleminin bağımsız olarak değil, sıralı olarak gerçekleşmesidir. Güçlendirmede, denetimli öğrenmeye dayalı olarak ağırlıklar art arda ayarlanır ve birden çok öğrenme sonucu aranır. Bu sonuçlar daha sonra genel doğruluğu artırmak için birleştirilir (Hamori ve diğ., 2018).

Torbalama yöntemiyle çalışan en yaygın algoritmalarından biri RF'dir. RF'de kullanılan ağaçlar, ikili yinelemeli bölümleme ağaçlarına dayanmaktadır. Bu ağaçlar, her bir değişken üzerinde ikili bölmeler kullanarak tahmin alanını böler. Ağacın kök düğümü tüm tahmin alanını kapsar. Bölünmemiş düğümlere terminal düğüm denilmektedir ve bu düğümler tahmin uzayının son bölümünü oluşturur. Her bölümlenmiş düğüm biri solda diğeri sağda olmak üzere iki alt düğüme ayrılır. Sürekli bir tahmin değişkeni için ayrım bir bölme noktası ile belirlenir. Tahmincinin bölünme noktasından daha küçük olduğu noktalar sola, büyük olanlar ise sağa gider. RF yönteminde çıkış değeri olan y'yi tahmin etmek için, tahmin fonksiyonu olan f(x) bulunmaya çalışılır. Eğitim verisinin $D=\{(x_1,y_1), (x_2,y_2), \ldots(x_n,y_n)\}$ olduğu varsayıldığında, RF algoritmasının işlem adımları aşağıda maddeler halinde sıralanmıştır (Cutler ve diğ., 2011).

1) j=(1, 2, … , J) olmak üzere, veri setinden n boyutundan bir D_j örneği alınır.
2) D_j için bir ağaç yapısı oluşturulur.
3) Tek bir düğümde tüm gözlemlerle başlanır. Durdurma kriteri karşılanana kadar bölünmemiş her düğüm için 4, 5 ve 6. adımlar yinelemeli olarak tekrarlanır.
4) Mevcut p tahmin edicilerinden rastgele m tahmin edici seçilir.
5) i. adımdaki m tahmin edicilerinde, tüm ikili bölmeler arasından en iyisi bulunur.
6) 5. Adımda elde edilen en iyi düğüm iki alt düğüme bölünür.
7) Yeni bir x noktasında tahmin yapmak için Eşitlik (3.13)'deki denklem kullanılır.

$$\max f(x)=\sum_{j=1}^{J}(h_j(x)) \tag{3.13}$$

Eşitlik (3.13)'teki $h_j(x)$, j. ağaç kullanılarak x'teki çıkış değişkeninin (y) tahminini temsil etmektedir.

Güçlendirme yöntemiyle çalışan en yaygın algoritmalar ise GB ve XGBoost'tur. GB, daha güçlü bir model oluşturmak için birden çok karar ağacını birleştiren bir yöntemdir. GB hem sınıflandırma hem de regresyon için kullanılabilir. RF yönteminin aksine, GB ağaçları seri bir şekilde inşa eder ve her ağaç bir öncekinin hatalarını düzeltmeye çalışır. GB yönteminde ağaçlar genellikle birden beşe kadar derinlikte sığ bir şekilde inşa edilir. Böylece model bellek açısından küçültülür ve tahminler daha hızlı bir şekilde yapılır. GB yönteminde amaç, sığ ağaçlar gibi birçok basit yani düşük doğruluklu modelleri birleştirmektir. Her ağacın yalnızca verilerin bir kısmı hakkında iyi tahminler sağladığı varsayılır ve bu nedenle performansı yinelemeli olarak iyileştirmek için daha fazla ağaç eklenir (Müller ve Guido, 2016).

GB algoritmasında f_t (x) ile tahmin kuralını belirtilir. L (y, f (x)) kayıp ve I (·) olay göstergesi fonksiyonları varsayılarak, GB algoritmasının işlem adımları aşağıda maddeler halinde sıralanmıştır (Blagus ve Lusa, 2017).

1) Eşitlik (3.14) kullanılarak $f_0(x)$ başlangıç değeri hesaplanır.

$$f_0(x)=\arg\min_{\gamma}\sum_{i=1}^{n}L(y_i,\gamma) \tag{3.14}$$

2) Eşitlik (3.15) kullanılarak negatif gradyanın bileşenleri (r_{it}) hesaplanır.

$$r_{it} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right], i=1, 2, ..., n \quad (3.15)$$

3) R_{jt} regresyon ağacı oluşturulur.

4) Eşitlik (3.16) kullanılarak güncel yanıt için güncellemeler hesaplanır.

$$\gamma_{jt} = \arg\min_{\gamma} \sum_{x_i \in R_{jt}} L(y_i, f_{t-1}(x_i) + \gamma), j=1, 2, ..., J_t \quad (3.16)$$

5) Eşitlik (3.17) kullanılarak $f_t(x)$ güncellenir.

$$f_t(x) = f_{t-1}(x) + \sum_{j=1}^{Jt} \gamma_{jt} I(x \in R_{jt}) \quad (3.17)$$

6) Durdurma kriteri karşılanana kadar (t=T), 2-5 arası adımlar tekrarlanır.

7) $f(x) = f_t(x)$ çıktısı elde edilir.

Bir başka güçlendirme algoritması olan XGBoost, gradyan kaldırma karar ağacı yöntemini temel alan, regresyon ağacı modellerini sınıflandırmaktadır. GB algoritmasından esinlenerek geliştirilen XGBoost, seyrek verilerin verimli işlenmesini sağlamasının yanı sıra paralel hesaplamayı ve yaklaşık ağaç oluşturmayı optimize etmektedir (Jiang ve diğ., 2019, Feng ve diğ., 2019). XGBoost algoritmasında ilk ağaç mevcut örnek veriler üzerinden tahmin edilir ve daha sonra tahmin ile gerçek değer arasındaki fark hesaplanarak ikinci ağacın tahmini yapılır. Böylece her adımda fark alınarak önceki eğitim sürecinde üretilen hata düzeltilir. Aynı zamanda XGBoost algoritmasında GB algoritmasında karşılaşılan overfitting problemini önlemek için amaç fonksiyonuna bir düzenleme terimi eklenir (Xu ve Wu, 2020). Düzenleme terimi algoritmanın karmaşıklığını kontrol ederek bu problemi çözer. XGBoost algoritmasında veri setindeki her bir örneğin m özelliği, etiket değerinin $\hat{y}$ olduğunu ve toplam k ağaç olduğu varsayıldığında, tüm modelin bu örnek üzerinde verdiği tahmin sonuçları Eşitlik (3.18)'deki gibi hesaplanır.

$$\hat{y}_i = \sum_{k}^{K} f_k(x_i) \quad (3.18)$$

Eşitlik (3.18)'de f_k; k. ağacı, x_i; örneğe karşılık gelen özellik vektörünü ve $f_k(x_i)$; k. ağacın i. örneğinin tahmin edilen skorunu temsil etmektedir. XGBoost'da amaç fonksiyon (A) Eşitlik (3.19)'daki gibi hesaplanır.

$$A=\sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} Q(f_k) \tag{3.19}$$

Eşitlik (3.19)'da i; veri setindeki i. örneği, N; k. ağaca aktarılan toplam veri miktarını, K; kurulan tüm ağaçları, *l*; gerçek etiket ile tahmin edilen değer arasındaki farkı ölçen hata fonksiyonunu, Q ise; değişkenlerdeki gürültüyü azaltmak için kullanılan düzenleme terimini ifade eder. t. yinelemede amaç fonksiyon Eşitlik (3.20)'deki gibi hesaplanmaktadır.

$$A^{(t)}=\sum_{i=1}^{N} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + Q(f_t) \tag{3.20}$$

Eşitlik (3.20)'de f_t fonksiyonu, tahmin edilen sonuç ile gerçek sonuç arasındaki hatayı en aza indirmek için seçilen fonksiyondur. Eşitlik (3.20)'nin karmaşıklığını azaltmak için ikinci dereceden Taylor açılımı yapıldığında Eşitlik (3.21) elde edilir.

$$A^{(t)}=\sum_{i=1}^{N}\left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + + \frac{1}{2} h_i f_t^2(x_i) \right] + Q(f_t) \tag{3.21}$$

$$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \tag{3.22}$$

$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \tag{3.23}$$

Eşitlik (3.21)'deki g_i (Eşitlik (3.22)) hata fonksiyonunun birinci türevi, h_i (Eşitlik (3.23)) ise hata fonksiyonunun ikinci türevidir. Böylece amaç fonksiyon her veri noktasında hata fonksiyonunun sadece birinci ve ikinci türevlerine bağlı olacak şekilde düzenlenir (Jiang ve diğ., 2019, Li ve diğ., 2020). Amaç fonksiyonunun nihai denklemi Eşitlik (3.24)'de gösterilmiştir.

$$A^{(t)}=\sum_{i=1}^{N}\left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + Q(f_t) \tag{3.24}$$

3.2. Denetimsiz Öğrenme

Makine öğrenmesinde yararlı bilgilerin her zaman sınıfların etiketlenmesiyle elde edildiğini düşünmek yanlış olur. Yararlı bilgiler, sınıfları bilinmeyen örneklerden de elde edilebilir. Bu şekilde öğrenilebilen yöntemler denetimsiz öğrenme olarak adlandırılır. Denetimli öğrenmede, girdiden çıktıya doğru değerler bir denetleyen tarafından sağlanmaktadır. Denetimsiz öğrenmede sadece girdi verileri vardır denetimli öğrenmedeki gibi bir denetçi yoktur. Denetimsiz öğrenmede amaç, girdideki düzenleri bulmaktır. Girdi örneklerinde belirli modellerin diğerlerinden daha sık meydana geldiği bir yapı vardır ve bu yapı öğrenilmeye çalışılır (Kubat, 2017, Alpaydın, 2004). Denetimsiz öğrenmede kullanılan en yaygın yöntemler kümeleme ve boyutsal küçültme yöntemleridir. Tez kapsamında bu yöntemler kullanılmadığı için detaylı olarak anlatılmamıştır.

3.3. Pekiştirmeli Öğrenme

Bazı uygulamalarda, sistemin çıktısı bir dizi olaydır. Böyle bir durumda önemli olan tek bir olması değil, hedefe ulaşmak için doğru olayların sırası olan politikadır. Bu tür öğrenme yöntemlerine pekiştirmeli öğrenme denilmektedir. Pekiştirmeli öğrenme oyun, finans sektörü, imalat, stok yönetimi ve robot navigasyonu gibi alanlarda yaygın olarak kullanılmaktadır. Bu alanlarda kullanılan pekiştirmeli öğrenme yöntemleri Q-Öğrenme ve derin Q-Öğrenme'dir. Tez kapsamında kullanılmadığı için bu yöntemler detaylı olarak anlatılmamıştır.

4. DERİN ÖĞRENME

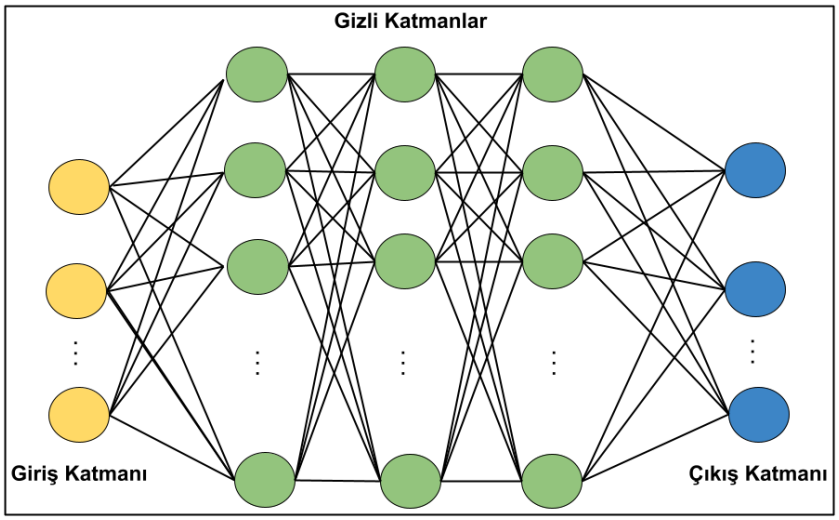
Derin öğrenme, insan beyninde bulunan nöronların yapısından esinlenen yapay sinir ağlarını kullanan makine öğrenimi metodolojilerinin belirli bir alt kümesidir. Derin öğrenmedeki derin ifadesi, birbirini izleyen temsil katmanlarına sahiptir. Modelin derinliği yapay sinir ağı modelindeki gizli katman sayısını ifade etmektedir (Verdhan, 2021).

Yapay sinir ağı, insan beyninin yapısından esinlenerek geliştirilmiştir. Sinir hücreleri nöronlar aracılığıyla birbirine bağlıdır. Bir nöron, bir veya daha fazla girdi değeri alan ve tek bir sayısal değer çıkaran matematiksel bir denklemden oluşmaktadır. Bu denklem Eşitlik (4.1)'de gösterilmiştir.

$$y=f\left(\sum_i x_i w_i + b\right) \tag{4.1}$$

Eşitlik (4.1)'de x_i giriş vektörü, w_i girdilere karşılık gelen ağırlık değerlerini, b bias değerini ve f ise aktivasyon fonksiyonunu temsil etmektedir. Bias değeri aktivasyon fonksiyonunun sağa ve sola ötelenmesi işlemini gerçekleştirir (Vasilev ve diğ., 2019).

Tipik bir derin sinir ağı giriş katmanı, çıkış katmanı ve gizli katmanlar olmak üzere 3 katmandan oluşmaktadır. Giriş katmanı, verilere ait özniteliklerin ağa giriş olarak verildiği katmandır. Giriş katmanında öznitelik sayısı kadar nöron bulunmaktadır. Gizli katmanda, işlenmemiş veri giriş katmanından alınıp işlenerek, çıktıyı verecek olan çıkış katmanına gönderilir. Çıkış katmanı ise nihai sonucun üretildiği katmandır. Sinir ağında katmanlar arasındaki düğümlerin birbirine bağlanması, ileri beslemeli ve tekrarlayan sinir ağı olmak üzere iki temel sınıfa ayrılmaktadır. İleri beslemeli sinir ağında, girdilerden çıktılara bilgi hareketi sadece tek yöndedir. Tekrarlayan sinir ağında ise bilgilerin bir kısmı ters yönde hareket etmektedir (Imran ve Alsuhaibani, 2019). Basit bir derin sinir ağının yapısı Şekil 4.1'de gösterilmiştir.



Şekil 4.1. Basit derin sinir ağı yapısı

4.1. Aktivasyon Fonksiyonları

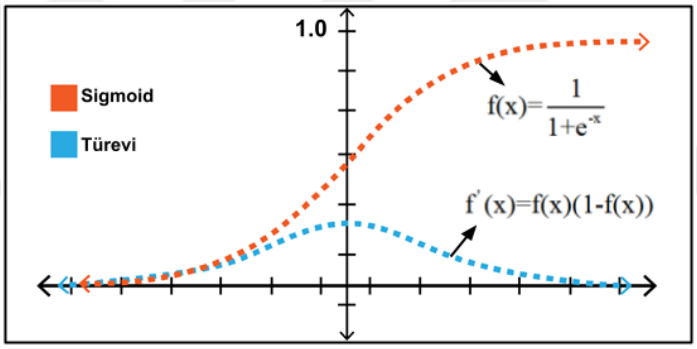
Aktivasyon Fonksiyonları, sinir ağlarında, bir giriş sinyalini, yığındaki bir sonraki katmana giriş olarak beslenen bir çıkış sinyaline dönüştürmek için kullanılan özel fonksiyonlardır. Bir sinir ağının tahmin doğruluğu, kullanılan katman sayısından çok kullanılan aktivasyon fonksiyonunun türüne göre belirlenmektedir. Bir sinir ağına, bir aktivasyon işlevi uygulanmadığı zaman, tahmin edilen çıktının sağlanan girdi ile aynı olduğu doğrusal bir regresyon modeli gibi çalışmaktadır. Bu yüzden, modelin birden çok gizli katmana sahip olduğu karmaşık, yüksek boyutlu ve doğrusal olmayan veri kümelerine sahip yapay sinir ağlarında ve derin öğrenmede aktivasyon fonksiyonları kullanılması gerekmektedir. Aktivasyon fonksiyonları doğrusal ve doğrusal olmayan fonksiyonlar olmak üzere ikiye ayrılmaktadır. Doğrusal bir aktivasyon fonksiyonunun sınırı doğrusaldır. Doğrusal fonksiyonlar kullanıldığında ağ yalnızca girdinin doğrusal değişikliklerine uyum sağlamaktadır. Fakat gerçek hayatta hatalar doğrusal olmayan özelliklere sahiptir. Böyle durumlarda doğrusal fonksiyonlar, sinir ağları hakkında sınırlı bilgi edinme becerisine sahip olmaktadır. Bu nedenle, sinir ağlarında doğrusal olmayan aktivasyon fonksiyonları, doğrusal aktivasyon fonksiyonlarına göre daha yaygın kullanılmaktadır. Aktivasyon fonksiyonlarında, eşik bazlı sınıflandırıcı olması oldukça önem arz etmektedir. Eşik bazlı sınıflandırıcı, doğrusal dönüşümün değerinin nöronu aktive etmesi gerekip gerekmediği veya aktivasyon fonksiyonunun girdisi daha büyükse bir nöronun aktive edilmesi işlemini gerçekleştirir. Eşik değeri olmaması durumunda çıktı değeri, bir sonraki katmana girdi olarak beslenemez (Sharma ve diğ.,

2020). Doğrusal olmama ve eşik bazlı sınıflandırma özellikleri dışında, aktivasyon fonksiyonlarında türevlenebilir olma ve orijin noktasında kendine yakınsama gibi özelliklerin bulunması diğerlerine göre daha avantajlı olduğu anlamına gelmektedir (Bircanoğlu ve Arıca, 2018).

Derin sinir ağlarında en çok kullanılan aktivasyon fonksiyonları sigmoid, hiperbolik tanjant, ReLU, Leaky ReLU ve softmax fonksiyonlarıdır.

Sigmoid, kullanılan en eski aktivasyon fonksiyonlarından biridir. Sigmoid fonksiyonu verileri (0,1) arasındaki aralığa dönüştürür. Matematiksel olarak sigmoid fonksiyonu Eşitlik (4.2)'deki gibi ifade edilir (Rao ve McMahan, 2019). Sigmoid fonksiyonunun grafiği Şekil 4.2'de gösterilmiştir.

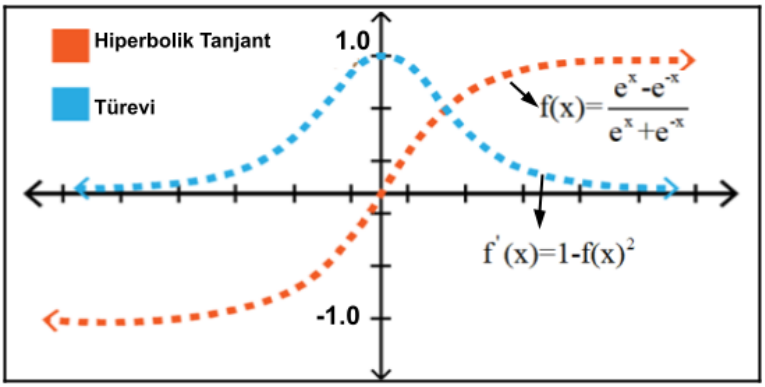
$$f(x)=\frac{1}{1+e^{-x}} \tag{4.2}$$



Şekil 4.2. Sigmoid fonksiyonu grafiği (URL-6, 2021)

Hiperbolik tanjant fonksiyonu sigmoid fonksiyonuna benzer bir yapıya sahiptir. Sigmoid fonksiyonundan farklı olarak hiperbolik tanjant fonksiyonu (-1,1) aralığında değer almaktadır. Ayrıca sigmoid fonksiyonuna göre türevinin daha dik olması yani daha çok değer alabilmesi, hızlı öğrenme ve daha iyi sınıflandırma yapabildiği anlamına gelmektedir (Kızrak, 2021). Matematiksel olarak hiperbolik tanjant fonksiyonu Eşitlik (4.3)'deki gibi ifade edilir. Hiperbolik tanjant fonksiyonunun grafiği Şekil 4.3'te gösterilmiştir.

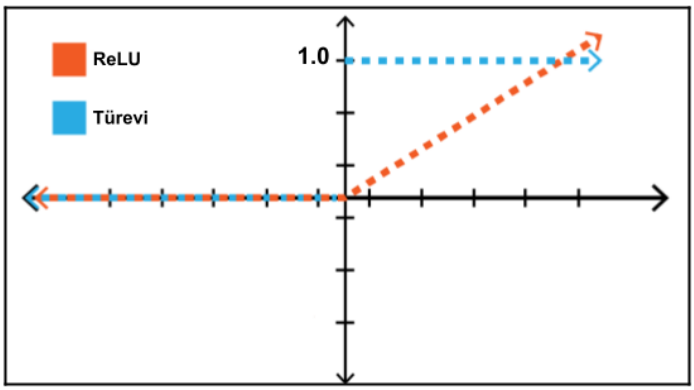
$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{4.3}$$



Şekil 4.3. Hiperbolik tanjant fonksiyonu grafiği (URL-6, 2021)

ReLU, sinir ağında yaygın olarak kullanılan doğrusal olmayan bir aktivasyon fonksiyonlarından biridir. ReLU fonksiyonunun kullanılmasının en büyük avantajı, tüm nöronların aynı anda etkinleştirilmemesidir. Böylece, bir nöron doğrusal dönüşümünün çıktısı sıfır olduğunda devre dışı bırakılabilir (Sharma ve diğ., 2020). Matematiksel olarak ReLU fonksiyonu Eşitlik (4.4)'teki gibi ifade edilir. ReLU fonksiyonunun grafiği Şekil 4.4'te gösterilmiştir.

$$f(x)=\begin{cases} 0, & x < 0 \text{ ise} \\ x, & x \geq 0 \text{ ise} \end{cases} \tag{4.4}$$

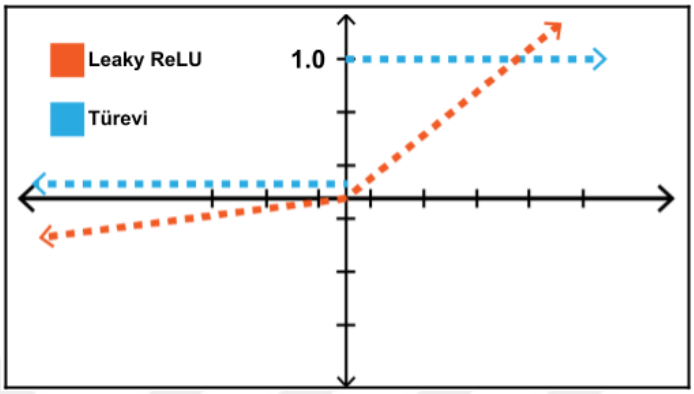


Şekil 4.4. ReLU fonksiyonu grafiği (URL-6, 2021)

Şekil 4.4'te görüldüğü gibi ReLU aktivasyon fonksiyonu negatif eksende sıfır değerini almaktadır. Negatif eksende sıfır olması türev değerinin de sıfır olacağı anlamına gelmektedir. Bu durum negatif eksende öğrenme işlemi gerçekleşmeyeceğini göstermektedir. ReLU fonksiyonunun bu dezavantajını ortadan kaldırmak için Leaky ReLU aktivasyon fonksiyonu önerilmiştir. Leaky ReLU fonksiyonunda Eşitlik (4.5)'te görüldüğü gibi, a değeri öğrenilmiş sızıntı parametresi olarak tanımlanmıştır. Negatif

düzlemde öğrenme bu parametre sayesinde sağlanmış olur (URL-6). ReLU fonksiyonunun grafiği Şekil 4.5'te gösterilmiştir.

$$f(x)=\begin{cases} ax, & x < 0 \text{ ise} \\ x, & x \geq 0 \text{ ise} \end{cases} \tag{4.5}$$



Şekil 4.5. Leaky ReLU fonksiyonu grafiği (URL-6, 2021)

Bir başka aktivasyon fonksiyonu olan softmax'ta sigmoid fonksiyonunda olduğu gibi, her birimin çıktısı 0 ile 1 arasında sıkıştırılır. Softmax fonksiyonunda her bir çıktı tüm çıktıların toplamına bölünür ve böylece m olası sınıf üzerinden ayrı bir olasılık dağılımı verilir (Rao ve McMahan, 2019). Matematiksel olarak softmax fonksiyonu Eşitlik (4.5)'deki denklemle ifade edilir.

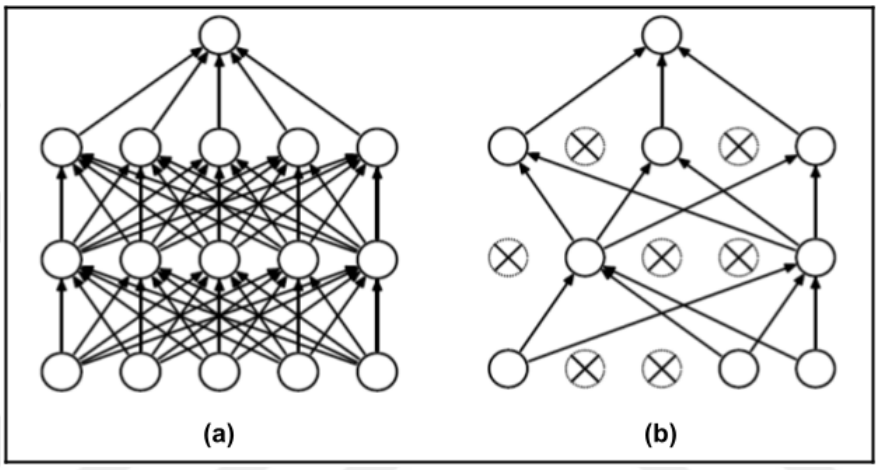
$$f(x_i)=\frac{e^{x_i}}{\sum_{k=1}^{m} e^{x_k}} \tag{4.6}$$

4.2. Derin Öğrenmede Kullanılan Teknikler

Eğitim süresini azaltmak ve modeli optimize etmek için derin öğrenme algoritmalarına uygulanabilecek birçok teknik vardır. Bu tekniklerden bazıları seyreltme, yığın normalleştirme ve optimizasyon algoritmasının belirlenmesidir.

Derin sinir ağları, girdileri ve çıktıları arasındaki çok karmaşık ilişkileri öğrenebilen modeller haline getirmeyi amaçlar. Bununla birlikte sınırlı eğitim verileriyle bu karmaşık ilişkilerin çoğu gürültü örneklemesinin sonucu olabilmektedir. Bu nedenle eğitim setinde başarılı olan modeller daha önce varlığından haberdar olunamayan gerçek test verilerinde başarılı olmayabilmektedir. Bu duruma aşırı uydurma

denilmektedir. Aşırı uydurmayı azaltmak için birçok yöntem geliştirilmiştir. Bu yöntemlerden en çok tercih edileni seyreltme yöntemidir. Seyreltme bir sinir ağındaki birimleri (gizli ve görünür) bırakmayı ifade etmektedir. Şekil 4.6'da gösterildiği gibi tüm gelen ve giden bağlantılarıyla birlikte birimi ağdan geçici olarak çıkarmak anlamına gelir. Örneğin seyreltme değeri 0,5 olarak belirlendiğinde bir sonraki katmandaki birim sayısı yarıya düşürülmüş olur. Seyreltme yönteminde hangi birimlerin bırakılacağının seçimi ise rastgele gerçekleşmektedir (Srivastava ve diğ., 2014).



Şekil 4.6. Derin sinir ağlarında seyreltme yönteminin uygulanması (a) seyreltme yöntemi uygulanmadan önce, (b) seyreltme yöntemi uygulandıktan sonra (Srivastava ve diğ., 2014)

Yığın Normalleştirme, derin sinir ağlarında daha hızlı ve daha kararlı eğitim yapılmasını sağlayan yaygın olarak kullanılan tekniklerden biridir. Derin sinir ağlarında ağırlık değerleri, her iterasyonda geriye yayılım işlemi ile ağ üzerinde geriye dönük olarak gradyan hesaplaması yapılarak güncellenmektedir. Güncelleme işleminde veri sayısının fazla olması, hesaplama işleminin de o oranda fazla süreceği anlamına gelmektedir. Bu durum eğitim aşamasında zaman ve bellek maliyeti açısından olumsuz sonuçlar oluşturmaktadır. Bu problemi çözmek için, veri seti küçük gruplara bölünüp, öğrenme işleminin seçilen bu küçük gruplar üzerinde yapılması gerçekleştirilebilir. Bu işleme yığın normalleştirme denilmektedir. Yığın normalleştirmede mini-yığın parametresi olarak belirlenen değer, modelin aynı anda kaç veriyi işleyeceği anlamına gelmektedir (URL-6). Bir mini grup üzerindeki x değerleri; $B=\{x_1, x_2, \ldots, m\}$, öğrenilecek parametreler; γ, β ve çıktı değeri y olmak

üzere, yığın normalleştirme yöntemine ait işlem adımları aşağıda maddeler halinde sıralanmıştır (Ioffe ve Szegedy, 2015):

1) Eşitlik (4.6) kullanılarak mini yığındaki x değerlerinin ortalaması alınır.
2) Eşitlik (4.7) kullanılarak mini-yığına ait varyans değeri hesaplanır.
3) Eşitlik (4.8) kullanılarak x değerleri normalize edilir.
4) Eşitlik (4.9) kullanılarak normalize edilmiş veri γ ve β parametreleri kullanılarak ölçeklendirilip kaydırılır ve bu değer (y), diğer sinir ağı katmanlarına aktarılır.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4.7}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{4.8}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{4.9}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{4.10}$$

Derin öğrenmede eğitim hızı ve modellerin son tahmin performansını belirlemek için optimizasyon algoritmaları kullanılmaktadır. Derin öğrenmede optimize ediciler, kayıpları azaltmak için sinir ağının ağırlık ve öğrenme hızı gibi özelliklerini değiştirmek için kullanılan algoritmalar veya yöntemlerdir. Optimizasyon algoritmaları, kayıpları azaltmak ve mümkün olan en doğru sonuçları sağlamaktan sorumludur. Kullanılan optimizasyon algoritmaları minimum noktaya adım adım ilerleyen gradyan alçalma (1. Türev) tabanlı yöntemlerdir. Adım miktarının (öğrenme katsayısı) büyük seçilmesinin minimum noktaya ulaşamama, küçük seçilmesinin ise minimum noktaya zaman açısından geç ulaşma dezavantajlarını oluşturabilir. Bu yüzden optimizasyon algoritması seçimi derin öğrenmede önemli aşamalardan biridir. Uyarlanabilir gradyan yöntemleri, derin öğrenmede yaygın olarak kullanılan optimizasyon yöntemlerinden biridir. Uzun yıllar boyunca tercih edilen stokastik gradyan iniş algoritması olması, derin sinir ağlarını eğitirken büyük ölçekli veri kümeleri için kötü koşullandırma ve zaman gerekliliği gibi ciddi sorunların üstesinden gelmekte zorluk çekebilmektedir. Ayrıca stokastik gradyan iniş algoritmasında öğrenme hızının manuel olarak ayarlanması gerekmektedir ve paralelleştirilmesi zordur. Bu sorunlar, optimizasyon algoritmalarının daha gelişmiş versiyonlarının çıkmasına sebep oldu. Son zamanlarda derin öğrenme için kullanılan optimizasyon

algoritmaları, eğitim sırasında öğrenme oranlarını uyarlamaktadır (Soydaner, 2020). Bu algoritmaların en yaygın kullanılanları aşağıda maddeler halinde listelenmiştir (Seyyarer ve diğ., 2019, Saleem ve diğ., 2020):

- SGD: Hızlı yakınsama yeteneğine sahip en basit derin öğrenme iyileştiricilerinden biridir. Tüm parametreler için statik bir öğrenme hızı, tüm eğitim süresi boyunca geçerlidir.
- Adagrad: Adagrad optimizasyon algoritmasında, modeldeki her parametre için farklı öğrenme oranları kullanılmaktadır. Öğrenme oranı, her parametrenin güncelleme sıklığına göre güncellenmektedir.
- RMSProp: Adagrad'daki eğitim süresini azaltmak için, öğrenme hızının üssel olarak azaltılmasını sağlayan algoritmadır.
- Adadelta: Adagrad optimizasyon algoritmasının daha gelişmiş versiyonu olan Adadelta'da, önceki gradyanlar sabit bir zaman aralığında biriktirilmektedir. Böylece birçok yinelemeden sonra bile öğrenmenin devam etmesi sağlanır. Adadelta algoritmasında, negatif gradyanlarda güncelleme yönünü sağlamak için Hessian yaklaşımı kullanılmaktadır.
- Adam: Adam algoritmasında Adagrad ve RMSProp algoritmalarının avantajlarını birleştiren bir yaklaşım benimsenmiştir. Bu algoritmada öğrenmeyi hızlandırmak için önceki gradyanlar kullanılmaktadır.
- Adamax: Sonsuzluk normuna dayanan Adamax algoritması, seyrek parametre güncellemelerinde Adam algoritmasından daha iyi sonuçlar vermektedir.

4.3. Derin Öğrenme Yöntemleri

Kullanım amaçlarına göre farklı derin öğrenme yöntemleri bulunmaktadır. Bu yöntemlerden en popüler olanları CNN, RNN, DBN ve Deep Feedforward Neural Network (DFNN) algoritmalarıdır. Tez kapsamında DFNN algoritması kullanılmıştır. Yöntemin detaylı açıklaması aşağıda verilmiştir.

Derin sinir ağlarında nöronlar arasındaki bağlantıların türüne bağlı olarak, ileri beslemeli ve tekrarlayan sinir ağları olmak üzere iki ana ağ mimarisi kategorisi vardır. Nöronların çıktılarından ağ boyunca girişlere doğru geri besleme yoksa, o ağa ileri beslemeli sinir ağı denilmektedir. Ağlarda geri besleme varsa, başka bir deyişle

çıktılardan girişlere doğru sinaptik bir bağlantı olduğu durumlarda, o ağa ise tekrarlayan sinir ağı denilmektedir. Derin öğrenmede yaygın olarak kullanılan DFNN'de bilgi, giriş katmanından çıktı katmanına kadar yalnızca tek yönde hareket ettiği için ileri beslemeli sinir ağlarından biridir (Sazlı, 2006).

DFNN'de, giriş ve çıkış katmanındaki nöronların sayısı probleme özgü olarak belirlenmektedir. Gizli katmanların sayısını ve bunların gizli nöronlarını belirlemek için herhangi bir kriter yoktur. Gizli katmanların sayısı DFNN'nin performansından sorumludur. Küçük boyutlu bir ağ, yetersiz uyuma neden olabilirken, büyük bir ağda ise aşırı uydurmaya sebep olabilmektedir. Bu problemin çözümü için DFNN'de, bir önceki bölümde bahsedilen optimizasyon teknikleri kullanılarak en düşük test hatasıyla optimum bir yapı bulma amaçlanmaktadır (Gupta ve Raza, 2020).

DFNN'de ağırlıkların eğitilmesi için geri yayılım algoritması kullanılmaktadır. Geri yayılım algoritmasında çıkış katmanındaki her bir nörona ait hata sinyallerinin karelerinin toplamıyla elde edilen bir uygunluk fonksiyonu tanımlanmıştır. Bu fonksiyon, giriş-çıkış nöronları arasındaki uyumun ne kadar sağlandığını ölçmeyi amaçlamaktadır. Uygunluk fonksiyonu (E), Eşitlik (4.10) kullanılarak hesaplanmaktadır. Eşitlik (4.10)'daki Eğitimin n. yinelemesinde çıkış katmanındaki j. nörona ait hata değeri (e_j) ise Eşitlik (4.11) kullanılarak hesaplanmaktadır (Sazlı, 2006).

$$E=\frac{1}{2}e_j^2(n) \tag{4.11}$$

$$e_j(n)=d_j-y_j(n) \tag{4.12}$$

Eşitlik (4.11)'deki d_j, j. nörona ait istenen çıktıyı, yj(n) ise, ağın mevcut ağırlıkları kullanılarak, n. yinelemede j. nöron için hesaplanan gerçek çıktıdır.

Geri yayılım algoritmasında, uygunluk fonksiyonu minimum yapılma amaçlanmaktadır. Uygunluk fonksiyonunun minimuma yaklaşılması için nöronların ağırlıkları algoritma ile her iterasyonda güncellenmektedir. Güncelleme işlemi Eşitlik (4.12) kullanılarak hesaplanmaktadır. Eşitlik (4.12)'de k, geri yayılım algoritmasındaki önceden belirlenmiş sabit bir değer olan öğrenme katsayısını temsil etmektedir (Karadeniz ve diğ., 2001).

$$\Delta w_{ij} = -k \frac{\partial E(\vec{w})}{w_{ij}} \tag{4.13}$$

Ağırlık güncelleme işleminden sonra n. yinelemede j. nöron için hesaplanan gerçek çıktı değerinin nihai hali Eşitlik (4.13) ile ifade edilmektedir.

$$y_j(n) = f\left(\sum_{i=0}^{m} w_{ij}(n) y_i(n)\right) \tag{4.14}$$

Eşitlik (4.13)'te f aktivasyon fonksiyonunu, m bir önceki katmandan j nöronuna yapılan toplam girdi sayısını temsil etmektedir (Sazlı, 2006).

5. ÖNERİLEN YÖNTEMLER

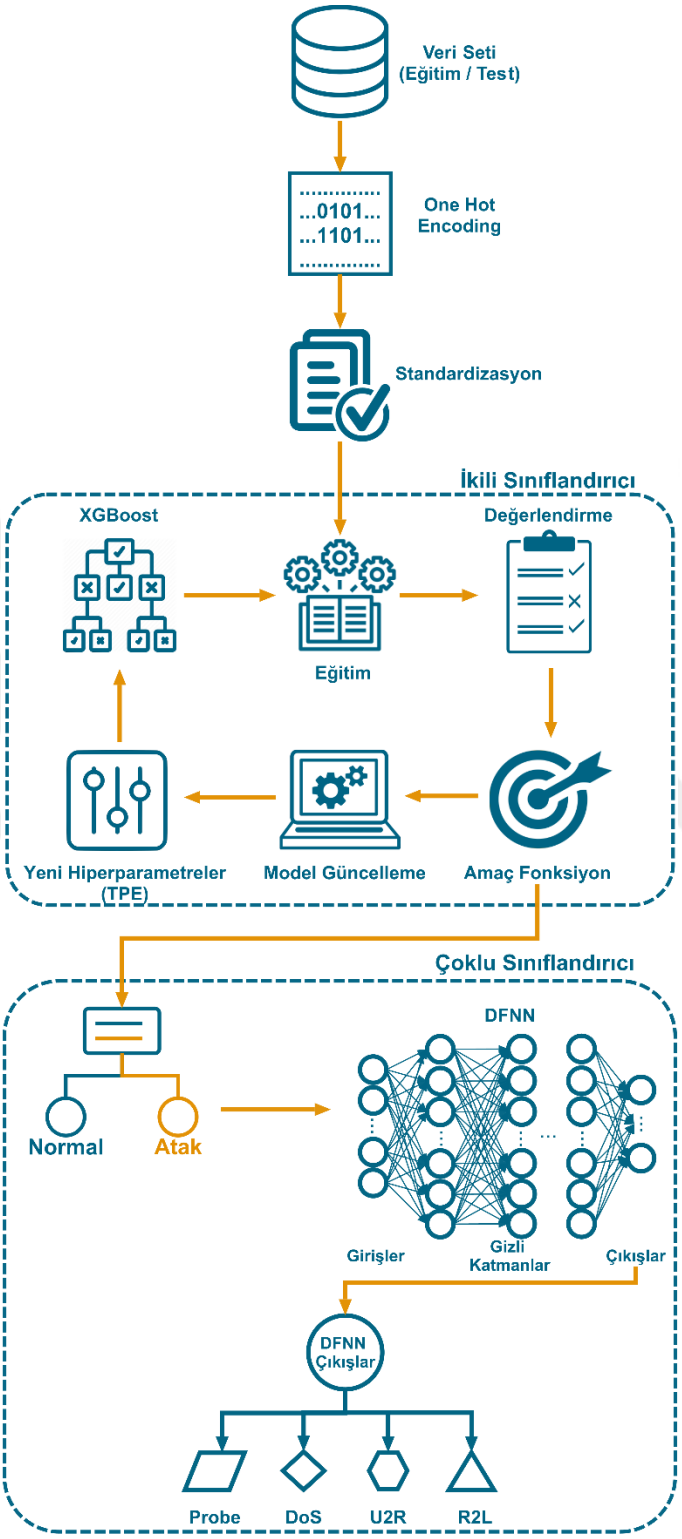
Tez kapsamında Probe, DoS, U2R ve R2L saldırılarının tespiti ve DNS tünelleme saldırılarının engellenmesi için iki ayrı yöntem önerilmiştir. Bu yöntemler alt bölümlerde detaylı olarak anlatılmıştır.

5.1. Probe, DoS, U2R ve R2L Saldırılarının Tespiti

Probe, DoS, U2R ve R2L saldırıları için tasarlanan saldırı tespit modelinin genel mimarisi Şekil 5.1'de gösterilmiştir. Önerilen model temelde iki aşamalıdır. İlk aşamada bir makine öğrenmesi yöntemi olan XGBoost yöntemi kullanılarak verinin ikili olarak sınıflandırılması işlemi gerçekleştirilmiştir. Daha sonra ikili olarak sınıflandırılmış olan veriler, atak türlerinin tespiti amacıyla bir derin öğrenme modeli kullanılarak ikinci kez sınıflandırılmıştır. Önerilen modelin ayrıntıları aşağıda maddeler halinde açıklanmıştır.

1) İlk olarak veri seti içerisindeki kategorik değerlerin one hot encoding yöntemi kullanılarak sayısallaştırılması ve tüm veri setinin standardizasyon işlemi gerçekleştirilmiştir.
2) İkinci aşamada veri setinin ikili olarak sınıflandırılması amacıyla XGBoost algoritması kullanılmıştır. XGBoost algoritması içerisinde üç farklı karar ağacı içermektedir. Algoritmanın probleme özgü olarak en iyi sonucu üretebilmesi için hem doğru karar ağacı modelinin hem de modele özgü parametrelerin (hiperparametreler) tespit edilmesi gerekmektedir. Bu aşamada hiperparametre seçimi için TPE algoritması kullanılmıştır. Ağaç yapısı olarak ise Dart modeli tercih edilmiştir. Model, seçilen parametreler ve ağaç yapısı kullanılarak eğitim veri seti ile eğitilmiştir. Oluşan model ile verilerin atak-normal olarak ikili sınıflandırılması gerçekleştirilmiştir.
3) Üçüncü aşamada XGBoost algoritmasının atak olarak sınıflandırdığı veri üzerinde DFNN modeli uygulanmıştır. DFNN modeline yalnızca atak olarak sınıflandırılan verilerin gönderilmesiyle veri setine herhangi bir müdahale yapılmadan, yöntemin uygun verilerle eğitilmesi sağlanmış ve yüksek başarımlı bir model elde edilmiştir.

4) Son olarak geliştirilen hibrit model, iki ayrı test veri setleri üzerinde doğruluğu test edilmiştir.



Şekil 5.1. Probe, DoS, U2R ve R2L saldırıları için tasarlanan STS mimarisi

5.1.1. Kullanılan veri seti

Probe, DoS, U2R ve R2L saldırılarının tespiti için NSL-Knowledge Discovery and Data mining (NSL-KDD) veri seti kullanılmıştır. Hem eğitim hem test aşamasında verilerdeki kayıtlar, herhangi bir değişikliğe tabi tutulmadan orijinal haliyle kullanılmıştır. NSL-KDD veri seti, KDDCup99 veri setinden üretilmiştir. KDDCup 99 veri setindeki benzer kayıtların çokluğu nedeniyle saldırı tespitinde kullanılan algoritmaların özellikle sık kayıtlarda ön yargılı davranmasına sebep olmaktadır. Bu problemin çözümü için (Tavallaee ve diğ., 2009), KDDCup 99 veri setinin daha gelişmiş versiyonu olan NSL-KDD veri setini önermişlerdir. NSL-KDD, STS'lerin doğruluğunun test edilmesi için kullanılan en yaygın veri setlerinden biridir (Dhanabal ve Shantharajah, 2015). Bu veri seti KDDTrain, KDDTest+ ve KDDTest-21 olmak üzere 3 kayıt türünden oluşmaktadır. KDDTrain eğitim için kullanılan ve 125973 kayıttan oluşan veri setidir. KDDTest+ ve KDDTest-21 test veri setleridir ve sırasıyla 22544 ve 11850 kayıttan oluşmaktadır. NSL-KDD veri seti Normal, Probe, DoS, U2R ve R2L olmak üzere 5 sınıftan oluşmaktadır. Normal dışındaki diğer 4 sınıfta 39 farklı saldırı türü mevcut olup, bunların 22'si eğitim veri setinde, 17'si ise test veri setinde bulunmaktadır (Hu ve diğ., 2020). Saldırı türlerinin sınıflara göre dağılımı Tablo 5.1'de, veri setindeki kayıt türlerinin dağılımı ise Tablo 5.2'de verilmiştir.

Tablo 5.1. NSL-KDD veri setindeki saldırı türlerinin sınıflara göre dağılımı

Saldırı Sınıfı	Saldırı Türü
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named

Tablo 5.2. NSL-KDD veri setindeki kayıt türlerinin dağılımı

Veri Tipi	KDDTrain	KDDTest+	KDDTest-21
Normal	67343	9711	2152
Probe	45927	2421	2402
DoS	11656	7458	4342
U2R	52	200	200
R2L	995	2754	2754
Toplam	125973	22544	11850

NSL-KDD veri setinde bulunan özelliklerin listesi Tablo 5.3'te gösterilmiştir. Veri setinde yer alan 41 özelliğin 38'i sayısal değer alırken, 3 özellik (protocol type, service ve flag) sayısal olmayan değer almaktadır.

Tablo 5.3. NSL-KDD veri setindeki özelliklerin listesi (Dhanabal ve Shantharajah, 2015)

No	Özellik Adı	Tanım
1	Duration	Bağlantının süresinin uzunluğu
2	Protocol type	Bağlantıda kullanılan protokol
3	Service	Kullanılan hedef ağ hizmeti
4	Flag	Bağlantının durumu - normal veya hata
5	Source bytes	Tek bağlantıda kaynaktan hedefe aktarılan veri bayt sayısı
6	Destination bytes	Tek bağlantıda hedeften kaynağa aktarılan veri bayt sayısı
7	Land	Kaynak ve hedef IP adresleri ve bağlantı noktası numaraları eşitse 1, değilse 0 değerini alır.
8	Wrong fragment	Bağlantıdaki toplam yanlış parça sayısı
9	Urgent	Bağlantıdaki acil bitin etkinleştirildiği paketlerin sayısı
10	Hot	İçerikteki 'sıcak' göstergelerinin sayısı, örneğin: bir sisteme girme.
11	Number failed logins	Başarısız giriş denemelerinin sayısı
12	Logged in	Oturum Açma Durumu: oturum açıldysa 1, açılmadıysa 0 değerini alır.
13	Num compromised	İhlal yapılan koşulların sayısı
14	Root shell	Root shell elde edilirse 1, değilse 0 alır.
15	Su attempted	'Su root' komutu denendiyse veya kullanıldıysa 1, aksi halde 0 değerini alır.
16	Num root	Root olarak gerçekleştirilen işlemlerin sayısı
17	Num file creations	Bağlantıdaki dosya oluşturma işlemlerinin sayısı
18	Num shells	Shell bilgi istemlerinin sayısı
19	Num access files	Erişim kontrol dosyalarındaki işlem sayısı
20	Num outbound cmds	Ftp oturumunda giden komutların sayısı
21	Is host login	Root veya yönetici listesinden giriş yapıldıysa 1, aksi halde 0 değerini alır.
22	Is guest login	Misafir girişi yapıldıysa 1, aksi halde 0 değerini alır.
23	Count	Son iki 2 saniyede mevcut bağlantıyla aynı hedef ana bilgisayara yapılan bağlantı sayısı
24	Srv count	Son iki saniyede mevcut bağlantıyla aynı bağlantı noktası numarasına yapılan bağlantı sayısı

Tablo 5.3. (Devam) NSL-KDD veri setindeki özelliklerin listesi (Dhanabal ve Shantharajah, 2015)

25	Serror rate	Count içinde toplanan bağlantılar arasında s0, s1, s2 veya s3 bayrağını etkinleştiren bağlantıların yüzdesi
26	Srv serror rate	Srv count içinde toplanan bağlantılar arasında s0, s1, s2 veya s3 bayrağını etkinleştiren bağlantıların yüzdesi
27	Rerror rate	Count içinde toplanan bağlantılar arasında REJ bayrağını etkinleştiren bağlantıların yüzdesi
28	Srv rerror rate	Srv count içinde toplanan bağlantılar arasında REJ bayrağını etkinleştiren bağlantıların yüzdesi
29	Same srv rate	Count içinde toplanan bağlantılar arasında aynı hizmete yapılan bağlantıların yüzdesi
30	Diff srv rate	Count içinde toplanan bağlantılar arasında farklı hizmetlere yapılan bağlantıların yüzdesi
31	Srv diff host rate	Srv count içinde toplanan bağlantılar arasında farklı hedef makinelere yapılan bağlantıların yüzdesi
32	Dst host count	Aynı hedef ana bilgisayar IP adresine sahip bağlantı sayısı
33	Dst host srv count	Aynı port numarasına sahip bağlantı sayısı
34	Dst host same srv rate	Dst host count içinde toplanan bağlantılar arasında aynı hizmete yapılan bağlantıların yüzdesi
35	Dst host diff srv rate	Dst host count içinde toplanan bağlantılar arasında farklı hizmetlere yapılan bağlantıların yüzdesi
36	Dst host same src port rate	Dst host srv count içinde toplanan bağlantılar arasında aynı kaynak bağlantı noktasına yapılan bağlantıların yüzdesi
37	Dst host srv diff host rate	Dst host srv count içinde toplanan bağlantılar arasında farklı hedef makinelere yapılan bağlantıların yüzdesi
38	Dst host serror rate	Dst host count içinde toplanan bağlantılar arasında s0, s1, s2 veya s3 bayrağını etkinleştiren bağlantıların yüzdesi
39	Dst host srv serror rate	Dst host srv count içinde toplanan bağlantılar arasında s0, s1, s2 veya s3 bayrağını etkinleştiren bağlantıların yüzdesi
40	Dst host rerror rate	Dst host count içinde toplanan bağlantılar arasında REJ bayrağını etkinleştiren bağlantıların yüzdesi
41	Dst host srv rerror rate	Dst host srv count içinde toplanan bağlantılar arasında REJ bayrağını etkinleştiren bağlantıların yüzdesi

5.1.2. Veri seti ön işleme

NSL-KDD veri setinde protocol type, service ve flag olmak üzere üç tane sayısal olmayan özellik bulunmaktadır. Bu özelliklerin sayısal hale dönüştürülmesi için one-hot encoding işlemi yapılmıştır. Örnek olarak protocol type özelliğinde yer alan TCP, UDP, ICMP kategorileri üç boyutlu ikili vektörler haline dönüştürülerek sırasıyla (1, 0, 0), (0, 1, 0), (0, 0, 1) değerlerini almıştır. Böylece protocol type özelliği 3, service özelliği 64 ve flag özelliği 11 özellik ile temsil edilmiştir. Sonuç olarak 41 özelliğe sahip veri seti toplamda 116 özelliğe sahip veri setine dönüşmüştür. Veri setinde yer alan label özelliği ise ikili kodlama yapılarak normal ve saldırı olmak üzere iki tip olarak sınıflandırılmıştır.

Makine öğrenmesi teknikleri uygulanmadan önce kullanılacak verinin belirli bir ölçeğe uygun hale getirilmesi gerekmektedir. Özellikle sürekli değerler alan özelliklerin minimum ve maksimum değer aralıklarında ciddi farklılıklar olabilmektedir. Örneğin çalışma kapsamında kullanılan NSL-KDD veri setinde yer alan land özelliği [0-1] aralığında değer alırken, source bytes özelliği [0-1.37x109] aralığında değer almaktadır. Bu durum yüksek değerlere sahip özelliklerin karar verme aşamasında modeldeki ağırlığının daha fazla olmasına sebep olmaktadır. Veriler üzerinde uygulanacak normalizasyon ya da standardizasyon işlemleri ile bu sorun çözülebilmektedir. Her iki teknikte verinin yeniden ölçeklendirilmesini sağlarken, normalizasyon işlemi aykırı değerlerin elimine edilmesine sebep olmaktadır. Bu nedenle çalışmamızda en yaygın standardizasyon yöntemlerinden biri olan Z-score yöntemi kullanılmıştır. Yöntem Eşitlik (5.1)'de gösterildiği gibi formülize edilmektedir.

$$z = \frac{x - \mu}{\sigma} \tag{5.1}$$

Eşitlik (5.1)'de yer alan z değeri, normalize edilmiş veriyi, x girdi değerini, m girdi setinin ortalamasını ve σ setinin standart sapmasını ifade etmektedir. Z-score yönteminde veri setindeki tüm özelliklerin ortalaması sıfıra, standart sapması ise bire eşitlenmektedir.

5.1.3. Hiperparametre optimizasyonu

Makine öğrenmesinde veriler üzerinden elde edilen parametrelere ek olarak, modeli tasarlarken belirlenebilecek model parametreleri de bulunmaktadır. Veri üzerinden belirlenemeyen bu parametreler hiperparametreler olarak adlandırılmaktadır (Tanyıldızı ve Demirtaş, 2019). Bir veri seti üzerinden ölçülen en yüksek başarımı verecek algoritmanın hiperparametrelerini bulmak için hiperparametre optimizasyonuna ihtiyaç vardır. Bu parametrelerin konfigürasyonunun iyi yapılması, modelin başarımını önemli ölçüde arttırmaktadır (Erwianda ve diğ., 2019). Bu nedenle önerilen yöntemde hiperparametre optimizasyonu tekniği olarak Bayes optimizasyonu modelleme yaklaşımlarından biri olan TPE yöntemi kullanılmıştır. TPE yöntemi Sequential Bayesian model-based optimization (SBMO) yöntemlerinden biridir. SMBO yöntemlerinde farklılık vekil model olan p (y | x)'in oluşturulma şeklidir. TPE'de p (y | x) değeri, Eşitlik (5.2)'de gösterildiği şekilde hesaplanmaktadır.

$$p(y|x)=\frac{p(x|y)p(y)}{p(x)} \tag{5.2}$$

Eşitlik (5.2)'deki, p (x | y); amaç fonksiyonunda puan verilen hiperparametrelerin olasılığını, x; önerilen hiperparametreler kümesini, y; x parametrelerini kullanan amaç fonksiyonunun değerini ve p (y | x) ise; x olarak verilen y'nin olasılığını ifade eden vekil olasılık modelini temsil etmektedir. p (x | y) değeri Eşitlik (5.3)'de gösterildiği gibi hesaplanmaktadır. Burada yer alan y *; amaç fonksiyonunun eşik değerini göstermektedir. Amaç fonksiyonunun değeri eşikten küçükse *l(x)*, büyükse *g(x)* fonksiyonuna atanır.

$$p(x|y)=\begin{cases} l(x), & y<y^{*}\text{ise} \\ g(x), & y \geq y^{*}\text{ise} \end{cases} \tag{5.3}$$

TPE'de, amaç fonksiyonunun her değerlendirmesinden sonra vekil olasılık modeli sürekli güncellenmektedir. Vekil modelden seçilen bir sonraki hiperparametre kümesine ait kriteri seçim fonksiyonu belirlemektedir. Bu kriterlerden en uygun olanı ise Eşitlik (5.4)'teki formülle ifade edilen beklenen iyileştirme (Bİ) metriği ile seçilmektedir. Yöntemin nihai amacı Bİ ile ifade edilen kriteri optimize etmektir.

$$Bİ_{y^{*}} = \int_{-\infty}^{y^{*}} (y^{*}-y)p(y|x)dy \tag{5.4}$$

Her adımda yinelenen Bİ, Bayes Kuralına göre nihai olarak Eşitlik (5.5)'teki formülle hesaplanmaktadır (Bergstra ve diğ., 2011).

$$B\dot{I}_{y^*}(x) = \frac{\gamma l y^* l(x) - l(x) \int_{-\infty}^{y^*} p(y)dy}{\gamma l(x) + (1-\gamma) g(x)} \infty (\gamma + \frac{g(x)}{l(x)}(1-\gamma))^{-1} \quad (5.5)$$

5.1.4. XGBoost algoritması

XGBoost algoritması içerisinde genel, güçlendirici ve öğrenme görevi parametreleri olmak üzere üç farklı tip parametreye sahiptir. Genel parametreler, algoritmanın genel olarak işlevini tanımlamak amacıyla kullanılan parametrelerdir. Güçlendirici parametreler, genel parametrelerin seçiminde belirlenen güçlendiricinin performansını belirleyen parametrelerdir. Öğrenme görevi parametreleri ise kullanılacak veriye uygun olarak amaç fonksiyonunun optimizasyonu, öğrenme ve değerlendirme süreçlerinde kullanılan parametrelerdir. Çalışma kapsamında probleme özgü olarak optimum sonucun elde edilebilmesi için hiperparametre ayarı gerçekleştirilmiştir. Bu amaçla TPE yöntemi kullanılmıştır. XGBoost algoritmasına ait elde edilen hiperparametreler Tablo 5.4'te gösterilmiştir. Tabloda yer alan ilk parametre genel kategorisine ait parametreyi, tablonun sonundaki iki parametre öğrenme görevi kategorisine ait parametreleri, diğer parametreler ise güçlendirici kategorisine ait optimum parametreleri göstermektedir.

Tablo 5.4. Hiperparametre tablosu

Parametre	Değer
booster	gbtree
lambda	10
alpha	0.1
max_depth	20
eta	0.3
gamma	0
grow_policy	depthwise
colsample_bylevel	1
min_child_weight	5
colsample_bytree	1
subsample	1
tree_method	exact
nround	10000
objective	binary:logistic
max_delta_step	0

Tablo 5.5'te XGBoost yönteminde veri setine ait özelliklerin her birinin çıkışa olan etkisi önem ve yüzde değerleri üzerinden gösterilmiştir. Tabloda algoritmanın başarısını en çok ve en az etkileyen dört özellik yer almaktadır. Önem değeri en önemli özellik 1, en düşük özellik 0 olacak şekilde hesaplanmaktadır. Örneğin src_bytes özelliğine ait önem değeri 1 olduğu için sınıf etiketini belirleme konusunda çok net bir etkiye sahip olduğu söylenebilir.

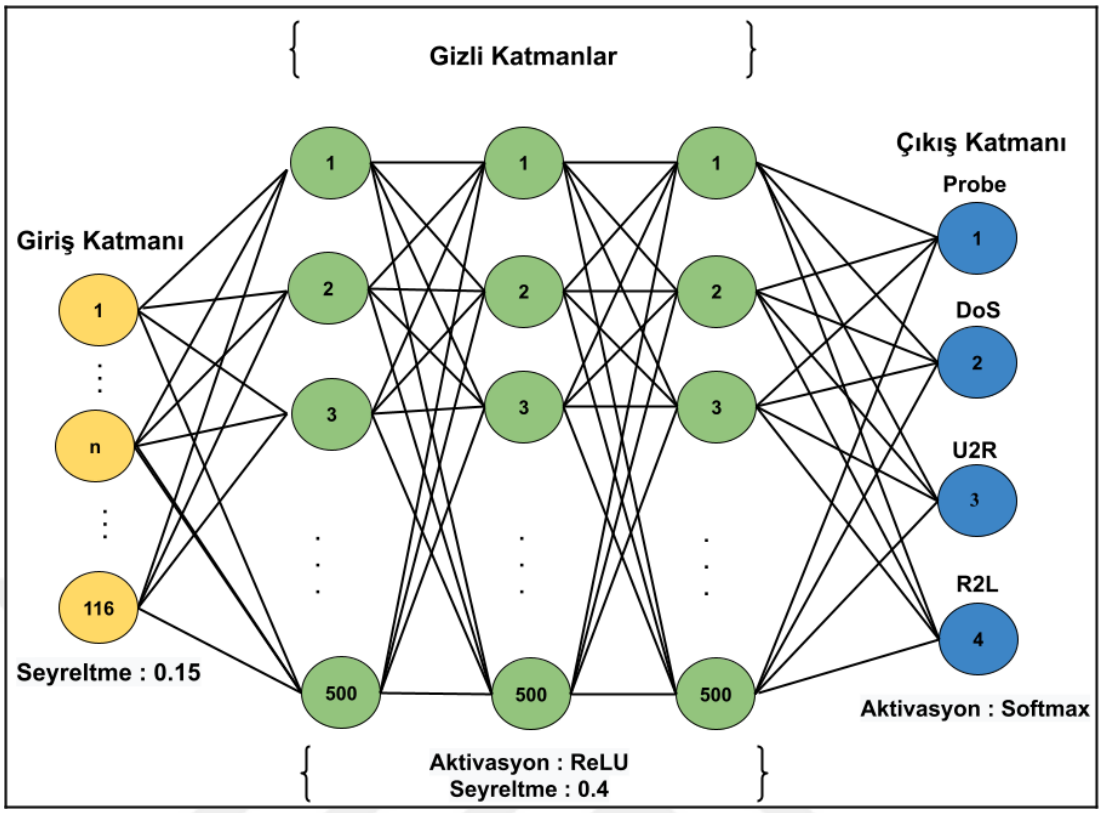
Tablo 5.5. XGBoost yönteminde veri setine ait özelliklerin çıkışa olan etkisi

Özellik adı	Önem
src_bytes	1.0
service.ecr_i	0.1044
dst_host_srv_diff_host_rate	0.1040
level	0.0858
...	
dst_bytes	0.0265
dst_host_serror_rate	0.0185
dst_host_same_srv_rate	0.0093
dst_host_srv_serror_rate	0.0085

5.1.5. DFNN yöntemi

Probe, DoS, U2R ve R2L saldırıları için önerilen DFNN yöntemine ait ağ yapısı Şekil 5.2'de gösterilmiştir. Probleme özgü olarak modellenen DFNN mimarisi 3 gizli katman içermektedir. Giriş katmanı 116 birim, gizli katmanlar 500'er birim ve çıkış katmanı 4 birimden oluşmaktadır. Aşırı uydurmadan kaçınmak için, giriş ve gizli katmanlarda seyreltme tekniği kullanılmıştır. Aktivasyon fonksiyonu olarak gizli katmanlarda ReLU, çıkış katmanında ise softmax kullanılmıştır.

Tablo 5.6'da DFNN mimarisinde giriş olarak kullanılan 116 özelliğin her birinin çıkışa olan etkisi önem ve yüzde değerleri üzerinden gösterilmiştir. Veri setinde yer alan 41 özellik one-hot encoding sonucu 116'ya çıktığı için tablodaki değerler de bu özellikler üzerinden hesaplanmıştır. Tabloda ağ yapısını en çok ve en az etkileyen yedi özellik yer almaktadır. Örneğin wrong_fragment özelliğine ait önem değeri 1 olduğu için sınıf etiketini belirleme konusunda çok net bir etkiye sahip olduğu söylenebilir.



Şekil 5.2. Probe, DoS, U2R ve R2L saldırıları için önerilen DFFN ağ yapısı

Tablo 5.6. DFNN mimarisinde veri setine ait özelliklerin çıkışa olan etkisi

Özellik adı	Önem
wrong_fragment	1.0
dst_host_diff_srv_rate	0.7331
service.eco_i	0.7228
flag.S0	0.7186
service.ecr_i	0.6297
service.ftp_data	0.6121
diff_srv_rate	0.6036
...	...
urgent	0.4877
flag.SF	0.4837
flag.RSTR	0.4826
su_attempted	0.4821
protocol_type.udp	0.4811
service.auth	0.4791
dst_host_rerror_rate	0.4775

5.2. DNS Tünelleme Tespiti ve Engellenmesi

Bu bölümde DNS trafiği üzerinden yapılan tünel saldırılarının gerçek zamanlı olarak engellenmesi için yapılan işlemler sırasıyla alt bölümlerde detaylı olarak açıklanmıştır.

5.2.1. Veri toplama

Legal DNS verisinin elde edilmesi için dünya genelinde en çok ziyaret edilen web siteleri (URL-7) kullanılmıştır. Bu sitelere erişim, yazılan bir web tarayıcı uygulaması üzerinden sağlanmış, wireshark ve tcpdump uygulamaları ile ağ dinlenilerek legal DNS verileri elde edilmiştir.

DNS tünel verisinin elde edilmesi için Iodine, Dns2cat, Dns2tcp DNS tünelleme araçları kullanılmıştır. Bu araçlar Ubuntu işletim sistemi üzerinde aynı bilgisayar ve aynı ağ üzerinde çalıştırılmıştır. Yine legal DNS verisindeki gibi Alexa top 1 milyon listesindeki sitelere, yazılan shell scripti ile istekte bulunulmuş, wireshark ve tcpdump uygulamaları ile tünelin bulunduğu ağ dinlenerek tünelli DNS verileri toplanmıştır. Legal ve tünel verilerin dağılımı Tablo 5.7'de verilmiştir.

Tablo 5.7. Toplanan legal ve tünel verisine ait dağılım

<table>
<tr><td rowspan="2">Legal DNS verisi (paket)</td><td colspan="3">DNS Tünelleme verisi (paket)</td></tr>
<tr><td>Iodine</td><td>DNS2Cat</td><td>DNS2TCP</td></tr>
<tr><td>164692</td><td>127496</td><td>93921</td></tr>
<tr><td>412587</td><td colspan="3">386109</td></tr>
</table>

5.2.2. Veri ön işleme ve özellik çıkarma

Topladığımız verinin deri sinir ağına giriş olarak verilmesinden önce ön işleme aşamasının gerçekleşmesi gerekmektedir. Veri ön işleme, ağ başarımını doğrudan etkileyen temel aşamalardan biridir. Ön işlemeyle verinin kalitesinin artması, eğitim süresinin azalmasına ve kötü sınıflandırma performanslarının önüne geçilmesine fayda sağlamaktadır (Huang ve diğ., 2015). Çalışmamızda ön işlemede aşamasında yapılan işlemler aşağıda maddeler halinde anlatılmıştır:

- Her bir DNS paketi tek bir satır haline getirilmiştir. Legal ve tünel verisine ait satırlar birleştirilerek rastgele dağıtıldı. Böylece tünelli ve tünel olmayan verilerin bir arada olduğu bir veri seti oluşturuldu.
- DNS verisine ait tek bir satırda bulunan özellikler sütunlar halinde ayrıştırıldı.
- DNS verilerinde sayısal olmayan özellikler ve özelliklerin sahip olduğu değerler encode edilip etiketleme işlemi gerçekleştirildi.
- Encode işleminden sonra verideki gürültüden kurtulmak için tünelleme tespitinde kullanılmayan özellikler kaldırıldı. Tünel tespitinde kullanılan özelliklerinden biri de

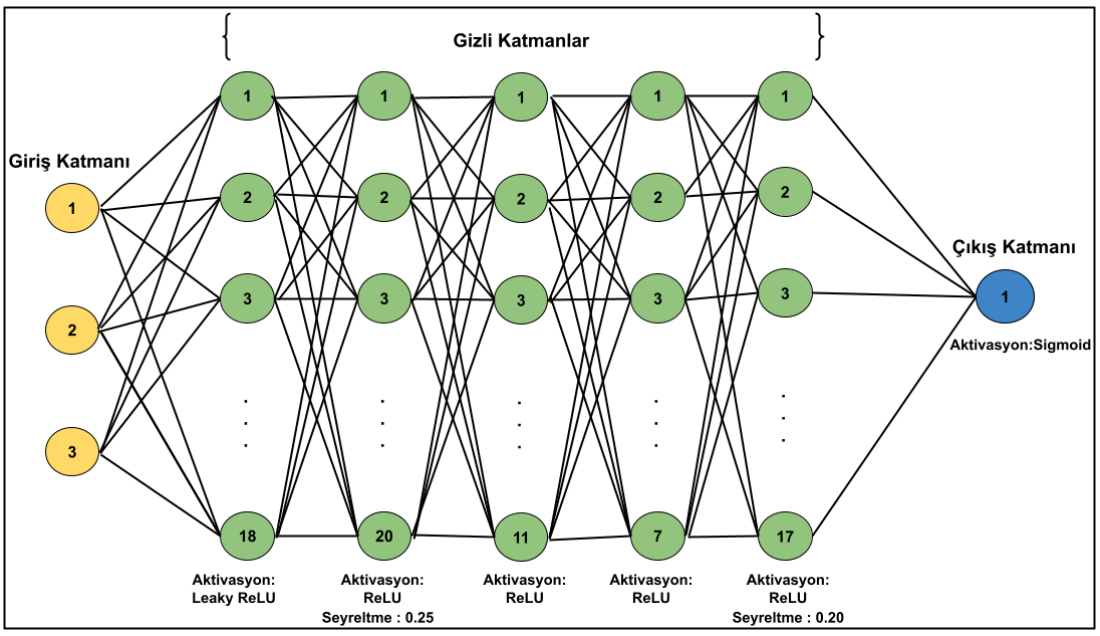
sorgu adına ait entropi bilgisi olduğu için, bu değerler de hesaplanarak veri setine dahil edildi.

• Ön işlemenin son aşamasında ise özellik ölçeklendirme işlemi gerçekleştirilmiştir. Özellik ölçeklendirme ön işlemede özelliklerin alabileceği değerleri standartlaştırmak için uygulanan önemli bir adımdır (Bollegala, 2017). Böylece tünel tespiti için kullanılan özelliklere özellik ölçeklendirme işlemi uygulanarak, özelliklere ait değerlerdeki değişimin ne kadar etkili olup olmadığı tespit edilmiştir. Özellik ölçeklendirme işlemiyle DNS tünelleme tespiti için DNS paketlerine ait IP uzunluğu, sorgu adı uzunluğu ve sorgu adı entropi bilgilerinin kullanılmasının uygun olduğu görülmüştür.

5.2.3. DFFN yöntemi

DNS tünelleme tespiti için önerilen DFFN yöntemine ait sinir ağı yapısı Şekil 5.3'te gösterilmiştir. Giriş katmanı DNS paketlerine ait IP uzunluğu, sorgu adı uzunluğu ve sorgu adı entropisi olmak üzere 3 hücreden oluşmaktadır. Sorgu adının entropi değeri, Eşitlik (5.6)'da gösterilen Shannon denklemi kullanılarak hesaplanmıştır.

$$H(x) = -\sum_{i=1}^{k} P(x_i) \log_2 P(x_i) \tag{5.6}$$



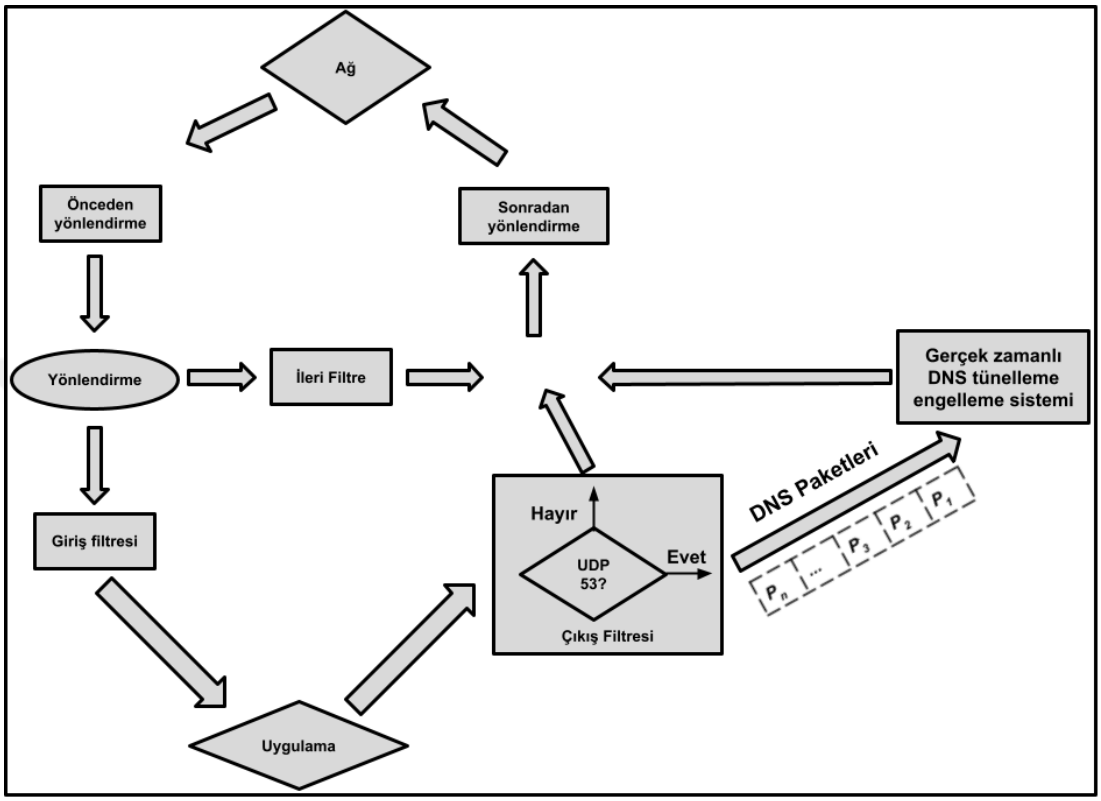
Şekil 5.3. DNS tünelleme tespiti için önerilen DFFN ağ yapısı

Önerilen DFFN mimarisi 5 gizli katmandan oluşturulmuştur. Birinci gizli katman 18, ikinci gizli katman 20, üçüncü gizli katman 11, dördüncü gizli katman 7 ve son gizli katman 17 hücreden oluşmaktadır. İlk iki gizli katmandaki hücre sayısının fazla, üçüncü ve dördüncü gizli katmanların nispeten az ve son gizli katmandaki hücre sayısının yine fazla olması şeklinde inşa edilmesi yoğun-seyrek-yoğun eğitim akışı olarak adlandırılmaktadır. Bu yöntem uygulandığında sınıflandırma problemlerinde elde edilen sonuçların daha başarılı olduğu görülmektedir (Han ve diğ., 2016). Aktivasyon fonksiyonu olarak ilk gizli katmanda Leaky ReLU tercih edilirken, diğer gizli katmanlarda ReLU tercih edilmiştir. Leaky ReLU, ReLU'ya göre daha iyi sonuç vermektedir. Fakat işlem yükü ve zamanı ReLU'ya göre çok daha fazladır. Bu yüzden sadece ilk katmanda Leaky ReLU kullanılması ağ optimizasyonu açısından en iyi durum olarak gözlemlenmiştir. DFFN ağında, ikinci gizli katmanda %25, beşinci gizli katmanda %20 seyreltme oranı kullanmıştır. DNS tünel tespiti bir sınıflandırma problemi olmasından dolayı çıkış katmanı tek hücreden oluşmaktadır (tünel yok=0, tünel var=1). Bu katmanda en yüksek başarı oranı sigmoid aktivasyon fonksiyonu ile elde edilmiştir. Bu nedenlerle çıkış katmanı için sigmoid fonksiyonu tercih edilmiştir.

5.2.4. DNS paketlerini yakalama ve yazılıma gönderme

DNS tünelleme tespitinin gerçeklenmesi aşamasından sonra tünel yoluyla gelen DNS paketlerinin engellenmesi için ilk olarak canlı bir ağ üzerindeki DNS paketlerini yakalamak gerekmektedir. Paket yakalama işlemi için Linux çekirdeği tarafından sağlanan Netfilter / IP tabloları sistemi (Wang ve diğ., 2016) kullanılmıştır. IP tabloları tablolar, zincirler ve hedefler olmak üzere 3 temel yapıdan oluşmaktadır. Tablolar, paket işleme sisteminin en önemli kısmıdır. Tablolar, filtre (giriş, çıkış, ileri), Mangle (önceden yönlendirme, sonradan yönlendirme, giriş, çıkış, ileri) ve NAT (önceden yönlendirme, sonradan yönlendirme, çıkış) olmak üzere 3 kısımdan oluşur. Paketlerin işlenmesi standart şekilde gerçekleşecekse filtre, TCP üstbilgisi gibi çeşitli üst kısımları değiştirilecekse mangle kullanılmaktadır. Paketlerin kaynağını veya hedefini yeniden yazmak için ise nat kullanılmaktadır. Zincirler bir tablo içindeki kurallar listesidir ve trafiğin kesilebileceği veya harekete geçirilebileceği yerlerdir. Hedefler ise kurallardan biriyle eşleşme olduğunda zincir içindeki pakete ne olacağını belirler. Mesela, belirli bir trafiğe uyan bir kural yazıp paketi bırak veya kabul et hedeflerine yönlendirmek örnek olarak gösterilebilir (Xuan ve Wu, 2015).

Tez kapsamında yapılan çalışmada da asıl amaç ağ trafiğindeki DNS paketlerini yakalamak olduğu için, filtre tablosundaki çıkış zincirindeki kurallara UDP protokolündeki 53 portundan çıkan tüm paketleri yazılıma; yani gerçek zamanlı DNS tünelleme engelleme sistemine yönlendirme işlemi eklenmiştir (Şekil 5.4).

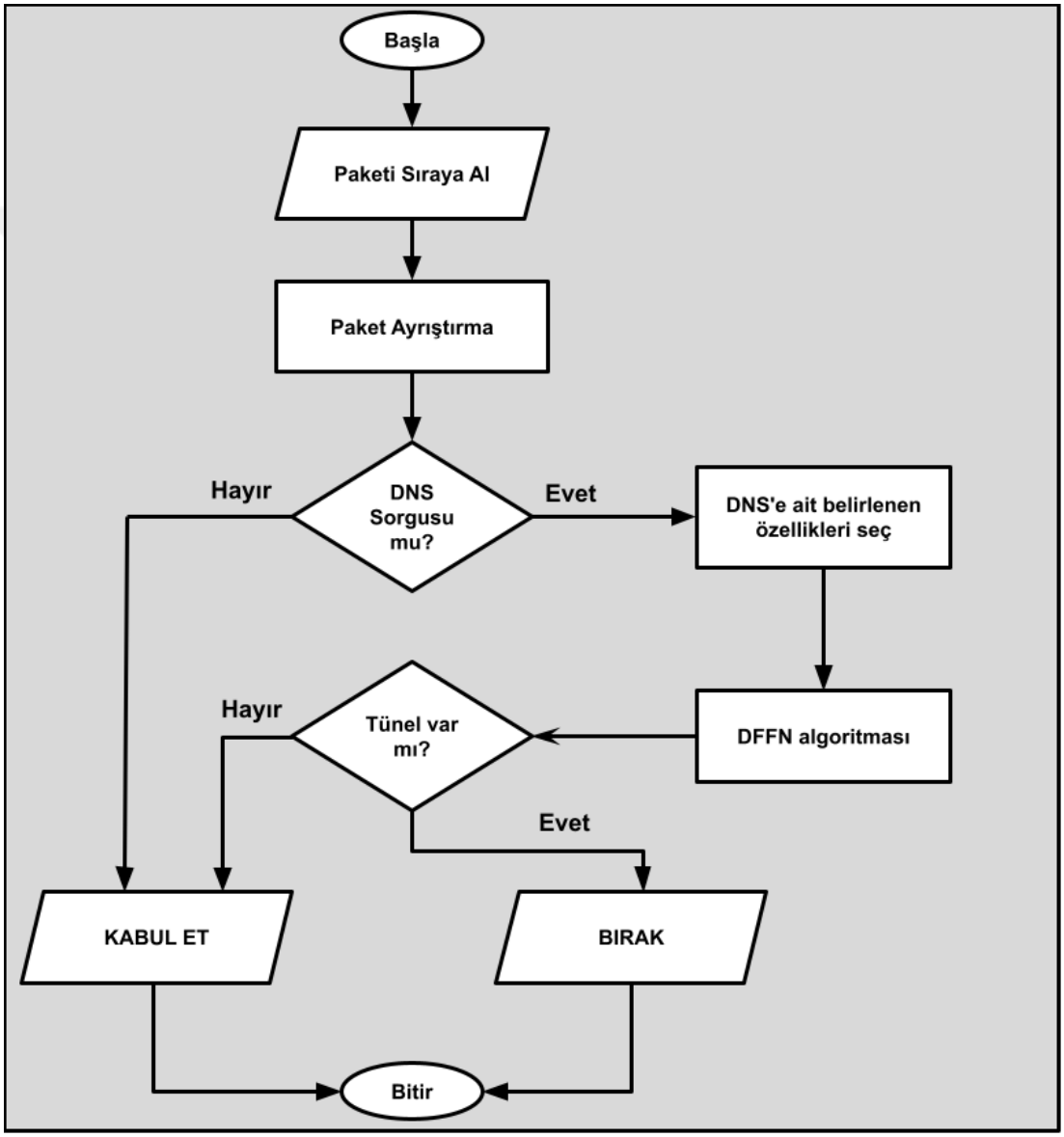


Şekil 5.4. DNS paketlerini yakalama ve sisteme gönderme

5.2.5. Gerçek zamanlı DNS tünelleme engelleme sistemi

Netfilter'dan gelen paketler ikili sistem formatındadır. Paketin çözümlenmesi zor ve zahmetli bir işlem olduğu için paket çözümleme işlemi için Philippe Biondi tarafından Python'da yazılan Scapy kütüphanesi (Rohith ve diğ., 2018) kullanılmıştır. İlk olarak Netfilter'dan gelen paketin sıraya alma işlemi gerçekleştirilmiştir. İkinci aşamada ikili sistem formatındaki verinin OSI (Open Systems Interconnection) referans modelindeki ağ, taşıma ve uygulama katmanına ait kısmı Scapy aracılığıyla kullanılmak üzere ayrıştırılmıştır. Ayrıştırma işleminde ilk olarak IP uzunluğu bilgisi ağ katmanından, gelen verinin DNS verisi olup olmadığının bilgisi, DNS verisi UDP 53 portu üzerinden taşındığı için taşıma katmanından ve sorgu adı bilgisi ise uygulama katmanından elde edilmiştir.

Pakete ait istenen bilgilerin elde edilmesinden sonra, gelen paketin DNS istek paketi olup olmadığı kontrolü yapılır. Eğer istek paketi değilse, paketin ilerlemesini devam ettirir. İstek paketi ise, DNS paketine ait belirlenen özellikler DFFN algoritmasına gönderilir. Algoritmanın verdiği cevaba göre paketin tünelli olup olmadığına karar verilir. Tünelli ise engellenir, değilse ilerlemesine devam ettirilir. Gerçek zamanlı engelleme işlemine ait akış şeması Şekil 5.5'te gösterilmiştir.



Şekil 5.5. Gerçek zamanlı DNS tünelleme engelleme sistemine ait akış şeması

6. DENEYSEL ÇALIŞMALAR

Deneysel çalışmalar, Intel (R) Core (TM) i7-9750H CPU 2.60GHZ'de, 16 GB RAM ve Windows 10'da çalışan bir bilgisayarda gerçekleştirilmiştir. Önerilen yöntemler Python'da açık kaynak kodlu bir makine öğrenmesi kütüphanesi olan H2O üzerinde gerçeklenmiştir.

6.1. Değerlendirme Metrikleri

Önerilen modellerin başarımı Accuracy, Precision, Recall, FAR ve F1-Score olmak üzere 5 farklı metrik kullanılarak değerlendirilmiştir. Bu metrikleri tanımlamak için TP, TN, FP ve FN olarak adlandırılan dört parametre kullanılmaktadır. Bu parametrelerden TP, STS'de doğru şekilde sınıflandırılan saldırı örneklerinin sayısını, TN doğru şekilde sınıflandırılan normal örneklerin sayısını, FP yanlış şekilde sınıflandırılan normal örneklerin sayısını, FN ise yanlış şekilde sınıflandırılan saldırı örneklerinin sayısını ifade etmektedir.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

$$\text{FAR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{6.4}$$

$$\text{F1-Score} = 2\frac{\text{Precision*Recall}}{\text{Precision} + \text{Recall}} \tag{6.5}$$

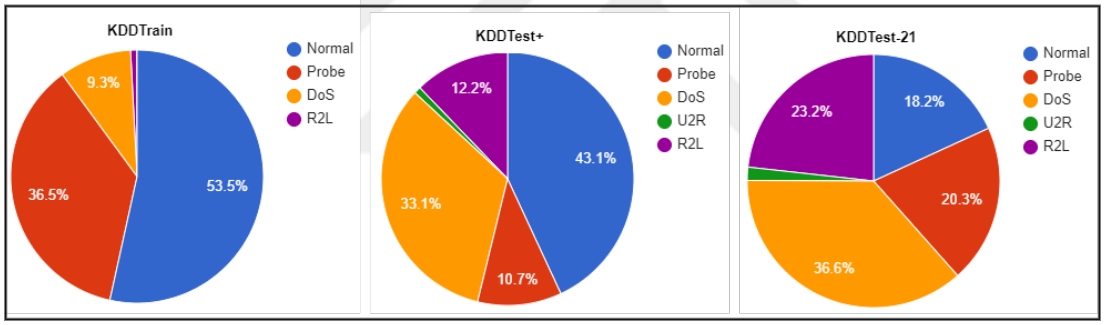
Accuracy (Eşitlik (6.1)), doğru sınıflandırılan örneklerin toplam örneklere oranını, Precision (Eşitlik (6.2)), doğru sınıflandırılan saldırı örneklerinin saldırı olarak tahmin edilen örneklerin toplamına oranını, Recall (Eşitlik (6.3)), doğru sınıflandırılan saldırı örneklerinin toplam saldırı örneklerine oranını, FAR (Eşitlik (6.4)), yanlış sınıflandırılan normal örneklerin toplam normal örneklere oranını ve F1-Score (Eşitlik

(6.5)) ise, elde edilen precision ve recall değerlerinin harmonik ortalamasını ifade etmektedir.

6.2. Probe, DoS, U2R ve R2L Saldırılarının Tespitinde Elde Edilen Sonuçlar

Önerilen yöntemin doğruluğu hem ikili hem de çoklu sınıflandırma olarak NSL-KDD veri setinde yer alan test setleri üzerinden gerçekleştirilmiştir. İkili sınıflandırmada sınıf etiketleri normal ve atak iken, çoklu sınıflandırmada Probe, DoS, U2R ve R2L saldırılarıdır.

Çalışmada kullanılan eğitim ve test veri setlerine ait örneklerin dağılım grafiği Şekil 6.1'de gösterilmiştir. Şekilde görüldüğü gibi normal örneklerin oranı KDDTrain veri setinde en fazla iken, KDDTest-21 veri setinde %18,2 oranıyla en azdır. Atak örneklerinin oranı KDDTest+ veri setinde %56,9 iken, KDDTest-21 veri setinde %81,8'dir.



Şekil 6.1. Eğitim ve test veri setlerinin dağılımı

Çalışma kapsamında elde edilen deneysel sonuçlar ikili ve çoklu sınıflandırma olmak üzere iki kısımda verilmiştir. İkili sınıflandırmada normal ve anormal atakların tespiti yapılırken, çoklu sınıflandırmada atak türlerinin (Probe, DoS, U2R, R2L) tespiti gerçekleştirilmiştir. İkili sınıflandırma işlemine ait karmaşıklık matrsisi sonuçları KDDTest+ ve KDDTest-21 test verileri için sırasıyla Tablo 6.1 ve Tablo 6.2'de gösterilmiştir.

Tablo 6.1. KDDTest+ veri setinde ikili sınıflandırma işlemine ait karmaşıklık matrisi

Gerçek / Öngörülen	Normal	Atak
Normal	9184	329
Atak	527	12504

Tablo 6.2. KDDTest-21 veri setinde ikili sınıflandırma işlemine ait karmaşıklık matrisi

Gerçek / Öngörülen	Normal	Atak
Normal	1653	329
Atak	499	9369

Tablo 6.1 ve Tablo 6.2'deki karmaşıklık matrisi değerleriyle hesaplanan Accuracy, Precision, Recall, FAR ve F1-Score değerleri ise Tablo 6.3'te verilmiştir.

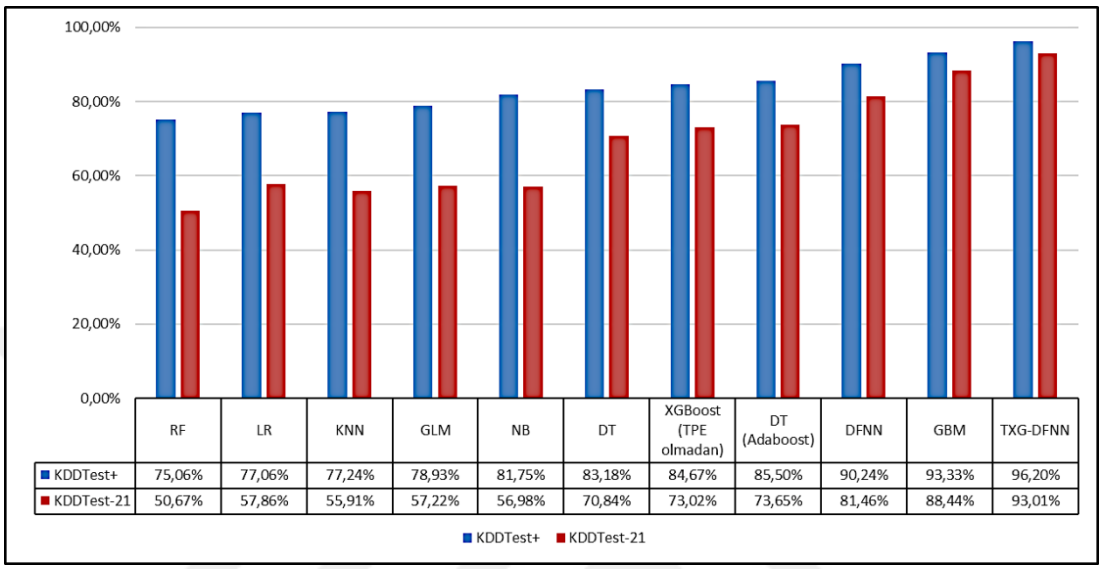
Tablo 6.3. İkili sınıflandırmaya ait deneysel sonuçlar

Kategori	Accuracy (%)	Precision (%)	Recall (%)	FAR (%)	F1-Score (%)
KDDTest+	96,20	96,54	94,57	0,02	95,55
KDDTest-21	93,01	83,40	76,81	0,03	80,12

İkili sınıflandırmada KDDTest+ veri setinde Accuracy, Precision, Recall, FAR ve F1-Score değerleri sırasıyla %96,20, %96,54%, %94,57, %0,02 ve %95,55'tir. KDDTest-21 veri setinde ise Accuracy, Precision, Recall, FAR ve F1-Score değerleri sırasıyla %93,01, %83,40, %76,81, %0,03 ve %80,12'dir. KDDTest-21 veri setinde Precision, Recall ve F1-Score değerleri accuracy değerine oranla düşüktür. Bu durum test veri setinde normal iken atak olarak sınıflandırılan örneklerin sayısından kaynaklanmaktadır. KDDTest-21 veri seti, KDDTest+ veri setine oranla daha az miktarda (yaklaşık olarak 4 kat) normal veri içermektedir. Aynı zamanda bu veriler özellik olarak atak şeklinde sınıflandırılmaya daha yatkın olan verilerin bir araya geldiği tespiti zor, nadir verilerdir. Bu nedenle KDDTest-21 veri seti literatürdeki pek çok çalışmada kullanılmamıştır. KDDTest-21 veri setinde yer alan bu duruma rağmen önerdiğimiz model ile Accuracy değerinde iyi sonuç alındığı, F1-Score değerinin ise Accuracy değerine göre dengeli olduğu görülmüştür. Aynı zamanda TXG-DFNN yöntemi literatürde yer alan, veri dengeleme işlemi uygulayan pek çok çalışmadan da daha iyi Accuracy ve F1-Score değeri elde etmiştir.

İkili sınıflandırmada Accuracy değerleri STS'de kullanılan geleneksel yöntemlerle karşılaştırıldığında elde edilen sonuçlar Şekil 6.2'de gösterilmiştir. Önerilen yöntem RF, LR, KNN, Generalized Linear Model (GLM), NB, DT, XGBoost (TPE yöntemi olmadan), Adaboost kullanılan DT, DFNN ve Gradient Boosted Trees (GBM) yöntemleri ile karşılaştırılmıştır. KDDTest+ test verisinde klasik makine yöntemleri arasında en başarılı sonuç Adaboost kullanılan DT (%85,50) iken, DFNN'de %90,24, GBM yönteminde ise %93,33 Accuracy elde edilmiştir. KDDTest-21 test verisinde ise

klasik makine yöntemleri arasında en başarılı sonuç yine Adaboost kullanılan DT (%73,65) iken, DFNN'de %81,46, GBM'de ise %88,44 Accuracy elde edilmiştir. Önerdiğimiz yöntem, geleneksel yöntemlere oranla hem KDDTest+ hem de KDDTest-21 veri setlerinde en başarılı sonucu elde etmiştir.

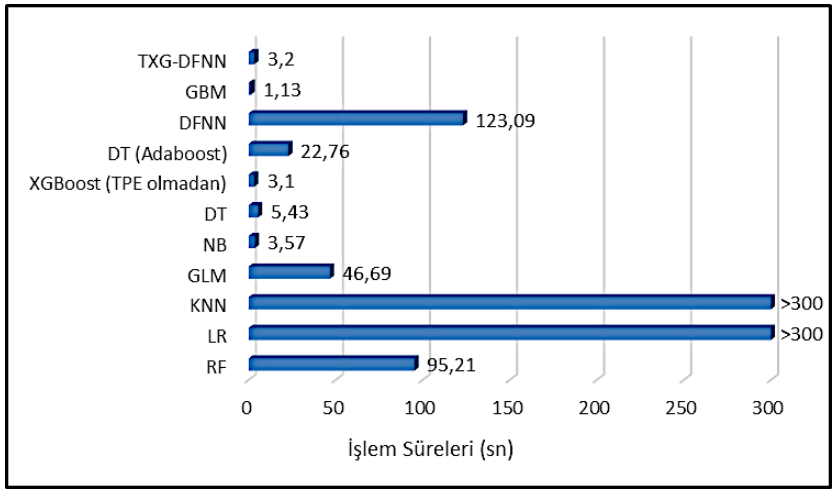


	RF	LR	KNN	GLM	NB	DT	XGBoost (TPE olmadan)	DT (Adaboost)	DFNN	GBM	TXG-DFNN
KDDTest+	75,06%	77,06%	77,24%	78,93%	81,75%	83,18%	84,67%	85,50%	90,24%	93,33%	96,20%
KDDTest-21	50,67%	57,86%	55,91%	57,22%	56,98%	70,84%	73,02%	73,65%	81,46%	88,44%	93,01%

Şekil 6.2. İkili sınıflandırmada elde edilen accuracy değerlerinin geleneksel yöntemlerle karşılaştırılması

Şekil 6.3'te ikili sınıflandırma işleminde karşılaştırılan yöntemlere ait işlem süreleri verilmiştir. İşlem süreleri, eğitim ve doğrulama sürelerinin toplamı olarak hesaplanmıştır. Şekilde de görüldüğü gibi LR ve KNN yöntemleri işlem süreleri açısından en kötü (300 saniyeden fazla) performansı göstermiştir. DFNN yöntemi 123,09 saniyede işlem süresini tamamlarken, GBM 1,13 saniye olarak en hızlı sınıflandırma işlemini gerçekleştirmiştir. Önerilen yönteme ait işlem süresi ise 3,20 saniyedir. GBM algoritması işlem süresi açısından en hızlı yöntem olmasına rağmen doğruluk açısından TXG-DFNN yöntemini geçememiştir. TXG-DFNN yöntemi kullandığı hibrit yapı sayesinde ikili sınıflandırma açısından hem doğruluğu yüksek hem de işlem süresi optimuma yakın bir sonuç elde etmiştir.

Çoklu sınıflandırma için KDDTest+ ve KDDTest-21 test verilerine ait karmaşıklık matrisi sonuçları sırasıyla Tablo 6.4 ve Tablo 6.5'te gösterilmiştir.



Şekil 6.3. İkili sınıflandırmada işlem süresinin geleneksel yöntemlerle karşılaştırılması

Tablo 6.4. KDDTest+ veri setinde çoklu sınıflandırma işlemine ait karmaşıklık matrisi

Gerçek / Öngörülen	Normal	Probe	DoS	U2R	R2L
Normal	9184	1	21	0	307
Probe	258	2232	138	132	246
DoS	207	188	7046	6	469
U2R	1	0	0	32	8
R2L	61	0	253	30	1724

Tablo 6.5. KDDTest-21 veri setinde çoklu sınıflandırma işlemine ait karmaşıklık matrisi

Gerçek / Öngörülen	Normal	Probe	DoS	U2R	R2L
Normal	1653	1	21	0	307
Probe	251	2213	137	132	246
DoS	192	188	3931	6	469
U2R	1	0	0	32	8
R2L	55	0	253	30	1724

Karmaşıklık matrisi değerleriyle hesaplanan Accuracy, Precision, Recall, FAR ve F1-Score değerleri ise Tablo 6.6 ve Tablo 6.7'de verilmiştir.

Tablo 6.6. KDDTest+ veri setinde çoklu sınıflandırma işlemine ait deneysel sonuçlar

Kategori	Accuracy (%)	Precision (%)	Recall (%)	FAR (%)	F1-Score (%)
Normal	96,20	96,54	94,57	4,04	95,54
Probe	95,73	74,25	92,19	0,97	82,25

Tablo 6.6. (Devam) KDDTest+ veri setinde çoklu sınıflandırma işlemine ait deneysel sonuçlar

DoS	94,31	89,01	94,48	2,82	91,66
U2R	99,21	78,05	16,00	0,75	26,55
R2L	93,91	83,37	62,60	5,03	71,51
Ortalama	89,68				

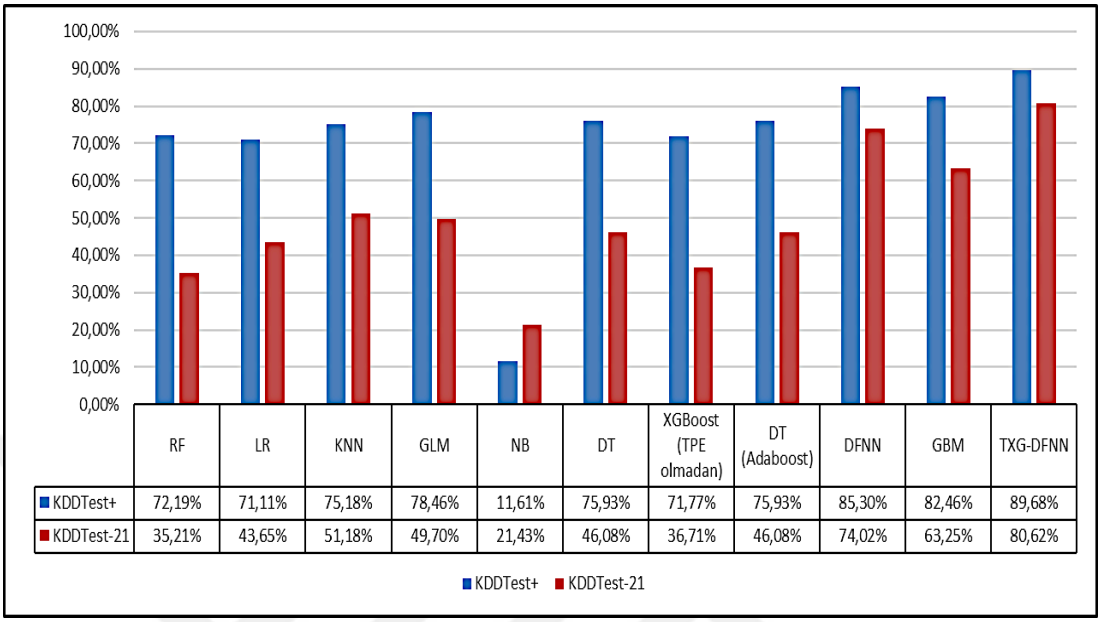
Tablo 6.7. KDDTest-21 veri setinde çoklu sınıflandırma işlemine ait deneysel sonuçlar

Kategori	Accuracy (%)	Precision (%)	Recall (%)	FAR (%)	F1-Score (%)
Normal	93,01	83,40	76,81	5,06	79,97
Probe	91,94	74,29	92,13	2,13	82,25
DoS	89,32	82,14	90,53	5,82	86,13
U2R	98,51	78,05	16,00	1,42	26,55
R2L	88,46	83,61	62,60	10,52	71,60
Ortalama	80,62				

Çoklu sınıflandırma için önerilen modelde, KDDTest+ veri setinde %89,68, KDDTest-21 veri setinde ise %80,62 accuracy elde edilmiştir. Önerilen model geleneksel yöntemlerle karşılaştırıldığında elde edilen sonuçlar Şekil 6.4'te gösterilmiştir. KDDTest+ test verisinde geleneksel makine yöntemleri arasında en başarılı sonuç GLM (%78,46) iken, DFNN 'de %85,30, GBM yönteminde ise %82,46 accuracy elde edilmiştir. KDDTest-21 test verisinde ise klasik makine yöntemleri arasında en başarılı sonuç yine GLM (%49,70) iken, DFNN'de %74,02, GBM'de ise %63,25 accuracy elde edilmiştir. Elde edilen sonuçlardan geleneksel makine öğrenmesi ve derin öğrenme tekniklerinin tek başlarına kullanılmaları durumunda düşük başarılar elde ettiği ve yetersiz oldukları anlaşılmıştır. TXG-DFNN yöntemi hem makine öğrenmesi hem de DFNN yöntemlerinin avantajlarını bir araya getirerek, geleneksel yöntemlere oranla hem KDDTest+ hem de KDDTest-21 veri setlerinde yüksek accuracy değerleri elde etmiştir.

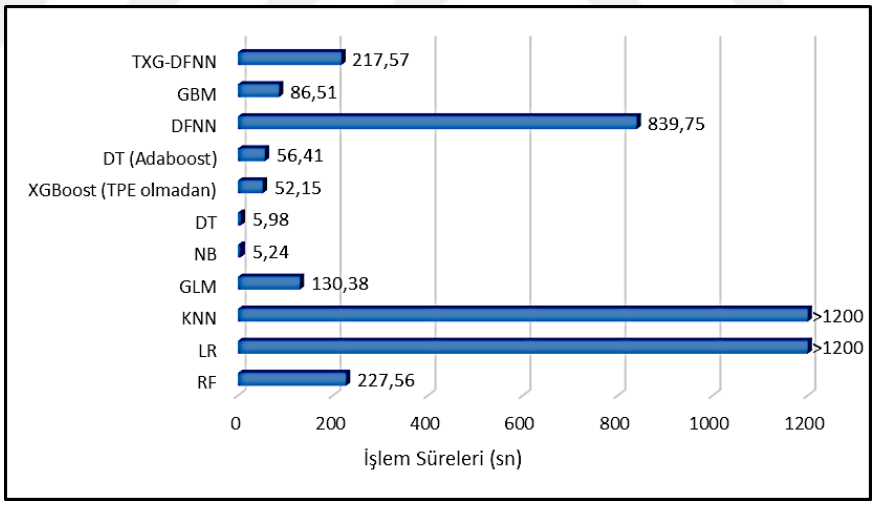
Şekil 6.5'te çoklu sınıflandırma işleminde karşılaştırılan yöntemlere ait işlem süreleri verilmiştir. Şekilden de görüldüğü gibi kısa sürede sınıflandırma yapan tekniklerin hata oranının yüksek olduğu, %80 üzeri başarısı olan yöntemlerin ise işlem sürelerinin fazla olduğu görülmektedir. Kullanılan hibrit mimari sayesinde klasik DFNN mimarisinin bir dezavantajı olan yüksek işlem süresinin, ilk aşamada kullanılan XGBoost yöntemi sayesinde yarı yarıya düşürüldüğü gözlemlenmiştir. Böylece çoklu

sınıflandırma işlemi için de ortalama sürelerde işlem yapan doğruluğu yüksek bir yöntem elde edilmiştir.



	RF	LR	KNN	GLM	NB	DT	XGBoost (TPE olmadan)	DT (Adaboost)	DFNN	GBM	TXG-DFNN
KDDTest+	72,19%	71,11%	75,18%	78,46%	11,61%	75,93%	71,77%	75,93%	85,30%	82,46%	89,68%
KDDTest-21	35,21%	43,65%	51,18%	49,70%	21,43%	46,08%	36,71%	46,08%	74,02%	63,25%	80,62%

Şekil 6.4. Çoklu sınıflandırmada elde edilen accuracy değerlerinin geleneksel yöntemlerle karşılaştırılması



Şekil 6.5. Çoklu sınıflandırmada işlem süresinin geleneksel yöntemlerle karşılaştırılması

Tablo 6.8 ve Tablo 6.9, KDDTest+ ve KDDTest-21 veri setleri üzerinde önerilen modelin literatürde yer alan çalışmalarla karşılaştırmalı sonuçlarını göstermektedir. Sonuçlar accuracy değerleri üzerinden ikili ve çoklu sınıflandırma için ayrı ayrı verilmiştir. Tablo 6.8'de önerdiğimiz model KDDTrain eğitim veri seti üzerinde,

çalışmamızda da olduğu gibi dengeleme işlemi yapmayan çalışmalarla, Tablo 6.9'da ise, KDDTrain üzerinde dengeleme işlemi yapan çalışmalarla karşılaştırılmıştır.

Tablo 6.8. Önerilen modelin KDDTrain eğitim veri setinde değişiklik yapılmamış çalışmalarla karşılaştırılması

Yazar	İkili Sınıflandırma Accuracy (%)		Çoklu Sınıflandırma Accuracy (%)	
	KDDTest+	KDDTest-21	KDDTest+	KDDTest-21
Wu ve diğ., (2018)	81,29	64,67	79,48	60,71
Latah ve Toker (2020)	84,29	-	-	-
Wang ve diğ., (2020)	80,60	-	80,30	-
Su ve diğ., (2020)	84,25	69,42	-	-
Al-Qatf ve diğ., (2018)	84,96	-	80,48	-
Tchakoucht ve Ezziyyani (2018)	83,00	66,20	-	-
Li ve diğ., (2020)	86,95	76,65	81,33	64,81
Wu ve diğ., (2020)	85,73	70,25	82,36	66,25
Liu ve diğ., (2020)	81,30	64,70	78,30	60,90
Rathore ve Park (2018)	86,53	75,77	-	-
Tama ve diğ., (2019)	85,79	72,52	-	-
Önerilen model	96,20	93,01	89,68	80,62

Tablo 6.9. Önerilen modelin KDDTrain eğitim veri setinde değişiklik yapılan çalışmalarla karşılaştırılması

Yazar	İkili Sınıflandırma Accuracy (%)		Çoklu Sınıflandırma Accuracy (%)	
	KDDTest+	KDDTest-21	KDDTest+	KDDTest-21
Hu ve diğ., (2020)	-	-	84,08	72,54
Kasongo ve Sun (2019)	87,74	-	86,19	-
Hou ve diğ., (2020)	-	-	83,85	69,73
Haggag ve diğ., (2020)	85,44	-	83,57	70,59
Liu ve diğ., (2020)	81,30	64,70	78,30	60,90
Jiang ve diğ., (2020)	83,58	-	80,05	-

Tablo 6.9. (Devam) Önerilen modelin KDDTrain eğitim veri setinde değişiklik yapılan çalışmalarla karşılaştırılması

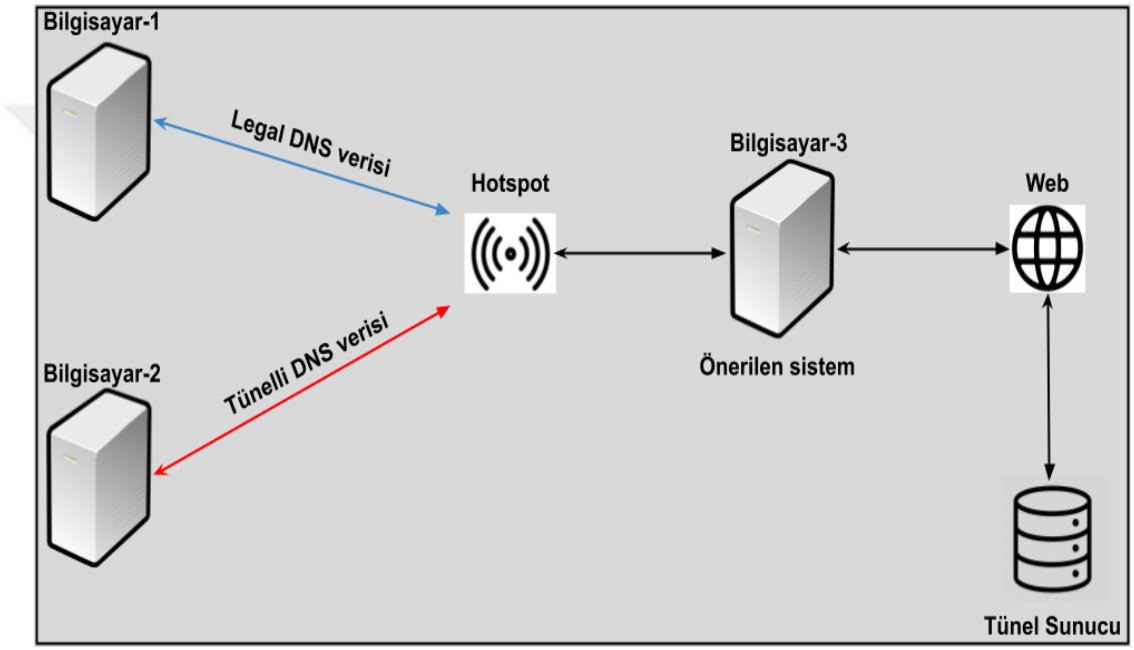
Yazar	İkili Sınıflandırma Accuracy (%)		Çoklu Sınıflandırma Accuracy (%)	
	KDDTest+	KDDTest-21	KDDTest+	KDDTest-21
Ieracitano ve diğ., (2020)	84,21	-	87,00	-
Gao ve diğ., (2018)	-	-	84,54	71,29
Yang ve diğ., (2019)	-	-	82,08	66,18
Önerilen model	96,20	93,01	89,68	80,62

Tablo 6.8 ve Tablo 6.9'da görüldüğü gibi Jiang ve diğ., (2020), Latah ve Toker (2020), Wang ve diğ., (2020), Al-Qatf ve diğ., (2018), Ieracitano ve diğ., (2020)'nin yaptıkları çalışmada sadece KDDTest+ veri seti üzerinden değerlendirme yapılırken, Latah ve Toker (2020), Su ve diğ., (2020), Tama ve diğ., (2019), Tchakoucht ve Ezziyyani (2018), Rathore ve Park (2018) tarafından yapılan çalışmalarda sadece ikili sınıflandırma Hu ve diğ., (2020), Hou ve diğ., (2020), Gao ve diğ., (2018), Yang ve diğ., (2019) tarafından yapılan çalışmalarda ise sadece çoklu sınıflandırma yapılmıştır. Tablo 6.8'de görüldüğü gibi TXG-DFNN yöntemi dengeleme işlemi yapmayan güncel çalışmalar arasında hem KDDTest+, hem de KDDTest-21 veri setlerinde en iyi sonucu elde etmiştir. Tablo 6.9'da görüldüğü gibi ise, dengeleme işlemi yapan güncel bazı çalışmalardan daha iyi başarım gösterdiği tespit edilmiştir. Böylece önerilen modelin, dengesiz veri setlerinde ihtiyaç duyulan indirgeme, veri arttırma, veri temizleme vb. işlemlere ihtiyaç olmadan da özellikle ikili sınıflandırma açısından yüksek başarım elde edildiği açıkça görülmüştür.

6.3. DNS Tünelleme Tespiti ve Engellenmesinde Elde Edilen Sonuçlar

Tez kapsamında DNS tünelleme tehditlerinin tespiti ve engellenmesine yönelik gerçek zamanlı çalışan test düzeneği oluşturulmuştur (Şekil 6.6). Şekilden de görüleceği üzere test düzeneği için bir ağ oluşturulmuştur. Ağ üzerindeki 3 cihazdan Bilgisayar-1 legal, Bilgisayar-2 ise tünelli DNS paketlerini Bilgisayar-3'e göndermektedir. Bilgisayar-1 ve Bilgisayar-2 internete direkt olarak bağlı değildir. Bilgisayar-3 tarafından kurulan ağ üzerinden (hotspot) internete çıkış yapmaktadır. Bilgisayar-1 standart şekilde web'e

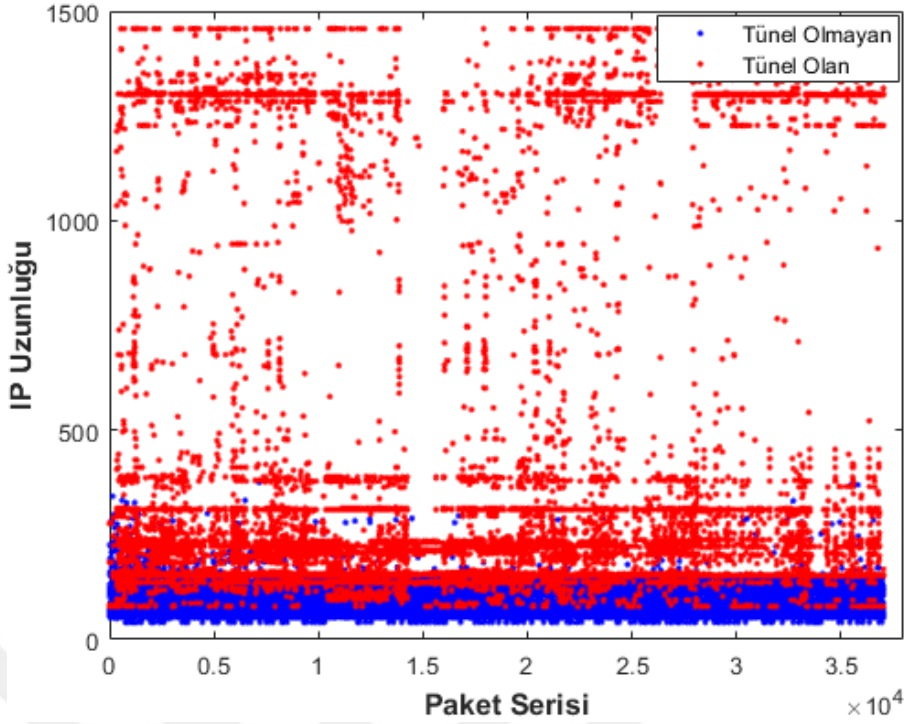
istekte bulunurken, Bilgisayar-2, web üzerinden çalışan harici bir tünel sunucusu üzerinden istekte bulunmaktadır. Bilgisayar-1 ve Bilgisayar-2 istekte bulunmaya başladığında, Bilgisayar-3'te kurulu olan çalışma kapsamında önerilen sistem çalıştırılmaya başlatılmıştır. Bu bilgisayar, algoritmanın verdiği cevaba göre paketin tünelli olup olmadığına karar verir. Tünelli ise engellenir, değilse ilerlemesine devam ettirilir. Bilgisayar-1 ve Bilgisayar-2'den gelen DNS paketlerinin özellikleri Tablo 6.10'da gösterilmiştir. DNS paketlerinin %10'una ait IP uzunluğu, sorgu adı uzunluğu ve sorgu adı entropi değerleri Şekil 6.7, Şekil 6.8 ve Şekil 6.9'da gösterilmiştir.



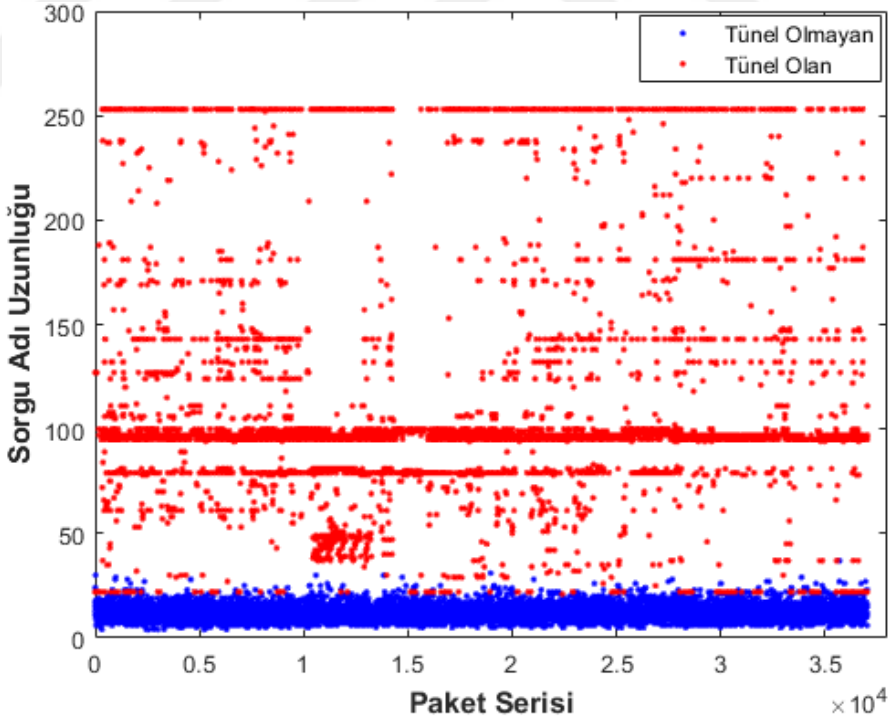
Şekil 6.6. Gerçek zamanlı test şeması

Tablo 6.10. Gerçek zamanlı test veri seti özeti

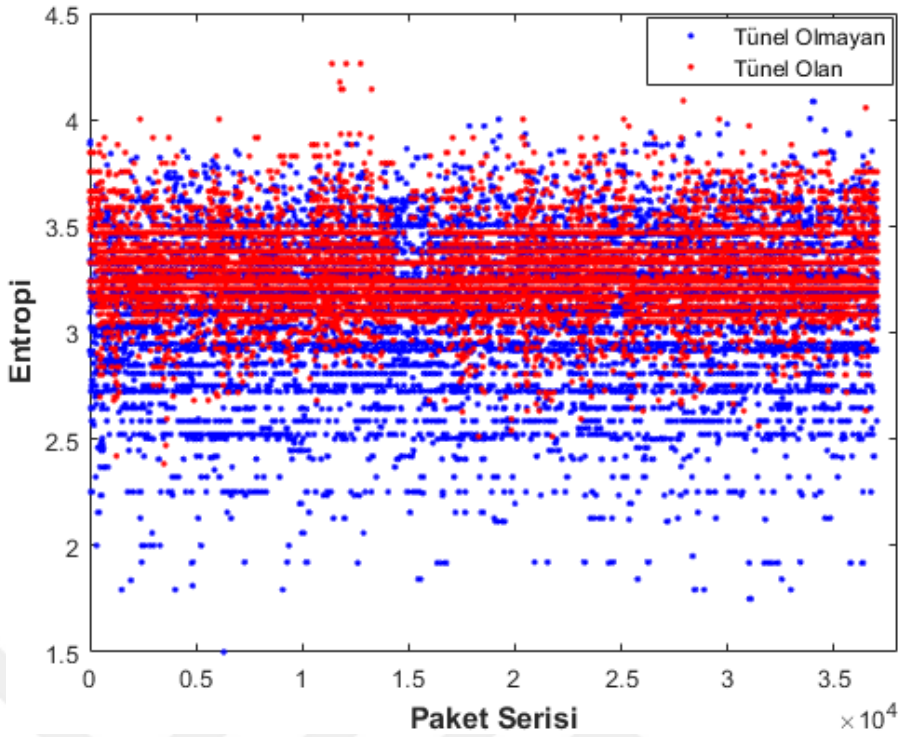
Legal DNS paketleri	390992
Tünelli DNS paketleri	370548
DNS sorguları	412214
DNS yanıtları	349326
Giden DNS sorguları	361747
Toplam DNS paketleri	761540



Şekil 6.7. Canlı ağa gönderilen DNS paketlerine ait IP uzunluğu değerleri



Şekil 6.8. Canlı ağa gönderilen DNS paketlerine ait sorgu adı uzunluğu değerleri



Şekil 6.9. Canlı ağa gönderilen DNS paketlerine ait entropi değerleri

Şekil 6.6'daki Bilgisayar-1 ve Bilgisayar-2'den gelen DNS paketleri ile DFNN algoritmasıyla analiz edilen sonuçların karşılaştırılmasıyla elde edilen ikili sınıflandırma değerlendirme sonuçları Tablo 6.11'de verilmiştir. Tablo 6.11'deki Accuracy, doğru tahmin edilen toplam legal ve tünelli DNS paket sayısının toplam DNS paket sayısına oranını ifade etmektedir. Precision, doğru tahmin edilen tünelli paket sayısının, doğru ve yanlış tahmin edilen tünelli paket sayısına oranıyla elde edilmiştir. Recall, doğru tespit edilen tünelli paket sayısının, doğru tespit edilen tünelli ve yanlış tespit edilen legal paket sayılarının toplamına oranını ifade eder. F1-Score ise, elde edilen Precision ve Recall değerlerinin tek bir değer olarak değerlendirilmesi için, bu iki değerin harmonik ortalaması alınarak hesaplanmıştır.

Tablo 6.11. DNS tünelleme tespiti ikili sınıflandırma sonuçları

Accuracy	%99,91
Precision	%99,98
Recall	%99,84
F1-Score	%99,91

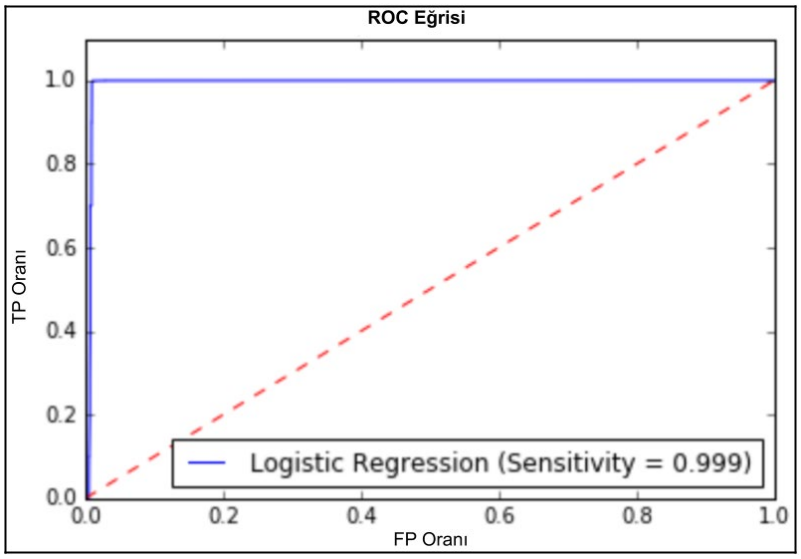
Tablo 6.12'de, önerilen yöntem için veri setinde yer alan 3 ve 7 özellik kullanılarak ayrı ayrı hesaplanan accuracy değerleri gösterilmiştir. İlk aşamada kullanılan 7 özellik

DNS tünellemede yaygın olarak kullanılan özellikler arasından seçilmiştir. Diğer aşamada kullanılan 3 özellik ise veri setine özgü olarak yapılan özellik indirgeme işlemi sonrasında elde edilen özellikleri temsil etmektedir. Sonuçlardan da anlaşılacağı gibi özellik indirgeme işlemi ile veri seti üzerinde en belirleyici olan özellikler tespit edilmiştir. Böylece hem daha yüksek accuracy değerleri hem de daha az özellik sayısı ile modelin efektif olarak eğitilmesi sağlanmıştır. Tablodaki sonuçlardan veri setinde yer alan etkisi az özelliklerin kullanımının yöntem başarısı üzerinde olumlu yönde etkisi olmadığı anlaşılmaktadır.

Tablo 6.12. Özellik sayısına bağlı olarak elde edilen accuracy değerleri

	3 Özellik	7 Özellik
Özellik Adı	IP uzunluğu, sorgu adı uzunluğu, sorgu adı entropisi	IP uzunluğu, sorgu adı uzunluğu, sorgu adı entropisi, IP başlık uzunluğu, IP_flag_mf, IP_flag_rb, IP_flag_z
Accuracy	%99,91	%99,32

Önerilen sisteme ait ROC (Receiver Operating Characteristic) eğrisi Şekil 6.10'da gösterilmiştir. Literatürde mükemmele yakın bir sistemin ROC eğrisinin dikey (0,0)'dan (0,1)'e ve yatayda (1,1)'den geçecek bir eğriye sahip olması gerektiği ifade edilmiştir (Tomak ve Bek, 2009). Şekilden de anlaşılacağı gibi önerilen sistem sol üst köşeye çok yakın bir eğriye sahiptir. Sistemin yüksek bir doğruluk oranına sahip olduğu ROC eğrisinden de açıkça görünmektedir.



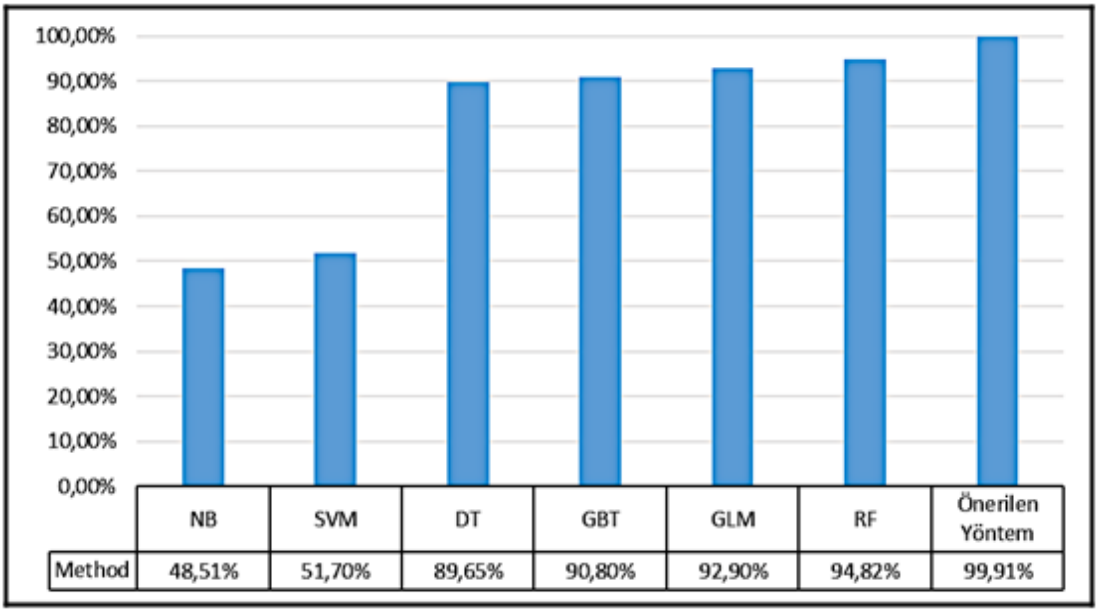
Şekil 6.10. Önerilen sistemin ROC eğrisi

Önerilen gerçek zamanlı sistemde, her bir paket için DNS tüneli yapılıp yapılmadığının tespit edilmesi ve tünel yoluyla gelen bir paketin engellenmesi için geçen ortalama süre Tablo 6.13'te gösterilmiştir.

Tablo 6.13. Gerçek zamanlı sistemde her bir DNS paketi için geçen süre

DNS tünellemenin tespiti	0,614 ms.
DNS tünellemenin engellenmesi	0,309 ms.
Her DNS paketi için geçen toplam süre	0,923 ms.

Önerilen yöntemde elde edilen accuracy değerlerinin genel olarak DNS tünellemede kullanılan yöntemlerle karşılaştırılması Şekil 6.11'de gösterilmiştir. Önerilen yöntem, NB, SVM, DT, GBM, GLM ve RF yöntemleri ile karşılaştırılmıştır. Geleneksel makine öğrenme yöntemleri arasında en başarılı sonuç RF (%94,82) ile elde edilirken, NB ile %48,51, SVM ile %51,70, DT ile %89,65, GBT ile %90,80 ve GLM yöntemi ile %92,90 accuracy elde edilmiştir. Tablo 6.12'den görülebileceği gibi, veri setinden seçilen 3 özellik, sınıflandırma süreci üzerinde yüksek bir etkiye sahiptir. Ancak Şekil 6.11'de görüldüğü gibi geleneksel makine öğrenme yöntemlerinde bu üç özellik ile işlemler gerçekleştirilse bile, önerilen modelde daha başarılı sonuçlar elde edilmiştir.



Şekil 6.11. Önerilen yöntemin doğruluk değerinin geleneksel yöntemlerle karşılaştırılması

Derin öğrenmede eğitim hızı ve modellerin son tahmin performansını belirlemek için optimizasyon algoritmaları kullanılmaktadır. Kullanılan optimizasyon algoritmaları minimum noktaya adım adım ilerleyen Gradient Descent (1. Türev) tabanlı yöntemlerdir. Adım miktarının (öğrenme katsayısı) büyük seçilmesinin minimum noktaya ulaşamama, küçük seçilmesinin ise minimum noktaya zaman açısından geç ulaşma dezavantajlarını oluşturabilir. Bu yüzden optimizasyon algoritması seçimi derin öğrenmede önemli aşamalardan biridir (Seyyarer ve diğ., 2019). Çalışma kapsamında en çok tercih edilen optimizasyon algoritmaları da karşılaştırılmış ve elde edilen başarı oranları Tablo 6.14'te verilmiştir. Tabloda görüldüğü gibi optimizasyon algoritmaları arasında başarı oranlarında yüksek oranda bir fark olmamasına rağmen, en başarılı sonuçlar Adamax ile elde edilmiştir. Ayrıca çalışmada batch size 32, epoch sayısı ise 50 olarak belirlenmiştir.

Tablo 6.14. Farklı optimizasyon algoritmalarına göre elde edilen accuracy değerleri

Optimizasyon Algoritması	Accuracy (%)
Sgd	Öğrenme yok
Rmsprop	Öğrenme yok
Adadelta	95,90
Nadam	97,84
Adagrad	98,89
Adam	99,12
Adamax	99,91

Tablo 6.15'te, önerilen modelin literatürdeki benzer çalışmalarla karşılaştırma sonuçlarını göstermektedir. Bu çalışmalarda, DNS tünelleme verileri bizim çalışmamızda olduğu gibi Iodine, DNS2Cat veya DNS2TCP araçlarıyla elde edilmiştir. Tabloda görüldüğü gibi Buczak ve diğ., (2016), Homem ve diğ., (2017), Ahmed ve diğ., (2019), Aiello ve diğ., (2019) ve Bubnov (2018) tarafından yapılan çalışmalarda accuracy değerleriyle karşılaştırmalar yapılmıştır. Ayrıca Sammour ve diğ., (2017), Almusawi ve Amintoosi (2018) tarafından yapılan çalışmalarda karşılaştırma işlemi accuracy değerleri verilmediği için F1-Score değerleriyle yapılmıştır. Ahmed ve diğ., (2019)'nin yaptıkları çalışmada tünel tespiti gerçek zamanlı yapılırken, diğer çalışmalarda gerçek zamanlı olarak tespit veya önleme

yapacak bir sistemin önerilmediği görülmektedir. Tablodan da görülebileceği gibi önerilen model ile literatürdeki çalışmalara göre gerçek zamanlı olarak DNS tünellemenin engellenmesinde yüksek başarı oranı elde edilmiştir.

Tablo 6.15. Önerilen yöntemin literatürdeki benzer çalışmalarla karşılaştırılması

Yazar	Accuracy (%)	F1-Score (%)	Gerçek Zamanlı Tespit	Gerçek Zamanlı Engelleme
Buczak ve diğ., (2016)	95,40	-	Yok	Yok
Homem ve diğ., (2017)	96,00	-	Yok	Yok
Ahmed ve diğ., (2019)	98,90	-	Var	Yok
Aiello ve diğ., (2019)	90,70	-	Yok	Yok
Bubnov (2018)	83,00	-	Yok	Yok
Sammour ve diğ., (2017)	-	83,00	Yok	Yok
Almusawi ve Amintoosi (2018)	-	80,00	Yok	Yok
Önerilen yöntem	99,91	99,91	Var	Var

7. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında farklı saldırı türleri için iki farklı sistem önerilmiştir. İlk aşamada Probe, DoS, U2R ve R2L saldırılarının tespit edilmesi gerçekleştirilmiştir. İkinci aşamada ise DNS üzerinden yapılan en yaygın saldırı türlerinden biri olan DNS tünellemeyi gerçek zamanlı olarak engelleyen bir sistem önerilmiştir.

Tez kapsamında ilk aşamada, XGBoost algoritması ve DFNN yöntemini birleştiren hibrit bir saldırı tespit sistemi önerilmiştir. Önerilen mimarinin testleri, NSL-KDD veri kümesi üzerinde gerçekleştirilmiştir. Yöntemin gerçek ağ verilerine uygunluğunu göstermek için, NSL-KDD veri kümesindeki kayıtlar hem eğitim hem test aşamasında herhangi bir değişikliğe tabi tutulmadan orijinal haliyle kullanılmıştır. TXG-DFNN olarak isimlendirilen mimari, sınıflandırma işlemini iki aşamalı olarak gerçekleştirmektedir. İlk aşamada TPE tabanlı optimize edilmiş XGBoost yöntemi kullanılarak ikili sınıflandırma işlemi, ikinci aşamada ise ikili sınıflandırmada atak olarak tespit edilen verilerin türlerine ayrılması işlemi gerçekleştirilmektedir. Bu aşamada probleme özgü olarak oluşturulmuş bir DFNN mimarisi kullanılmıştır. Önerilen modele göre ikili sınıflandırma accuracy değeri KDDTest+ veri setinde %96,20 iken, KDDTest-21 veri setinde %93,01, çoklu sınıflandırma accuracy değeri KDDTest+ veri setinde %89,68, KDDTest-21 veri setinde %80,62 olarak elde edilmiştir. Deneysel sonuçlar geleneksel yöntemlerle kıyaslandığında, önerilen hibrit yöntemin hem ikili hem de çoklu sınıflandırmada daha iyi sonuçlar verdiğini göstermektedir. Literatürdeki çalışmalar ile karşılaştırıldığında ise TXG-DFNN yönteminin veri setini benzer şekilde kullanan çalışmalara göre daha başarılı olduğu görülmüştür. Aynı zamanda yöntem veri seti üzerinde dengeleme işlemi yapan bazı çalışmalardan da daha etkili sonuçlar göstermiştir.

Tez kapsamında ikinci aşamada ise DNS üzerinden yapılan tünelleme saldırılarını canlı ağlarda gerçek zamanlı olarak engelleyen derin ağ tabanlı bir sistem önerilmiştir. Çalışmada ilk olarak Alexa top 1 million sitesindeki veriler kullanılarak, legal ve tünel verileri elde edilmiştir. Daha sonra bu veriler kullanılarak 2 farklı topoloji ile derin ağlar eğitilmiştir. En yüksek başarı oranı elde edilen topoloji sisteme entegre edildikten

sonra, sistemin başarımı canlı ağ üzerinde test edilmiştir. Testler esnasında sistem gerçek zamanlı olarak gelen tehditleri engelleyecek şekilde ağa entegre edilmiştir. Yapılan analizlerde tehdit sayıları ve bu tehditlerin ne kadarının sistem tarafından önlendiğine dair istatistikler çıkarılmıştır.

Deneysel sonuçlar, DNS tünellemenin %99,91 Accuracy, %99,98 Precision, %99,84 Recall ve %99,91 F1-Score oranlarında tespit edildiği görülmektedir. Bu oranlar sistemin DNS trafiğindeki tünelleme tehditlerini engellemede yüksek başarı oranı elde ettiğini göstermiştir.

Tablo 6.13'te görüldüğü gibi önerilen sistemin bir DNS paketinin tünelli olup olmadığına karar vermesi ortalama 0,614 ms ve engelleme süresi ise 0,309 ms'dir. Her bir DNS sorgusuna toplamda 0,923 ms'de karar veren sistemin, saniyede yaklaşık olarak 1080 DNS sorgusuna yanıt verebileceğini göstermiştir. Ahmed ve diğ., (2019) tarafından yapılan çalışmada belirtilen bir kampüs ağında saniyede en fazla 800 DNS sorgusuna yanıt verildiği düşünülürse, önerilen sistemin kampüs ağları gibi DNS trafiği büyüklüğüne sahip ağlarda yeterince hızlı cevap verebileceği görülmektedir.

KAYNAKLAR

Ahmed J., Gharakheili H. H., Raza Q., Russell C., Sivaraman V., Monitoring Enterprise DNS Queries for Detecting Data Exfiltration from Internal Hosts, *IEEE Transactions on Network and Service Management*, 2019, **17(1)**, 265-279.

Aiello M., Mongelli M., Papaleo G., Basic Classifiers for DNS Tunneling Detection, *IEEE Symposium on Computers and Communications*, Split, Crotia, 7-10 July 2013.

Aiello M., Mongelli M., Papaleo G., DNS Tunneling Detection through Statistical Fingerprints of Protocol Messages and Machine Learning, *International Journal of Communication Systems*, 2015, **28(14)**, 1987-2002.

Aiello M., Mongelli M., Muselli M., Verda D., Unsupervised Learning and Rule Extraction for Domain Name Server Tunneling Detection, *Internet Technology Letters*, 2019, **2(2)**, 1-6.

Albon C., *Python machine learning cookbook*, 1st ed., O'Reilly Media, California, 2018.

Almusawi A., Amintoosi H., DNS Tunneling Detection Method Based on Multilabel Support Vector Machine, *Security and Communication Networks*, 2018, 6137098, 1-9.

Alpaydın E., *Introduction to Machine Learning*, 2nd ed., MIT press, London, 2010.

Al-Kasassbeh M., Khairallah T., Winning Tactics with DNS Tunnelling, *Network Security*, 2019, **2019(12)**, 12-19.

Al-Qatf M., Lasheng Y., Al-Habib M., Al-Sabahi K., Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection, *IEEE Access*, 2018, **6**, 52843-52856.

Bace R., Mell P., Intrusion Detection Systems, *National Institute of Standards and Technology*, 800-31, 1-51, 2001.

Basnet R., Mukkamala S., Sung A. H., Detection of Phishing Attacks: A Machine Learning Approach, Editor: Prasad B., *Fuzziness and Soft Computing*, Springer, Berlin, 373-383, 2008.

Benton J. C., Global Information Assurance Certification Paper, SANS Institute, 2.0, 1-86, 2003.

Bergstra J., Bardenet R., Bengio Y., Kegl B., Algorithms for Hyper-Parameter Optimization, *Advances in neural information processing systems*, **24**, 2546-2554, 2011.

Berry M. W., Mohamed A., *Supervised and Unsupervised Learning for Data Science*, 1st ed., Springer, Arkansas, 2019.

Biju J. M., Gopal N., Prakash A. J., Cyber Attacks and Its Different Types, *International Research Journal of Engineering and Technology*, 2019, **6(3)**, 4849-4852.

Bircanoğlu C., Arıca N., A Comparison of Activation Functions in Artificial Neural Networks, *26th Signal Processing and Communications Applications Conference*, İzmir, Türkiye, 2-5 May 2018.

Blagus R., Lusa L., Gradient Boosting for High-Dimensional Prediction of Rare Events, *Computational Statistics & Data Analysis*, 2017, 113, 19-37.

Bollegala D., Dynamic Feature Scaling for Online Learning of Binary Classifiers, *Knowledge-Based Systems*, 2017, 129, 97-105.

Bubnov Y., DNS Tunneling Detection Using Feedforward Neural Network, *European Journal of Engineering Research and Science*, 2018, **3(11)**, 16-19.

Buczak A. L., Hanke P. A., Cancro G. J., Toma M. K., Watkins L. A., Chavis J. S., Detection of Tunnels in PCAP Data by Random Forests, *11th Annual Cyber and Information Security Research Conference*, New York, United States, 5-7 April 2016.

Burkov A., *The hundred-page machine learning book*, 1st ed., Andriy Burkov, Canada, 2019.

Callegati F., Cerroni W., Ramilli M., Man-in-the-Middle Attack to the HTTPS Protocol, *IEEE Security & Privacy*, 2009, **7(1)**, 78-81.

Cambiaso E., Aiello M., Mongelli M., Papaleo G., Feature Transformation and Mutual Information for DNS Tunneling Analysis, *Eighth International Conference on Ubiquitous and Future Networks*, Vienna, Austria, 5-8 July 2016.

Chandramouli R., Rose S., Secure domain name system (DNS) deployment guide, *National Institute of Standards and Technology*, 800-81-2, 1-130, 2013.

Chauhan P., Chandra N., A Review on Hybrid Intrusion Detection System using Artificial Immune System Approaches, *International Journal of Computer Applications*, 2013, **68(20)**, 22-27.

Cutler A., Cutler D. R., Stevens J. R., Random Forests, Editors: Zhang C., Ma Y., *Ensemble Machine Learning*, Springer, Boston, 157-175, 2012.

Dhanabal L., Shantharajah S. P., A study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms, *International Journal of Advanced Research in Computer and Communication Engineering*, 2015, **4(6)**, 446-452.

Do V.T., Engelstad P., Feng B., Do V.T., Detection of DNS Tunneling in Mobile Networks Using Machine Learning, Editors: Kim K., Joukov N., *Information Science and Applications*, Springer, Singapore, 221-230, 2017.

Erwianda M. S. F., Kusumawardani S. S., Santosa P. I., Rimadana M. R., Improving Confusion-State Classifier Model Using XGBoost and Tree-Structured Parzen Estimator, *2019 International Seminar on Research of Information Technology and Intelligent Systems*, Yogyakarta, Indonesia, 5-6 December 2019.

Farnham G., Atlasis A., Detecting DNS tunneling, *SANS Institute InfoSec Reading Room*, 9, 1-32, 2013.

Feng Y., Liu L., Shu J., A Link Quality Prediction Method for Wireless Sensor Networks Based on XGBoost, *IEEE Access*, 2019, **7**, 155229-155241.

Gangan S., A Review of Man-in-the-Middle Attacks, *Cornell University*, 1504.02115, 1-12, 2015.

Gaddam R., Nandhini M., An Analysis of Various Snort Based Techniques to Detect and Prevent Intrusions in Networks Proposal with Code Refactoring Snort Tool in Kali Linux Environment, *2017 International Conference on Inventive Communication and Computational Technologies*, Coimbatore, India, 10-11 March 2017.

Gao Y., Liu Y., Jin Y., Chen J., Wu H., A Novel Semi-Supervised Learning Approach for Network Intrusion Detection on Cloud-Based Robotic System, *IEEE Access*, 2018, **6**, 50927-50938.

Gupta T. K., Raza K., Optimizing Deep Feedforward Neural Network Architecture: A Tabu Search Based Approach, *Neural Processing Letters*, 2020, **51**, 2855-2870.

Haggag M., Tantawy M. M., El-Soudani M. M., Implementing a Deep Learning Model for Intrusion Detection on Apache Spark Platform, *IEEE Access*, 2020, **8**, 163660-163672.

Hamori S., Kawai M., Kume T., Murakami Y., Watanabe C, Ensemble learning or deep learning? Application to default risk analysis, *Journal of Risk and Financial Management*, 2018, **11(1)**, 12.

Han J., Kamber M., Pei J., *Data Mining Concepts and Techniques*, 3rd ed., Elsevier, Waltham, USA, 2011.

Hangal S., Narayanan S., Chandra N., Chakravorty, S., Iodine: A Tool to Automatically Infer Dynamic Invariants for Hardware Designs, *42nd Design Automation Conference*, California USA, 13-17 June 2005.

Herrero A, Corchado E., *Mobile Hybrid Intrusion Detection*, Springer, Berlin, Germany, 2011.

Hodo E., Bellekens X., Hamilton A., Tachtatzis C., Atkinson R., Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey, *Cornell University*, 1701.02145, 1-43, 2017.

Homem I., Papapetrou P., Dosis S., Entropy-based Prediction of Network Protocols in the Forensic Analysis of DNS Tunnels, *Cornell University*, 1709.06363, 1-6, 2017.

Hou H., Xu Y., Chen M., Liu Z., Guo W., Gao M., Xin Y., Cui L., Hierarchical Long Short-Term Memory Network for Cyberattack Detection, *IEEE Access*, 2020, **8**, 90907-90913.

Hu Z., Wang L., Qi L., Li Y., Yang W., A Novel Wireless Network Intrusion Detection Method Based on Adaptive Synthetic Sampling and an Improved Convolutional Neural Network, *IEEE Access*, 2020, **8**, 195741-195751.

Huang J., Li Y. F., Xie M., An Empirical Analysis of Data Preprocessing for Machine Learning-based Software Cost Estimation, *Information and Software Technology*, 2015, **67**, 108-127.

Ieracitano C., Adeel A., Morabito F. C., Hussain A., A Novel Statistical Analysis and Autoencoder Driven Intelligent Intrusion Detection Approach, *Neurocomputing*, 2020, **387**, 51-62.

Imran M., Alsuhaibani S. A., A Neuro-Fuzzy Inference Model for Diabetic Retinopathy Classification, Editors: Hemanth D. J., Gupta D., Balas V. E., *Intelligent Data Analysis for Biomedical Applications*, 1st ed., Academic Press, Netherlands, 147-172, 2019.

Ioffe S., Szegedy C., Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *32nd International Conference on Machine Learning*, Lille, France, 7-9 July 2015.

Jiang K., Wang W., Wang A., Wu H., Network Intrusion Detection Combined Hybrid Sampling with Deep Hierarchical Network, *IEEE Access*, 2020, **8**, 32464-32476.

Jiang Y., Tong G., Yin H., Xiong N., A Pedestrian Detection Method Based on Genetic Algorithm for Optimize Xgboost Training Parameters, *IEEE Access*, 2019, **7**, 118310-118321.

Karadeniz M., Yüncü S., Aydemir M. T., Asenkron Motorlarda Stator Direncinin Yapay Sinir Ağları ile Tahmini, *Elektrik-Elektronik-Bilgisayar Mühendisliği 9. Ulusal Kongresi*, Kocaeli, Türkiye, 19-21 Eylül 2001.

Kasongo S. M., Sun Y., A Deep Learning Method with Filter Based Feature Engineering for Wireless Intrusion Detection System, *IEEE Access*, 2019, **7**, 38597-38607.

Kemalis K., Tzouramanis T., SQL-IDS: A Specification-Based Approach for SQL-Injection Detection, *2008 ACM symposium on Applied computing*, Fortaleza, Brazil, 16-20 March 2008.

Kemp S., Digital 2019: Global Digital Overview, *Datareportal*, https://datareportal.com/reports/digital-2019-global-digital-overview, (Ziyaret Tarihi: 10 Nisan 2021).

Kubat M., *An Introduction to Machine Learning*, 2nd ed., Springer, Cham, 2015.

Kızrak A., Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması, Medium, https://ayyucekizrak.medium.com/derin-ogrenme-icin-aktivasyon-fonksiyonlarinin -karsilastirilmasi, (Ziyaret Tarihi: 10 Nisan 2021).

Latah M., Toker L., An Efficient Flow-Based Multi-Level Hybrid Intrusion Detection System for Software-Defined Networks, *CCF Transactions on Networking*, 2020, **3**, 261-271.

Li N., Li B., Gao L., Transient Stability Assessment of Power System Based on XGBoost and Factorization Machine, *IEEE Access*, 2020, **8**, 28403-28414.

Li Y., Xu Y., Liu Z., Hou H., Zheng Y., Xin Y., Zhao Y., Cui L., Robust Detection for Network Intrusion of Industrial Iot Based on Multi-CNN Fusion, *Measurement*, 2020, **154**, 107450.

Liu H., Lang B., Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey, *Applied Sciences*, 2019, **9(20)**, 4396.

Liu J., Li S., Zhang Y., Xiao J., Chang P., Peng C., Detecting DNS Tunnel through Binary-Classification Based on Behavior Features, *16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Sydney, Australia, 1-4 August 2017.

Liu W., Research on Dos Attack and Detection Programming, *3rd International Symposium on Intelligent Information Technology Application*, Nanchang, China, 21-22 November 2009.

Liu X., Di X., Ding Q., Liu W., Qi H., Li J., Yang H., NADS-RA: Network Anomaly Detection Scheme Based on feature Representation and data Augmentation, *IEEE Access*, 2020, **8**, 214781-214800.

Meng W., Tischhauser E. W., Wang Q., Wang Y., Han J., When Intrusion Detection Meets Blockchain Technology: A Review, *IEEE Access*, 2018, **6**, 10179-10188.

Merlo A., Papaleo G., Veneziano S., Aiello M., A Comparative Performance Evaluation of DNS Tunneling Tools, Editors: Herrero A., Corchado E., *Computational Intelligence in Security for Information Systems*, Springer, Berlin, 84-91, 2011.

Mohri M., Rostamizadeh A., Talwalkar A., *Foundations of Machine Learning*, 1st ed., MIT press, London, 2012.

Mukherjee S., Sharma N., Intrusion Detection Using Naive Bayes Classifier with Feature Reduction, *Procedia Technology*, 2012, **4**, 119-128.

Müller A. C., Guido S., *Introduction to Machine Learning with Python*, 1st ed., O'Reilly Media, Massachusetts, 2016.

Nadler A., Aminov A., Shabtai A., Detection of Malicious and Low Throughput Data Exfiltration over the DNS Protocol, *Computers & Security*, 2019, **80**, 36-53.

Othman S. M., Alsohybe N. T., Ba-Alwi F. M., Zahary A. T., Survey on Intrusion Detection System Types, *International Journal of Cyber-Security and Digital Forensics*, 2018, **7**, 444-463.

Önal H., DNS Tünelleme, EnderUNIX, http://www.enderunix.org/docs/dns_tunellem e.pdf, (Ziyaret Tarihi: 10 Nisan 2021).

Paliwal S., Gupta R., Denial-of-service, Probing & Remote to User (R2L) Attack Detection Using Genetic Algorithm, *International Journal of Computer Applications*, 2012, **60(19)**, 57-62.

Rao D., McMahan B., *Natural Language Processing with Pytorch*, 1st ed., O'Reilly Media, Massachusetts, 2019.

Raschka S., Mirjalili V., *Python Machine Learning*, 2nd ed., Puckt Publishing, Birmingham, 2017.

Rathore S., Park J. H., Semi-Supervised Learning Based Distributed Attack Detection Framework for Iot, *Applied Soft Computing*, 2018, **72**, 79-89.

Rohith R., Moharir M., Shobha G., SCAPY-A powerful interactive packet manipulation program, *2018 International Conference on Networking, Embedded and Wireless Systems*, Bangalore, India, 27-28 December 2018.

Saleem M. H., Potgieter J., Arif K. M., Plant Disease Classification: A Comparative Evaluation of Convolutional Neural Networks and Deep Learning Optimizers, *Plants*, 2020, **9(10)**, 1319-1335.

Sammour M., Hussin B., Othman F. I., Comparative Analysis for Detecting DNS Tunneling Using Machine Learning Techniques, *International Journal of Applied Engineering Research*, 2017, **12(22)**, 12762-12766.

Saxena A. K., Sinha S., Shukla P., General Study of Intrusion Detection System And Survey Of Agent Based Intrusion Detection System, *2017 International Conference on Computing, Communication and Automation*, Greater Noida, India, 5-6 May 2017.

Sazlı M. H., A Brief Review of Feed-Forward Neural Networks, *Communications Faculty of Science University of Ankara*, 2006, **50(1)**, 11-17.

Seyyarer E., Uçkan T., Hark C., Ayata F., İnan M., Karci A., Applications and Comparisons of Optimization Algorithms Used in Convolutional Neural Networks, *International Artificial Intelligence and Data Processing Symposium*, Malatya, Türkiye, 21-22 September 2019.

Sharma S., Sharma S., Athaiya A., Activation Functions in Neural Networks, *International Journal of Engineering Applied Sciences and Technology*, 2020, **4(12)**, 310-316.

Sen J., Mehtab S., Machine Learning Applications in Misuse and Anomaly Detection, *Cornell University*, 2009.06709, 1-21, 2020.

Singh K., Singh P., Kumar K., Application Layer HTTP-GET Flood Ddos Attacks: Research Landscape and Challenges, *Computers & Security*, 2017, **65**, 344-372.

Skow T. K., Protection Against DNS Tunneling Abuses on Mobile Networks, Master Thesis, Norwegian University, Department of Telematics, Trondheim, 2016.

Soydaner D., A Comparison of Optimization Algorithms for Deep Learning, *Cornell University*, 2007.14166, 1-26, 2020.

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *The journal of machine learning research*, 2014, **15(1)**, 1929-1958.

Su T., Sun H., Zhu J., Wang S., Li Y., BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset, *IEEE Access*, 2020, **8**, 29575-29585.

Sundaram A., An Introduction to Intrusion Detection, *Crossroads*, 1996, **2(4)**, 3-7.

Tama B. A., Rhee K. H., An Extensive Empirical Evaluation of Classifier Ensembles for Intrusion Detection Task, *Computer Systems Science and Engineering*, 2017, **32(2)**, 149-158.

Tama B. A., Comuzzi M., Rhee, K. H., TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-based Intrusion Detection System, *IEEE Access*, 2019, **7**, 94497-94507.

Tanyıldızı E., Demirtaş F., Hyper Parameter Optimization, *1st International Informatics and Software Engineering Conference*, Ankara, Turkey, 6-7 November 2019.

Tavallaee M., Bagheri E., Lu W., Ghorbani A. A., A Detailed Analysis of The KDD CUP 99 Dataset. *2009 IEEE symposium on computational intelligence for security and defense applications*, Ottawa, Canada, 8-10 July 2009.

Tchakoucht T. A., Ezziyyani M., Multilayered Echo-State Machine: A Novel Architecture for Efficient Intrusion Detection, *IEEE Access*, 2018, **6**, 72458-72468.

Tomak L., Yüksel B. E. K., İşlem karakteristik eğrisi analizi ve eğri altında kalan alanların karşılaştırılması, *Journal of Experimental and Clinical Medicine*, 2009, **27(2)**, 58-65.

Tsangaratos P., Ilia I., Comparison of a Logistic Regression and Naive Bayes Classifier in Landslide Susceptibility Assessments: The Influence of Models Complexity and Training Dataset Size, *Catena*, 2016, **145**, 164-179.

URL-1: https://en.wikipedia.org/wiki/Cyberattack, (Ziyaret Tarihi: 10 Nisan 2021).

URL-2: https://phoenixnap.com/blog/man-in-the-middle-attacks-prevention, (Ziyaret Tarihi: 10 Nisan 2021).

URL-3: https://www.ipa.go.jp/security/fy11/report/contents/intrusion/ids-meeting/ids bg.pdf, (Ziyaret Tarihi: 10 Nisan 2021).

URL-4: http://sci.tamucc.edu/~cams/projects/320.pdf, (Ziyaret Tarihi: 10 Nisan 2021).

URL-5: http://www.otnira.com/2013/03/25/c4-5/, (Ziyaret Tarihi: 10 Nisan 2021).

URL-6: https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4, (Ziyaret Tarihi: 10 Nisan 2021).

URL-7: https://www.alexa.com/topsites, (Ziyaret Tarihi: 10 Nisan 2021).

Vasilev I., Slater D., Spacagna G., Roelants P., Zocca V., *Python Deep Learning*, 2nd ed., Puckt Publishing, Birmingham, 2019.

Verdhan V., *Computer Vision Using Deep Learning*, 1st ed., Apress, New York City, 2021.

Wang B., Lu K., Chang P., Design and Implementation of Linux Firewall Based on the Frame of Netfilter/Iptable, *11th International Conference on Computer Science & Education*, Nagoya, Japan, 23-25 August 2016.

Wang M., Zheng K., Yang Y., Wang X., An Explainable Machine Learning Framework for Intrusion Detection Systems, *IEEE Access*, 2020, **8**, 73127-73141.

Wu K., Chen Z., Li W, A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks, *IEEE Access*, 2018, **6**, 50850-50859.

Wu Y., Lee W. W., Xu Z., Ni M., Large-Scale and Robust Intrusion Detection Model Combining Improved Deep Belief Network with Feature-Weighted SVM, *IEEE Access*, 2020, **8**, 98600 – 98611.

Xuan L. F., Wu P. F., The Optimization and Implementation of Iptables Rules Set on Linux, *2nd International Conference on Information Science and Control Engineering*, Shanghai, China, 24-26 April 2015.

Xue W., Wu T., Active Learning-Based XGBoost for Cyber Physical System Against Generic AC False Data Injection Attacks, *IEEE Access*, 2020, **8**, 144575-144584.

Yan X., Jia M., A Novel Optimized SVM Classification Algorithm with Multi-domain Feature and Its Application to Fault Diagnosis of Rolling Bearing, *Neurocomputing*, 2018, 313, 47-64.

Yang Y., Zheng K., Wu C., Niu X., Yang Y., Building an Effective Intrusion Detection System Using the Modified Density Peak Clustering Algorithm and Deep Belief Networks, *Applied Sciences*, 2019, **9(2)**, 238.

Yassine S., Khalife J., Chamoun M., El Ghor H., A Survey of DNS Tunnelling Detection Techniques Using Machine Learning, *1st International Conference on Big Data and Cyber-Security Intelligence*, Beirut, Lebanon, 13-15 December 2018.

KİŞİSEL YAYIN VE ESERLER

Altuncu M. A., Kaya Gülağız F., Özcan H., Bayır Ö. F., Gezgin A., Niyazov A., Çavuşlu M. A., Şahin S., Deep Learning Based DNS Tunneling Detection and Blocking System, *Advances in Electrical and Computer Engineering,* 2021, **21(3)**. (Accepted)

Altuncu M. A., Kaya Gülağız F., Özcan H., İlkin S., Şahin S., Simulation of Academic Computer Networks Using Probability Distributions: A Case Study in A Campus Network, *Tehnicki Vjesnik*, 2021, **28(3)**, 1017-1024, DOI: 10.17559/TV-20200224091110.

Altuncu M. A., Türkoğlu B., Çavuşlu M. A., Şahin S., Implementation of K-means algorithm on FPGA, *26th IEEE Signal Processing and Communications Applications Conference*, İzmir, Turkey, 2-5 May 2018.

Altuncu M. A., Kösten M. M., Çavuşlu M. A., Şahin S., FPGA-Based Implementation of Basic Image Processing Applications as Low-Cost IP Core, *26th IEEE Signal Processing and Communications Applications Conference*, İzmir, Turkey, 2-5 May 2018.

Altuncu M. A., Hangişi F. S., Kaya Gülağız F., Şahin S., Performance Analysis of Image Restoration Techniques for Dermoscopy Images, *International Journal of Applied Information Systems*, 2017, **11**, 15-19.

Altuncu M. A., Kaya Gülağız F., Bir T., Özcan H., Şahin S., Imputation of Missing Data for Network Intrusion Detection, *IOSR Journal of Computer Engineering,* 2017, **19**, 8-12.

Altuncu M. A., Güven T., Becerikli Y., Şahin S., Real Time System Implementation for Image Processing with Hardware Software Co design on the Xilinx Zynq Platform, *International Journal of Information and Electronics Engineering*, 2015, **5**, 473-477.

Çavuşlu M. A., **Altuncu M. A.**, Özcan H., Kaya Gülağız F., Şahin S., Estimation of underwater acoustic channel parameters for Erdek/Turkey region, *Applied Acoustics*, 2021, 181, DOI: 10.1016/j.apacoust.2021.108135.

Çavuşlu M. A., **Altuncu M. A.**, Özcan H., Kaya Gülağız F., Sahin S., Sualtı Haberleşmede Çok Yolluluğun Bant Genişliği, Kapasite ve İletim Gücü Üzerindeki Etkisi, *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi,* 2020, **7(1)**, 404-420.

İlkin S., Gençtürk T. H., Kaya Gülağız F., Özcan H., **Altuncu M. A.**, Şahin S., hybSVM: Bacterial Colony Optimization Algorithm Based SVM for Malignant Melanoma Detection, *Engineering Science and Technology, an International Journal*, 2021, **24(5)**, 1059-1071, DOI: 10.1016/j.jestch.2021.02.002.

Özcan H., Kaya Gülağız F., **Altuncu M. A.**, İlkin S., Şahin S., A New Visual Cryptography Method based on the Profile Hidden Markov Model, *Advances in Electrical and Computer Engineering,* 2021, **21(1)**, 21-36, DOI:10.4316/AECE.2021.01003.

Özcan H., Kaya Gülağız F., **Altuncu M. A.**, Kaya S., Topuz G., Şahin S., A Leap Motion Based Mobile Game Design for Developing Hand and Wrist Movement in Children, *4th International Symposium on Innovative Approaches in Engineering and Natural Sciences,* Samsun, Turkey, 22-24 November 2019.

Özcan H., **Altuncu M. A.**, Küçük K., Şahin S., Simulation of Indoor Positioning System Based on Radio Frequency, *IOSR Journal of Computer Engineering,* 2017, **19**, 13-18.

Özcan H., Güven T., **Altuncu M. A.**, Kır Savaş B., Şahin S., Duman O., Desıgn and Implementatıon of a Content Delıvery Archıtecture for Museums, *International Conference on Research in Education and Science,* Kuşadası, Turkey, 19-21 May 2017.

Pınar Z., Kaya Gülağız F., **Altuncu M. A.**, Şahin S., Denim Kumaşlarda Görüntü İşleme İle Hata Tespiti, *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, 2020, **9(4)**, 1609-1620.

ÖZGEÇMİŞ

Mehmet Ali Altuncu ilk, orta ve lise öğrenimini Siirt'te tamamladı. 2006 yılında girdiği Sakarya Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nden 2010 yılında mezun oldu. 2012 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bilgisayar Bilimleri Anabilim Dalı'ndaki Yüksek Lisans eğitimini 2015 yılında tamamladı. 2015 yılından beri Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda doktora eğitimine devam etmektedir. Aynı zamanda 2012 yılından itibaren Kocaeli Üniversitesi Bilgisayar Mühendisliği bölümünde ÖYP kapsamında Araştırma Görevlisi olarak çalışmaktadır.