T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANARAK WEB GÜNLÜK KAYITLARINDAN ZARARLI İSTEKLERİN TESPİTİ VE BULUT TABANLI BİR SİSTEM GERÇEKLENMESİ

Salih ŞAHİN

DANIŞMAN
Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Ocak, 2023

TEZ KABUL VE ONAYI

Salih ŞAHİN tarafından, **Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN** danışmanlığında hazırlanan **"MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANARAK WEB GÜNLÜK KAYITLARINDAN ZARARLI İSTEKLERİN TESPİTİ VE BULUT TABANLI BİR SİSTEM GERÇEKLENMESİ"** başlıklı bu çalışma, jürimiz tarafından **20/01/2023** tarihinde yapılan sınav sonucunda **oy birliği** ile başarılı bulunarak **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Tez Jürisi

		İmza	**Sonuç**
DANIŞMAN	Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN İstanbul Üniversitesi-Cerrahpaşa Siber Güvenlik Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Dr. Öğr. Üyesi Özgür Can TURNA İstanbul Üniversitesi-Cerrahpaşa Siber Güvenlik Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Can EYÜPOĞLU Milli Savunma Üniversitesi Donanım Anabilim Dalı		☒ Kabul ☐ Ret

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve bilimsel etik kuralları içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını ve her türlü hukuki sorumluluğu aldığımı kabul ederim.

Salih ŞAHİN

Üzerimde emeği olan aileme ithaf ediyorum…

BÜTÇE DESTEKLERİ

MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANARAK WEB GÜNLÜK KAYITLARINDAN ZARARLI İSTEKLERİN TESPİTİ VE BULUT TABANLI BİR SİSTEM GERÇEKLENMESİ

Bu tez çalışması için herhangi bir kurumdan bütçe desteği alınmamıştır.

TEŞEKKÜR

Tez süresi boyunca yardımını eksik etmeyen ve yönlendirmeleri ile bu süreci tamamlamamda yardımcı olan sayın hocam Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN'a ve yüksek lisans eğitimim boyunca üzerimde emeği olan Sayın Doç. Dr. Muhammed Ali AYDIN hocama teşekkürü borç bilirim.

Üzerimde emeği çok fazla olan annem Elfida ŞAHİN'e ve babam Hidayet ŞAHİN'e sonsuz teşekkür ederim. Desteğini hep hissettiğim, eğitim hayatım boyunca yanımda olan desteğini esirgemeyen ağabeyim Ramazan ŞAHİN'e teşekkür ederim. İşimden ve eğitimimden dolayı sürekli yoğunluğum olmasına rağmen anlayış gösterip her aşamada yardımını ve desteğini eksik etmeyen eşim Fatma ŞAHİN'e teşekkür ederim.

Sayın okul müdürüm Ayşegül TURAN'a ve mesai arkadaşım Ümit BEYAZŞEKEROĞLU'na desteklerinden dolayı teşekkür ederim.

Ocak 2023 **Salih ŞAHİN**

İÇİNDEKİLER

Sayfa No

TEZ KABUL VE ONAYI ..... **ii**
BEYAN ..... **iii**
BÜTÇE DESTEKLERİ ..... **v**
TEŞEKKÜR ..... **vi**
İÇİNDEKİLER ..... **vii**
ŞEKİL LİSTESİ ..... **x**
TABLO LİSTESİ ..... **xii**
SİMGE VE KISALTMA LİSTESİ ..... **xiii**
ÖZET ..... **xv**
ABSTRACT ..... **xvii**
1. GİRİŞ **1**
2. KAVRAMSAL ÇERÇEVE **3**
2.1. WEB EKOSİSTEMİ 3
2.1.1. İstemci (Client) 3
2.1.2. Sunucu (Server) 3
2.1.3. İstek (Request) 3
2.1.4. Cevap (Response) 3
2.1.5. HTTP ve HTTPS 4
2.1.6. SSL 4
2.1.7. Temel HTTP Yöntemleri 4
2.1.8. Web Sunucu 5
2.1.9. Veri Tabanı Yönetim Sistemleri 5
2.1.10. SQL 5
2.2. WEB SALDIRI TEKNİKLERİ 6
2.2.1. Bozuk Erişim Kontrolü (A01:2021-Broken Access Control) 6
2.2.2. Kriptolojik Hatalar (A02:2021-Cryptographic Failures) 7
2.2.3. Enjeksiyonlar (A03:2021-Injection) 7
2.2.4. Güvensiz Tasarım (A04:2021-Insecure Design) 7

2.2.5. Yanlış Güvenlik Yapılandırması (A05:2021-Security Misconfiguration) 7
2.2.6. Zafiyetli ve Güncel Olmayan Bileşenler (A06:2021-Vulnerable and Outdated Components) 7
2.2.7. Tanımlama ve Doğrulama Hataları (A07:2021-Identification and Authentication Failures) 8
2.2.8. Yazılım ve Veri Bütünlüğü Hataları (A08:2021-Software and Data Integrity Failures) 8
2.2.9. Güvenlik Loglaması ve İzleme Hataları (A09:2021-Security Logging and Monitoring Failures) 8
2.2.10. Sunucu Taraflı İstek Sahteciliği (A10:2021-Server-Side Request Forgery) 8
2.3. YAPAY ZEKÂ 9
2.4. MAKİNE ÖĞRENMESİ 9
2.4.1. Destek Vektör Makinesi (Support Vector Machine - SVM) 10
2.4.2. Karar Ağaçları (Decision Tree - DT) 10
2.4.3. Rastgele Orman (Random Forest - RF) 11
2.4.4. Lojistik Regresyon (Logistic Regression - LR) 12
2.4.5. Yapay Sinir Ağları (Artificial Neural Networks - ANNs) 13
2.5. DERİN ÖĞRENME (DEEP LEARNING - DL) 15
2.5.1. Evrişimli Sinir Ağları (Convolutional Neural Networks - CNNs) 16
2.5.2. Tekrarlayan Sinir Ağları (RNNs) 27
2.5.3. Kapsül Ağlar 27
2.6. MEVCUT ÇALIŞMALAR 27
3. YÖNTEM 32
3.1. KAPSÜL AĞLAR 32
3.1.1. Temel Yapı 32
3.1.2. Ezme İşlemi 34
3.1.3. Kayıp Fonksiyon 35
3.2. Veri Seti 35
3.3. Veri Ön İşleme 38
3.4. Model Eğitimi 39
3.5. Uygulamanın Gerçekleştirilmesi 40
3.5.1. Bulut Hizmetler 41
3.5.2. Apache Kafka ve KRaft 41
3.5.3. Docker 42
3.5.4. Ortamın Hazırlanması 43

4. BULGULAR 47
4.1. Veri Ön İşleme 47
4.2. Eğitim Bulguları 48
4.2.1. Klasik Makine Öğrenmesi Algoritmaları Bulguları 48
4.2.2. Kapsül Ağlar Bulguları 49
4.3. Uygulama Sonuçları 52
5. TARTIŞMA 54
6. SONUÇ VE ÖNERİLER 56
KAYNAKLAR 57
EKLER 63
İNTİHAL RAPORU İLK SAYFASI 68
ÖZGEÇMİŞ 69

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1: İstemci-Sunucu Mimarisi 4
Şekil 2.2: OWASP Top 10 Web Uygulaması Güvenlik Riskleri [11] 6
Şekil 2.3: Makine Öğrenmesi Algoritmaları 9
Şekil 2.4: 2 Sınıflı Destek Vektör Makinesi 10
Şekil 2.5 Karar Ağacı Örneği 11
Şekil 2.6: Rastgele Orman Örneği 11
Şekil 2.7: Lojistik Regresyon 12
Şekil 2.8: Biyolojik Nöron 14
Şekil 2.9: Yapay Nöron Modeli 14
Şekil 2.10: Yapay Sinir Ağı Örneği 15
Şekil 2.11: Yapay Zekâ, Makine Öğrenmesi ve Derin Öğrenme Arasında İlişki [28] 16
Şekil 2.12: Evrişimli Sinir Ağı Örneği [30] 16
Şekil 2.13: Evrişim Katmanı İşlem Adımı [31] 17
Şekil 2.14: Aktivasyon Fonksiyonları 18
Şekil 2.15: Havuzlama Yöntemlerinden En Çok Kullanılan 3 Yöntemin Örneği [28] 19
Şekil 2.16: Doldurma İşlemi 24
Şekil 2.17: Seyreltme İşlemi Öncesi ve Sonrası 25
Şekil 2.18: 1 Adımlı Kaydırma İşlemi 26
Şekil 2.19: Basit Bir Tekrarlayan Sinir Ağı [34] 27
Şekil 3.1: Eğitim Aşaması 32
Şekil 3.2: CNN için benzer olan iki resim [55] 33
Şekil 3.3: Temel Kapsül Ağ [35] 33

Şekil 3.4: CapsNet Tam Bağlantılı Katman [35] 34
Şekil 3.5: Zararsız Bir Get İsteği 36
Şekil 3.6: Zararlı Bir Get İsteği 36
Şekil 3.7: Zararsız Bir Post İsteği 37
Şekil 3.8: Zararsız Bir Post İsteği 37
Şekil 3.9: Eğitim Test Süreci Adımları 40
Şekil 3.10: Uygulanan Sistemin Akış Şeması 41
Şekil 3.11: Zookeeper ve Kafka ile KRaft Yapısı [61] 42
Şekil 3.12: Sanal Makine ve Konteyner Yapısı 43
Şekil 3.13: Uygulama Ortamı 44
Şekil 3.14: Web Uygulamasının Çalışır Durumdaki Ekran Görüntüsü 44
Şekil 3.15: KRaft Kullanıcı Ara Birimi 45
Şekil 3.16: Güvenlik Olay Merkezi (SOC) Paneli 46
Şekil 4.1: Veri setine ait kelime bulutu 48
Şekil 4.2: Doğruluk Grafiği 50
Şekil 4.3: Kayıp Grafiği 50
Şekil 4.4:Test Karmaşıklık Matrisi 51
Şekil 4.5: Uygulama Sonucu Oluşan Karmaşıklık Matrisi 53

TABLO LİSTESİ

Sayfa No

Tablo 2.1: Temel SQL Komutları [9] 6

Tablo 2.2: Veri Seti ya da Kapsül Ağ ile ilgili çalışmalar 31

Tablo 3.1: Veri Seti Dosya Temel Bilgileri 36

Tablo 3.2: Veri Setinin İstatistiksel Değerlendirmesi 38

Tablo 4.1: Makine Öğrenmesi Yöntemlerine Ait Sonuçlar 49

Tablo 4.2: Eğitim Ortamları ve Ölçüm Sonuçları 49

Tablo 4.3: Sklearn Ölçüm Metrikleri 51

Tablo 4.4: Uygulama Sonuç Metrikleri 53

Tablo 5.1: Model Sonuçlarının Karşılaştırma Tablosu 55

SİMGE VE KISALTMA LİSTESİ

Simgeler **Açıklama**

$\boldsymbol{\eta}$: Öğrenme oranı (Eta)
$\sum$: Toplam Sembolü
$\boldsymbol{\partial}$: Kısmi Türev
γ : Momentum Değeri (Gama)
ϵ : Epsilon

Kısaltmalar **Açıklama**

AI : Yapay Zekâ (Artificial Intelligence)
ANNs : Yapay Sinir Ağları (Artificial Neural Networks)
API : Uygulama Programlama Arayüzü (Application Programming Interface)
BGD : Toplu Gradyan İniş (Batch Gradient Descent)
CDN : İçerik Dağıtım Ağı (Content Delivery Network)
CIA : Gizlilik Bütünlük Erişebilirlik (Confidentiality Integrity Availability)
CNN : Evrişimli Sinir Ağları (Convolutional Neural Network)
CPU : Merkezî İşlem Birimi (Central Process Unit)
CSRF : Siteler Arası İstek Saldırıları (Cross-Site Request Forgery)
DBMS : Veri Tabanı Yönetim Sistemi (Database Management System)
DL : Derin Öğrenme (Deep Learning)
DT : Karar Ağaçları (Decision Trees)
FI : Dosya Enjeksiyonları (File Include Injection)
GAP : Genel Ortalama Havuzlama (Global Average Pooling)
GB : Gradyan Arttırma (Gradient Boosting)
GPU : Grafik İşlem Birimi (Graphics Processing Unit)
GRU : Kapılı Tekrarlayan Birim (Gated Recurrent Unit)
HTTP : Hiper Metin Transfer Protokolü (Hypertext Transfer Protocol)
HTTPS : Güvenli Hiper Metin Transfer Protokolü (Secure Hypertext Transfer Protocol)

IANA : İnternet Tahsisli Numaralar Kurumu (Internet Assigned Numbers Authority)

IIS : İnternet Bilgi Servisleri (Internet Information Services)

LR : Lojistik Regresyon (Logistic Regression)

LSTM : Uzun Kısa Süreli Bellek (Long Short Term Memory)

ML : Makine Öğrenmesi (Machine Learning)

MLP : Çok Katmanlı Algılayıcı (Multilayer Perceptron)

MV : Çoğunluk Oylama (Majority Voting)

NB : Naive Bayes

NLP : Doğal Dil İşleme (Natural Language Processing)

NN : Sinir Ağı (Neural Network)

OWASP : Açık Web Uygulama Güvenliği Projesi (Open Web Application Security Project)

ReLU : Doğrultulmuş Lineer Birim (Rectified Linear Unit)

RF : Rastgele Orman (Random Forest)

RNNs : Tekrarlayan Sinir Ağları (Recurrent Neural Networks)

SGD : Stokastik Gradyan İniş (Stochastic Gradient Descent)

SOC : Siber Güvenlik Operasyon Merkezi(SOC (Security Operations Center)

SQLi : SQL Enjeksiyonu (SQL Injection)

SSL : Güvenli Soket Katmanı (Secure Sockets Layer)

SSRF : Sunucu Taraflı İstek Sahteciliği (Sever Side Request Forgery)

SVM : Destek Vektör Makinesi (Support Vector Machine)

VM : Sanal Makine (Virtual Machine)

VTYS : Veri Tabanı Yönetim Sistemi (Database Management System)

XSS : Siteler Arası Çapraz Sorgulama (Cross Site Scripting)

ÖZET

YÜKSEK LİSANS TEZİ

MAKİNE ÖĞRENMESİ ALGORİTMALARI KULLANARAK WEB GÜNLÜK KAYITLARINDAN ZARARLI İSTEKLERİN TESPİTİ VE BULUT TABANLI BİR SİSTEM GERÇEKLENMESİ

Salih ŞAHİN

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman: Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN

Günümüzde insanlar zamanlarının önemli bir kısmını İnternet'e bağlı olarak geçirmekte ve birçok işlemi internete bağlı cihazlar ile yapmaktadır. İçinde bulunduğumuz yüzyılda en değerli varlıklardan biri de kişisel bilgilerdir. Kişisel bilgilerin elde edilmesi sonucunda insanlar maddi ve manevi zararlar görebilmektedir. Özellikle geçtiğimiz yıllarda ortaya çıkan Covid-19 pandemisi sürecinde öncesine göre çok daha fazla sayıda işlem İnternet aracığıyla gerçekleştirilmiştir. İnternet ortamında yapılan işlemlerin hacminin çok fazla büyümesiyle ortaya çıkan ve işlenen verinin miktarı da "büyük veri" olarak adlandırılan seviyeye gelmiştir. Tüm bu sebeplerden dolayı bilgisayar sistemleri saldırganların daha fazla ve daha popüler hedefi haline gelmiştir ve siber saldırıların sayıları da buna bağlı olarak artmıştır. Sayıları oldukça artan bu siber saldırıların tespit edilmesi önem arz etmektedir. Özellikle olası bir siber saldırıda hem maddi kayıp hem de itibar kaybı yaşayacak olan firmalar için bu durum elzemdir. Ayrıca büyük verinin oluştuğu alanlarda klasik yöntemlerle sağlıklı veri analizi yapmak mümkün olmamaktadır. Bu tez çalışmasında, büyük veri üzerinden analiz yapma amacıyla

kullanılan derin öğrenme yöntemlerinden biri olan kapsül ağlar tabanlı bir model geliştirilmiştir. Geliştirilen model, CSIC 2010 HTTP veri seti ile eğitilmiş ve test işlemi gerçekleştirilmiştir. Geliştirilen model kullanılarak web günlüklerinin analizi yapılmıştır. Analiz sonucunda isteğin zararlı olup olmadığının tespiti yapılmıştır. Geliştirilen model %97,43 oranında başarı elde etmiştir. Daha sonra çalışan bir sisteme dahil edilmiş ve tespit işlemlerini bu sistem üzerinde gerçekleştirmiştir.

Ocak 2023 , 87 sayfa.

Anahtar kelimeler: Kapsül Ağlar, Log Analizi, Makine Öğrenmesi, Siber Güvenlik

ABSTRACT

M.Sc. THESIS

DETECTION OF MALICIOUS REQUESTS FROM WEB LOG RECORDS USING MACHINE LEARNING ALGORITHMS AND IMPLEMENTATION OF A CLOUD-BASED SYSTEM

Salih ŞAHİN

İstanbul University-Cerrahpaşa

Institute of Graduate Studies

Department of Computer Engineering

Programme of Computer Engineering

Supervisor : Assist. Prof. Dr. Gülsüm Zeynep GÜRKAŞ AYDIN

Today, people spend a significant part of their time connected to the Internet and do many transactions with devices connected to the Internet. One of the most valuable assets in our century is personal information. As a result of obtaining personal information, people may suffer material and moral damages. Especially during the Covid-19 pandemic that emerged in recent years, much more transactions were carried out via the Internet than before. The amount of data that has emerged and been processed with the huge growth of the volume of transactions made on the Internet has reached the "big data" level. For all these reasons, computer systems have become more and more popular targets of attackers, and the number of cyber attacks has increased accordingly. It is essential to detect these cyber attacks, the number of which is increasing considerably. This is essential for companies that will experience both financial and reputational loss in a possible cyber attack. In addition, it is impossible to carry out healthy data analysis with classical methods in areas where big data is formed. In this thesis, a capsule network-based model, one of the deep learning methods used for analyzing big data, has been

developed. The developed model has been trained and tested with the CSIC 2010 HTTP dataset. The web logs were analyzed using the developed model. As a result of the analysis, it has been determined whether the request is harmful or not. The developed model achieved a success rate of 97.43%. Later, it was included in a working system and the detection operations were carried out on this system.

January 2023, 87 pages.

Keywords: Capsule Networks, Log Analysis, Machine Learning, Cyber Security

1. GİRİŞ

Günümüzde teknolojik gelişmeler akıl almaz bir hızda ilerlemektedir ve özellikle bilişim sistemleri göz önüne alındığında bu ilerleme çok daha fazla fark edilmekte ve ön plana çıkmaktadır [1]. Bu gelişimin getirdiği birtakım olumsuzluklar da mevcuttur. Teknolojinin gelişmesi ve insanların birçok işini teknolojik cihazlar ile yapmaya başlaması sebebiyle bilişim sistemleri, siber korsanların hedefleri haline gelmiştir. Siber korsanların günümüze dek gerçekleştirmiş olduğu saldırılarda kullanıcıların milyarlarca kişisel verisi çalınmıştır. Ayrıca IBM'e göre bu veri ihlallerinden dolayı ortaya çıkan ortalama maliyetler 2021 yılında artış göstermiştir [2]. Örneğin, veri ihlalleri sonucunda Türkiye'deki firmaların uğradığı maddi zararların ortalama toplam maliyeti 2021 yılında 1,91 milyon dolar olduğu belirtilmiştir [3]. Bu sayılardan da anlaşılacağı gibi siber saldırıların engellenmesi hem kişisel verilerin açığa çıkmasını hem de maddi zararların oluşmasını engelleyecektir. Günümüzde bilgi teknolojileri sektöründe siber saldırıların çoğunun web sistemlerine yönelik olduğu belirtilmektedir [2]. Web saldırılarının birçok türü bulunmaktadır. SQL Enjeksiyonu (SQLi), XSS Enjeksiyonu, Dosya Enjeksiyonu (FI), Kaba Kuvvet Saldırıları, Siteler Arası İstek Saldırıları (CSRF) vb. saldırılar en yaygın olarak bilinen saldırı türlerine örnek olarak verilebilir [4]. Web sayfalarının güvenliğini sağlamak için birçok yöntem vardır. Web sistemlerinde her davranış için ayrı bir günlük kaydı oluşturulur. Bu günlük kayıtları incelenerek saldırının gerçekleşip gerçekleşmediği anlaşılabilmektedir. Bu çalışmada ise web saldırılarının HTTP günlüklerinin analizi ile tespitine odaklanılmıştır.

Son yıllarda web sitelerinin yoğun kullanımıyla günlük kayıtlarının hacmi çok fazla artmış ve bu sebeple günlük kayıtlarının statik olarak incelenmesi ya çok zor hale gelmiştir ya da mümkün olamamaktadır [5]. Buna çözüm olarak günlük kayıtlarının bilgisayarlar tarafından dinamik yöntemlerle işlenip incelenmesi ve bu işlemler ve incelemeler sonucunda saldırı olup olmadığı bilgisinin sistem yöneticilerine bildirilebilmesi tercih edilen ve istenen yöntem olarak karşımıza çıkmaktadır. Bu yaklaşımda karşımıza çıkan sorun ise tüm saldırıların ya da tüm iyi niyetli isteklerin doğru tespit edilip edilemeyeceğidir. Saldırı tespit yöntemleri temelde 3 grupta incelenebilmektedir: istatistik temelli, bilgi temelli ve yapay zekâ temelli yöntemlerdir. Bunlardan özellikle son yıllarda oldukça popüler hale gelen yöntemlerden biri olan saldırıların

yapay zekâ temelli tespiti ile ilgili birçok çalışma yapılmış, yöntemler geliştirilmiş ve bu geliştirilen yöntemlerle başarılı sonuçlar elde edilmiştir [6]. Bu çalışmaların içinde çoğunlukla web günlüğü analizi için makine öğrenmesi (ML) algoritmaları tabanlı yöntemler kullanılmıştır. Son zamanlarda ise derin öğrenme (DL) yöntemlerinin gelişmesiyle analiz işlemlerinde DL yöntemlerini kullanan çalışmaların sayısında da artış görülmüştür [7].

Bu tez çalışmasında, yapılan araştırma sonucunda web günlüklerinin analizi için daha önce kullanılmadığı tespit edilen derin öğrenme yöntemleri içinde yer alan Kapsül Ağlar'ın (Capsule Networks) kullanılması önerilmektedir.

Bu tez çalışmasının ikinci bölümünde tez kapsamında yer alan konular hakkında temel kavramlar anlatılmıştır ve daha sonra günlük analizi ile ilgili mevcut çalışmalar verilmiştir. Üçüncü bölümde önerilen yöntem ve uygulama hakkında bilgi verilmiştir. Dördüncü bölümde elde edilen bulgular sunulmuştur. Beşinci bölümde tez kapsamında gerçekleştirilen model eğitiminden ve uygulamadan elde edilen sonuçlar tartışılmıştır ve değerlendirmesi yer almıştır. Altıncı bölümde de yani son bölümde çalışmanın sonucu ve ileride yapılması düşünülen çalışmalara dair öneriler sunulmuştur.

2. KAVRAMSAL ÇERÇEVE

2.1. WEB EKOSİSTEMİ

Web uygulamalar, donanımlar, insanlar, servisler, oluşturulan veriler ve tüm bu bileşenlerin aralarındaki ilişkiler de kapsanmak üzere günden güne büyüyen ve yeni bileşenlerin katılabileceği, sorunların veya ihtiyaçların çözülebildiği ortama web ekosistemi denir [8]. Bu ekosistemde birçok nesne yer almasına rağmen tez konusu kapsamında istemciler, sunucular, istekler/cevaplar, protokoller, yöntemler ve uygulamalar ile ilgili bilgiler verilecektir.

2.1.1. İstemci (Client)

İsteklere cevap verecek cihazlardan herhangi bir dijital istekte bulunan sistemler ya da kişiler olarak tanımlanabilir. İstemcilere; insanlar, sensörler, el bilgisayarları, pos cihazları vb. örnek olarak verilebilir.

2.1.2. Sunucu (Server)

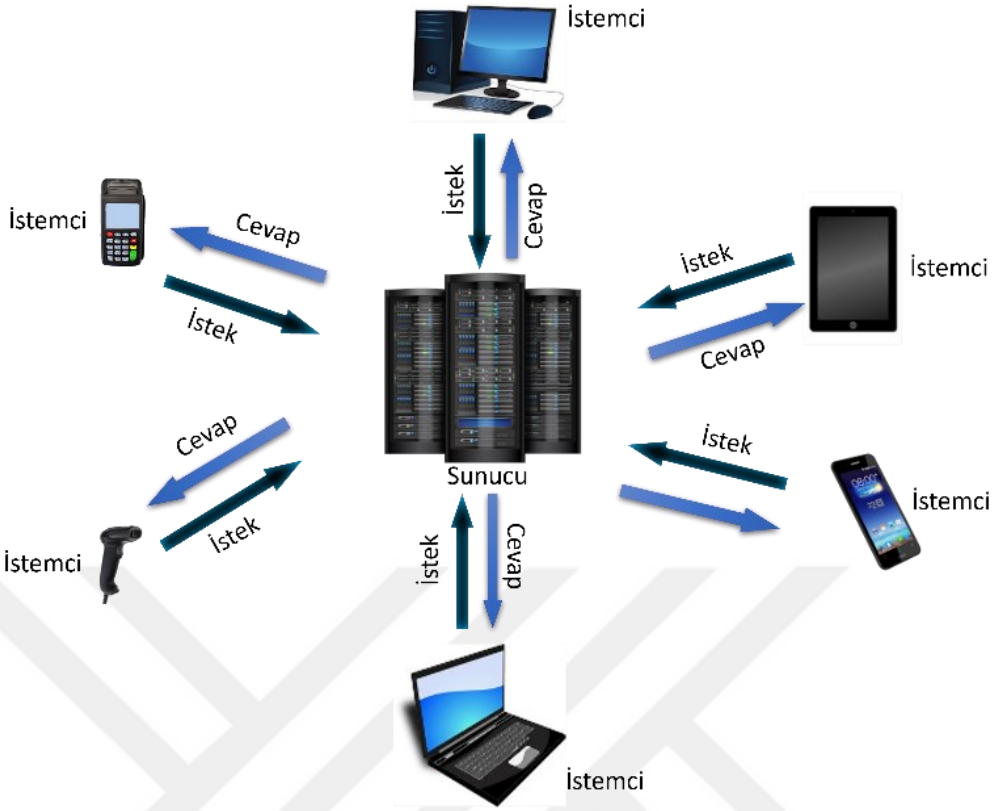
Kullanıcının ya da sistemlerin isteklerine 7/24 açık olan ve kesintisiz bir şekilde cevap veren özel ve güçlü bilgisayarlardır. Bu özel bilgisayarlar genellikle veri merkezlerinde bulundurulur.

2.1.3. İstek (Request)

İstemcilerin sunuculara gönderdiği bilgilerle beraber göndermiş olduğu HTTP metotlarına denir. Örnek: sunucudan bir sayfa talep edilmesi, sunucuya kişisel bilgileri göndererek kaydedilmesini talep etmesi gibi.

2.1.4. Cevap (Response)

Sunucunun, istemciden gelen isteğe karşılık verdiği cevap olarak tanımlanır. Talep edilen sayfanın gönderilmesi, gönderilen bilgilerin kaydedilmesi, istenen bilgilerin kaydedildiğinin bilgisinin verilmesi örnek olarak verilebilir. Cevapla birlikte istemciye durum kodu da gönderilmektedir. Durum koduna göre isteğe verilen cevabın durumu belirlenebilir. HTTP durum kodları IANA tarafından kaydedilmektedir. Durum kodları 5 ana grupta toplanmaktadır ve 3 basamaklıdır. Örneğin 404 durum kodu, istek yapılan sayfa bulunamadı anlamına gelmektedir.



Şekil 2.1: İstemci-Sunucu Mimarisi

2.1.5. HTTP ve HTTPS

HTTP, istemci ile sunucu arasında veri alışverişini sağlayan OSI uygulama katmanına dâhil olan protokoldür. HTTPS ise HTTP'nin SSL sertifikası sayesinde güvenli hale getirilmiş versiyonudur. Böylece bu iletişim protokolü kullanılarak yapılan bağlantılarda kullanıcı ile sunucu arasındaki veri aktarım şifreli olarak yapılması sağlanır. Ayrıca kullanıcı ile sunucu arasına başka bir kişinin girmesi engellenir ya da girmesi durumunda tespit edilmesi sağlanır.

2.1.6. SSL

Kullanıcılar (istemci) ile web siteleri(sunucu) arasındaki güvenli iletişimi sağlar ve güvenli bağlantı katmanı olarak isimlendirilir. Sertifika ile web sitesinin kendini doğrulaması ve kullanıcı ile sunucu arasında aktarılan bilgilerin gizliliği sağlanmış olur.

2.1.7. Temel HTTP Yöntemleri

Bu bölümde tez kapsamında veri seti içinde yer alan temel HTTP yöntemlerine değinilecektir. Bu yöntemler GET, POST, PUT ve DELETE olarak yer alacaktır.

2.1.7.1. GET

Kullanıcının sunucudan bir sayfa ya da bilgi isteğinde bulunduğu yöntemdir. Bir web sayfasının ya da sistemde kayıtlı olan öğrencilerin listesinin istenmesi örnek olarak verilebilir.

2.1.7.2. POST

Kullanıcının sunucuya göndermiş olduğu bilgilerin kaydedilmesi ya da sorgulanması isteğinde bulunduğu yöntemdir. Örnek, kullanıcı sayfaya giriş işlemini gerçekleştirirken sistem tarafından kullanıcı adının ve şifresinin doğruluğunun kontrol edilmesi ya da gönderilen kullanıcı bilgilerinin sisteme kaydedilmesi talebinde bulunulması bu yöntemle yapılmaktadır.

2.1.7.3. PUT

İstemcinin sunucuya göndermiş olduğu bilgilerle sistemdeki eski bilgilerin güncellenmesi talebinde bulunduğu yöntemdir. Telefon numarasını değiştiren personelin yeni telefon numarası gönderilerek güncellenme talebinde bulunulması örnek olarak verilebilir.

2.1.7.4. DELETE

İstemcinin sunucudaki bir bilginin silinmesi talebinde bulunduğu HTTP yöntemidir.

2.1.8. Web Sunucu

Varsayılan olarak 80 ve 443 numaralı portları dinleyen ve web sayfa isteklerine cevap veren yazılımlardır. Ayrıca oluşan her isteğin ve cevabın log kaydını istenilen düzeyde kaydederler. En çok bilinen web sunucuları Apache ve IIS'dir.

2.1.9. Veri Tabanı Yönetim Sistemleri

Verileri düzenli olarak kaydedildiği ortamlara veri tabanı denir. Veri tabanlarını yönetmek, analiz etmek, veri tabanları üzerinde işlemler yapmak ve diğer uygulamalardan gelen istekleri yerine getirmek gibi görevi olan yazılımlara ise veri tabanı yönetim sistemi (VTYS) denir.

2.1.10. SQL

Veri tabanı üzerinde işlemleri yapmaya yarayan yapılandırılmış sorgulama diline SQL denir. Temel SQL komutları bütün veri tabanları için ortaktır. Veri tabanları kendilerine özel işlemler için SQL'in genişletilmiş versiyonunu geliştirmekte ve kullanmaktadırlar. Microsoft için T-SQL, Oracle için PL/SQL bu duruma örnek verilebilir. Aşağıda temel SQL komutları ve yaptıkları işlemler Tablo 2.1'de yer almaktadır.

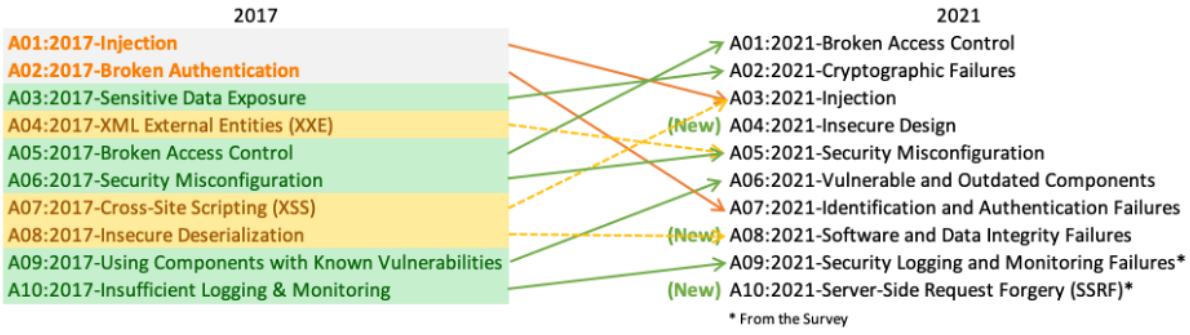
Tablo 2.1: Temel SQL Komutları [9]

KOMUT	YAPTIĞI İŞ
SELECT	Genellikle veri listelemek için kullanılan bu komut veri tabanından veri seçmek için kullanılır.
INSERT	Veri tabanına yeni bir veri ya da veriler eklemek için kullanılır.
UPDATE	Veri tabanında daha önce kayıtlı olan bir kaydı veya kayıtları güncellemek için kullanılır.
DELETE	Veri tabanında daha önce kayıtlı olan bir kaydı veya kayıtları silmek için kullanılır.
CREATE	Yeni bir veri tabanı nesnesi oluşturmak için kullanılır.
DROP	Daha önceden oluşturulmuş bir nesneyi silmek için kullanılır.

2.2. WEB SALDIRI TEKNİKLERİ

COVID-19 salgınından sonra çevrimiçi platformlardan yapılan alışverişlerin sayısı artış göstermiştir. Bu durum siber saldırganlar tarafından fırsat bilinmektedir ve son zamanlarda saldırıların yoğunluğu artmıştır. Bu saldırılardan en çok etkilenen ortamlardan birisi de web uygulamaları veya web hizmeti sağlayan sunuculardır [10].

Web saldırı kategorileri için referans alınan kaynaklardan biri olan OWASP (Open Web Application Security Project), 2021 yılında 3 yeni kategori eklemesinin yanında 4 kategorinin isminde ve kapsamında değişikliğe gitmiş ve bazı birleştirmeler yapmıştır [11]. OWASP'ın yayınlamış olduğu web uygulamaları risklerinin 2017 ile 2021 arasında değişimi Şekil 2.2'de yer alan eşleştirmeli görselde görülebilmektedir.



Şekil 2.2: OWASP Top 10 Web Uygulaması Güvenlik Riskleri [11]

2.2.1. Bozuk Erişim Kontrolü (A01:2021-Broken Access Control)

Erişim kontrolü, kullanıcıların verilen yetkiler doğrultusunda izin verilen alanlara erişim sağlamasıdır. Burada yapılacak bir hata, kullanıcıların izin verilmeyen alanlara erişmesine

sebep olacaktır. Bu da güvenliğin temel unsuru olan CIA'yı yani gizliliği (confidentiality), bütünlüğü (integrity) ve/veya erişebilirliği (availability) tehlikeye düşürecektir [12].

2.2.2. Kriptolojik Hatalar (A02:2021-Cryptographic Failures)

Web siteleri ile iletişime geçildiği bazı durumlarda o siteye telefon numarası, adres, şifre, kredi kartı bilgileri vb. gibi kişisel bilgilerimizi de vermek zorunda kalırız. Bu bilgiler kaydedilmek zorundaysa veri tabanlarında güvenli bir şekilde saklanmak zorundadır. Burada yapılacak hatalar kişisel verilerin ifşa edilmesine sebep olmaktadır. Kullanımdan kaldırılmış olan MD5 veya SHA1 gibi özetleme (hash) algoritmalarının kullanılması bu hataya örnek olarak verilebilir [13].

2.2.3. Enjeksiyonlar (A03:2021-Injection)

Enjeksiyon saldırıları kullanıcıdan alınan bilgilerin uygulama tarafından doğrulanmadan, filtrelemeden veya zararlı içeriklerin temizlenmeden sorgularda kullanılmasından ya da veri tabanına kaydedilmesinden dolayı meydana gelmektedir [14]. Enjeksiyon saldırılarına birçok kesim tarafından çok iyi bilenen SQLi örnek olarak verilebilir.

2.2.4. Güvensiz Tasarım (A04:2021-Insecure Design)

Projeler geliştirilirken Güvenli Geliştirme Yaşam Döngüsü prensiplerine uyulmalıdır. Bir güvenlik zafiyeti oluşturacak kodlama test aşamasında değil daha projeye başlamadan önce tespit edilip projenin geliştirmesi burada alınan önlemlere göre olmalıdır. Projenin tüm aşamalarında güvenlik testleri ve entegrasyonun yapılması ihmal edilmemelidir [15].

2.2.5. Yanlış Güvenlik Yapılandırması (A05:2021-Security Misconfiguration)

Uygulamanın ve çalıştığı ortam ayarlarının ve izinlerinin doğru bir şekilde yapılması önemlidir. Burada yapılacak yanlış veya yetersiz yapılandırmalar, gereksiz özelliklerin aktifleştirilmesi ya da yüklenmesi, varsayılan kullanıcıların aktif olması, basit veya varsayılan parolaların kullanılması, kullanıcıya olması gerekenden fazla detaylı hata mesajlarının verilmesi ya da yeteri kadar yapılmayan güvenlik sıkılaştırmaları(hardening) vb. durumlar uygulamada zafiyet oluşturabilir [16].

2.2.6. Zafiyetli ve Güncel Olmayan Bileşenler (A06:2021-Vulnerable and Outdated Components)

Güncellemeler yeni özelliklerin eklenmesinin yanında performans ve güvenlik düzeltmeleri de içermektedir. Bu sebeple işletim sisteminin, web ve uygulama sunucusunun, veri tabanı yönetim sisteminin (DBMS), uygulamaların, API'lerin, bileşenlerin, çalışma zamanı

ortamının ve kütüphanelerin güncellenmesi gerekmektedir. Bu güncellemelerin yeteri kadar yapılmadığı ya da güncellemelerin eksik yapıldığı durumlarda güvenlik zafiyeti oluşacaktır [17].

2.2.7. Tanımlama ve Doğrulama Hataları (A07:2021-Identification and Authentication Failures)

Kullanıcıyı benzersiz şekilde tanımlama (indentificaiton), kullanıcının gerçekten o kişi olduğunun doğrulanması (authentication) ve oturum yönetiminin (session) doğrulanması bir uygulama için en kritik süreçlerden biridir. Burada basit veya bilenen şifrelerin kullanılmasına izin verilmesi, etkisiz kimlik doğrulama faktörlerinin kullanımı, URL'de kimlik bilgisinin görünmesi vb. gibi durumların olmaması gerekmektedir [18].

2.2.8. Yazılım ve Veri Bütünlüğü Hataları (A08:2021-Software and Data Integrity Failures)

Yazılım ve veri bütünlüğü hataları, güvenilmeyen kaynaklardan, havuzlardan (repository) veya içerik dağıtım ağlarından (CDN) gelen eklentilerin, kütüphanelerin veya modüllerin kusurlu olmasından dolayı oluşan zafiyete denir. Aynı zamanda bu güvenilmeyen kaynaklardan dolayı serileştirilmiş verilerdeki bütünlüğün bozulma durumu da mevcuttur [19].

2.2.9. Güvenlik Loglaması ve İzleme Hataları (A09:2021-Security Logging and Monitoring Failures)

Günlük kayıtları(log) sistemde gerçekleşen durumların kaydedildiği dosyalardır. Doğru şekilde yapılandırılmış bir sistemdeki bu dosyalar sayesinde neler olup bittiği net şekilde görülebilmektedir. Güvenlik açısından bakıldığında günlük kayıtları ile sistemdeki aktif ihlallerin, yetki yükseltme işlemlerinin, saldırının zamanının, yerinin ve yönteminin tespit edilmesi mümkündür. İzleme ise bu log kayıtlarının analiz edilmesi ile ortaya çıkan bir ihlal varsa bunu yetkili kişilere göstermesi ve alarm oluşturmasıdır. Günlük kayıtlarında veya izlemede bir hata olması güvenlik zafiyetine sebep olmaktadır [20]. Bu tez kapsamında yapılan çalışma bu başlık altında yer almaktadır.

2.2.10. Sunucu Taraflı İstek Sahteciliği (A10:2021-Server-Side Request Forgery)

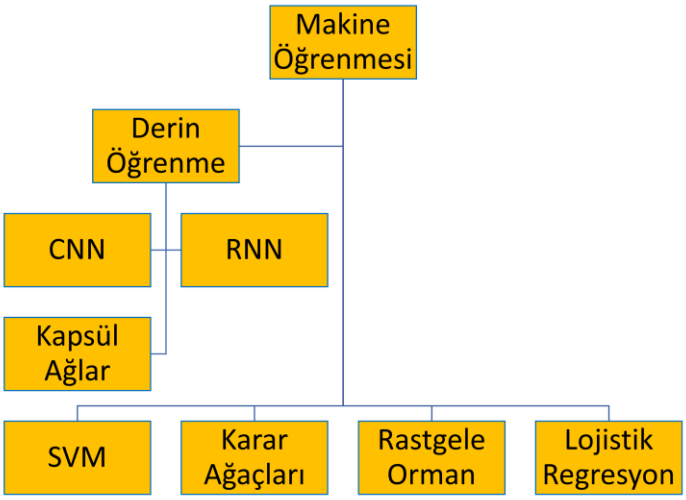
Saldırganın sağladığı URL'den sunucunun istekte bulunmasıyla orta çıkan saldırı türüdür. Sunucu tarafında bu tarz isteklerin olması gerekiyorsa uzak kaynaklara erişen ağın normal ağdan ayrılması, sadece intranet bağlantı varsa diğer trafiklerin tamamen engellenmesi, kullanıcı tarafından gönderilen bilgilerin filtreleme ve temizleme aşamalarından geçirilmesi gibi önlemler alınarak SSRF önlenebilmektedir [21].

2.3. YAPAY ZEKÂ

Nilsson'a göre yapay zekâ (artificial intelligence - AI) kısaca, makineleri akıllı hâle getirmeye adanmış faaliyetlerdir [22]. Yapay zekâ, insanların veya hayvanların davranışlarını taklit etme, benzerini yapma veya modelleme gayesiyle bilgisayar yazılımlarının geliştirilmesi olarak da tanımlanabilmektedir [23]. Yapay zekâ, makine öğrenmesini ve makine öğrenmesi de derin öğrenmeyi kapsamaktadır.

2.4. MAKİNE ÖĞRENMESİ

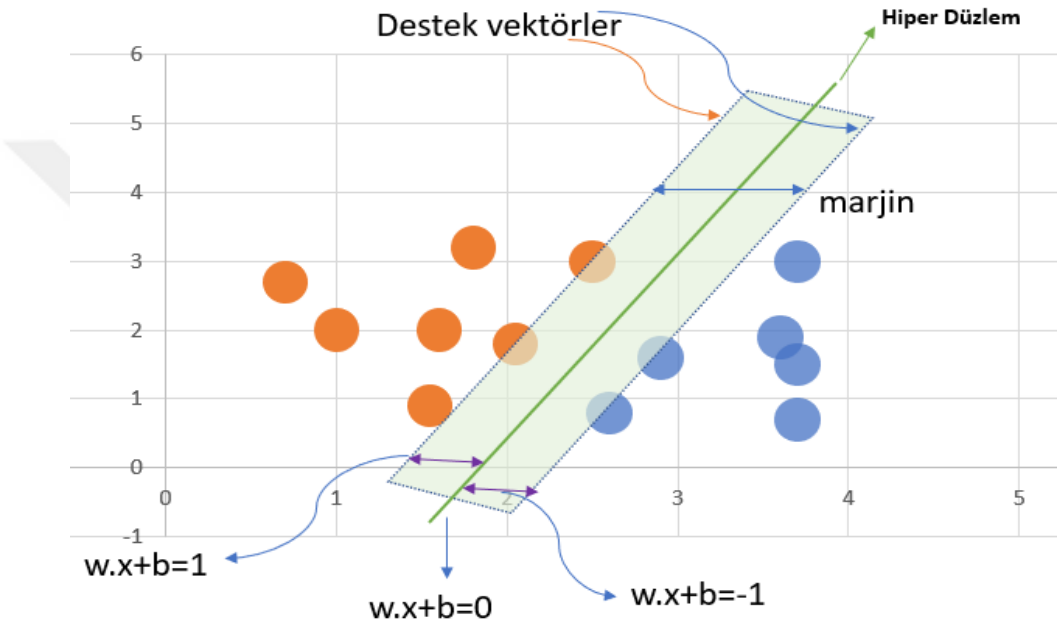
Makine öğrenmesinde verilen bilgiler eşliğinde edinmiş olduğu öğrenmelerinden bir model oluşturulur. Oluşturulan bu model kullanılarak gelecekte karşılaşılabilecek durumlar karşısında bir tahminde bulunulmasını ya da karar verilmesini sağlanır. Makine öğrenmesi ile bir ürünün gelecekteki değeri tahmin edilebilir, bir davranışın ya da nesnenin zararlı olup olmadığına karar verilebilir, nesneleri gruplayabilir, önerilerde bulunabilir, bir fotoğrafın kime ait olduğu tespit edilebilir vb. Bu ve diğer işlemler yüksek doğrulukla yapılabilir. Makine öğrenmesi ile ilgili farklı birçok türde sınıflandırma içeren şema bulunmaktadır. Tez kapsamında incelenen temel yöntemlerin bulunduğu makine öğrenmesi algoritmaları Şekil 2.3'te gösterilmiştir. Temel algoritmalardan destek vektör makineleri (SVM), karar ağaçları (DTs), rastgele orman (RF) ve lojistik regresyon (LR) algoritmaları ile ilgili bilgiler verilecektir. Derin öğrenme makine öğrenmesinin bir alt dalında yer almaktadır. CNN, RNN ve Kapsül Ağlar hakkında Derin Öğrenme başlığı altında bilgi verilecektir.



Şekil 2.3: Makine Öğrenmesi Algoritmaları

2.4.1. Destek Vektör Makinesi (Support Vector Machine - SVM)

Sınıflar arasındaki ayrımın optimum olmasını sağlayacak hiper doğru veya düzlemi bulmak olarak tanımlanabilir. SVM ilk olarak sınıflandırma amaçlı ortaya çıkmıştır [22]. SVM istatistiksel öğrenme temellidir. İki sınıflı SVM için; iki sınıfın uç gözlemlerinin olduğu yerlere çizilen paralellere destek vektörleri denir. İki destek vektör arasındaki alana marjin denir. Bu marjinin maksimum olduğu durumda çizilen doğruya hiper düzlem adı verilmektedir [24]. Bu temel terimler Şekil 2.4'te görülmektedir.

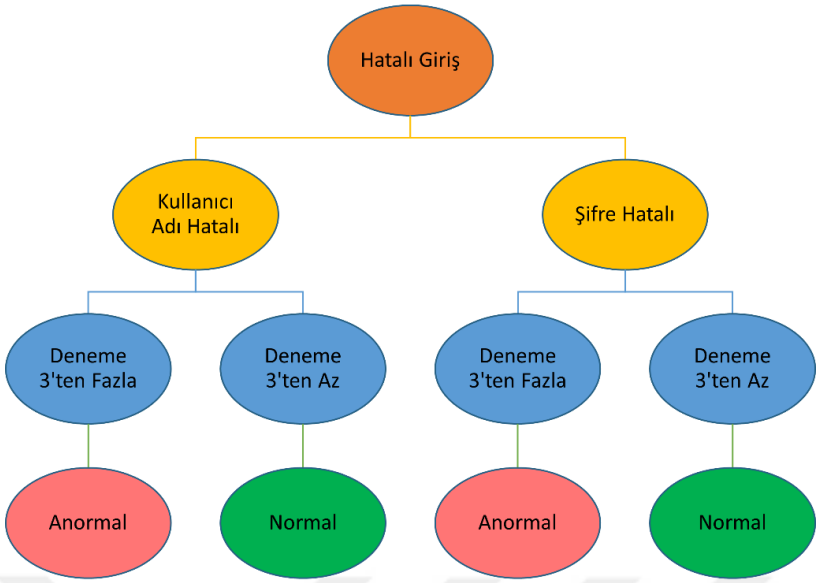


Şekil 2.4: 2 Sınıflı Destek Vektör Makinesi

Destek vektör makineleri iki sınıf arasında sınıflandırma yapmak üzere tasarlanmasına rağmen günümüzde daha fazla sınıftan oluşan veri setlerinde de başarıyla sınıflandırma işlemi yapılabilmektedir. Ayrıca SVM'ler regresyon tahmininde de kullanılabilirler.

2.4.2. Karar Ağaçları (Decision Tree - DT)

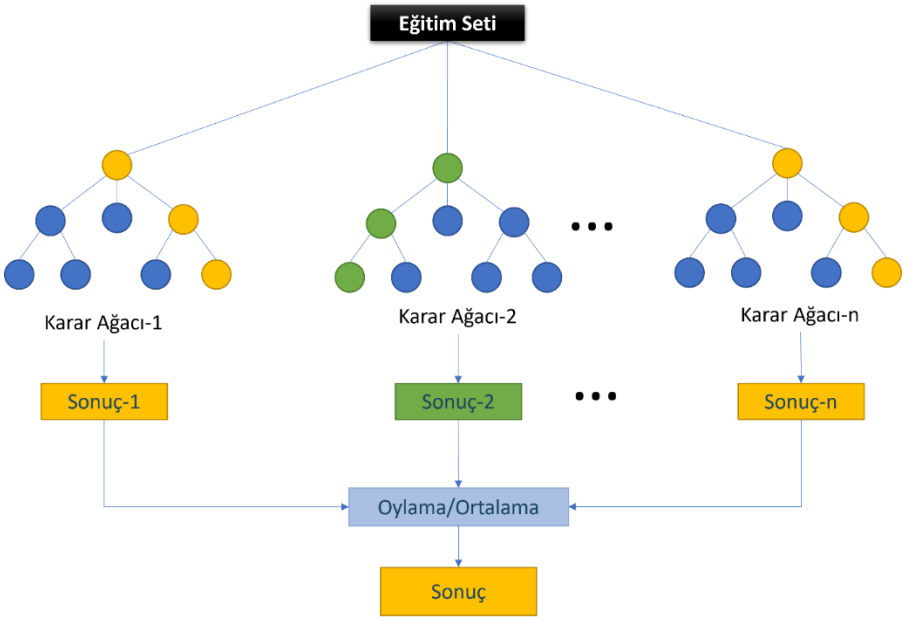
Karar ağaçları sınıflandırma ve regresyon problemlerinde sıklıkla kullanılan ML yöntemlerinden birisidir. Kök düğümdeki duruma göre sağ ve sol düğümlere dallanan ve en nihayetinde yapraklara ulaşarak sonuç üreten bir algoritmadır. Amaç eğitim verilerinden kural listesi çıkararak bir model oluşturmak ve oluşan model ile tahminde bulunmaktır. Tipik bir karar ağacı Şekil 2.5'te görülmektedir.



Şekil 2.5 Karar Ağacı Örneği

2.4.3. Rastgele Orman (Random Forest - RF)

Rastgele orman yöntemi karar ağaçlarının üzerine oluşturulmuş bir yöntemdir. Verilen parametre değeri kadar eğitim setinin farklı özelliklerinden farklı ağaçlar oluşturulur. Oluşturulan ağaçların sonuçlarından oylama ya da ortalama ile nihai sonuç üretilir. Karar ağaçlarının aşırı uyum problemine karşı bir çözümdür. Tipik bir rastgele orman şeması Şekil 2.6'daki şemada görülmektedir.

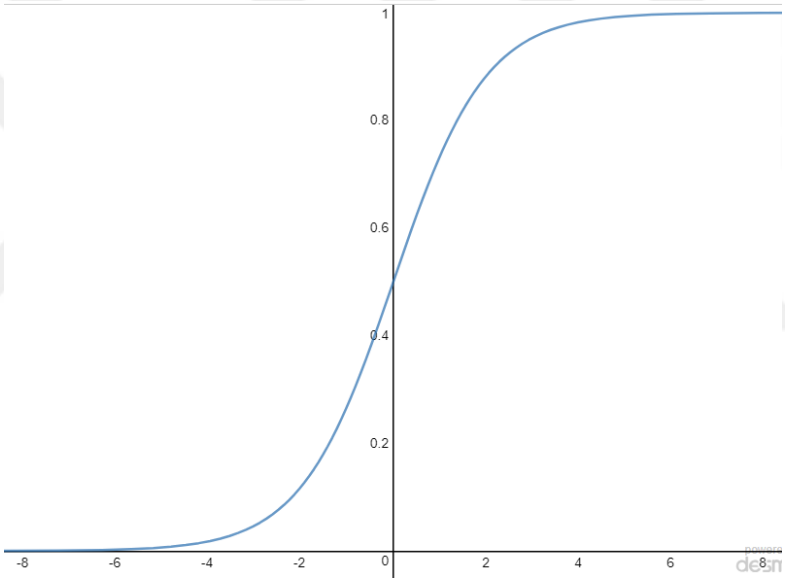


Şekil 2.6: Rastgele Orman Örneği

2.4.4. Lojistik Regresyon (Logistic Regression - LR)

Lojistik regresyon her ne kadar isminde regresyon ifadesi olsa da asıl görevi sınıflandırma yapmaktır. İkili, sıralı ve çoklu sınıflandırma olarak 3 türü bulunmaktadır. İkili sınıflandırmada bağımlı değişken 0 ve 1 olmak üzere iki değer alır. Sıralı sınıflandırmada bağımlı değişken sıralı kategorik değerler alır. Üçüncü türünde ise bağımlı değişken 3 veya daha fazla sıralı olmayan kategorik değer alır. Tezde yapılan çalışmada isteğin zararlı mı (1) zararsız mı (0) olduğunun tespiti yapılacaktır. Bu yüzden ikili (binary) lojistik regresyon kullanılacaktır.

Lojistik regresyon Şekil 2.7'de görüleceği üzere temel lojistik regresyon $-\infty$ ile ∞ arasında alacağı değerlere karşılık çıkış olarak 0 ile 1 arasında bir değer verecektir.



Şekil 2.7: Lojistik Regresyon

Lojistik regresyonun matematiksel tanımlaması şu şekildedir [25]: Denklem (2.1)'de LR'deki giriş değişkenleri olan x'ler bağımlı değişken olan y'nin olasılığını açıklar. Formüldeki $p(y = 1|x)$, 1 olma olasılığıyken $p(y = 0|x)$, 0 olma olasılığıdır. Dolayısıyla olasılıklar bu aralıkta olacaktır.

$$\frac{p(y = 1|x)}{p(y = 0|x)} = \frac{p(y = 1|x)}{1 - p(y = 1|x)} \tag{2.1}$$

Denklem (2.2)'de giriş değişkenleri ile odds veya log-odds'un logaritmik formu arasındaki doğrusal ilişki gösterilmektedir.

$$ln\frac{p(y=1|x)}{1-p(y=1|x)} = \beta_x \tag{2.2}$$

Lojistik fonksiyon ya da sigmoid fonksiyon kullanılarak Denklem (2.2) yeniden düzenlendiğinde Denklem (2.3) elde edilecektir.

$$\beta = (\beta_1, \beta_2, \dots, \beta_n) \epsilon R^n$$

$$p(y=1|x) = \frac{e^{\beta_x}}{1+e^{\beta_x}} \tag{2.3}$$

Denklem (2.4)'te sigmoid fonksiyon ya da lojistik fonksiyon olarak tanımlanır.

$$\beta_{(z)} = \frac{1}{1+e^{-z}}$$

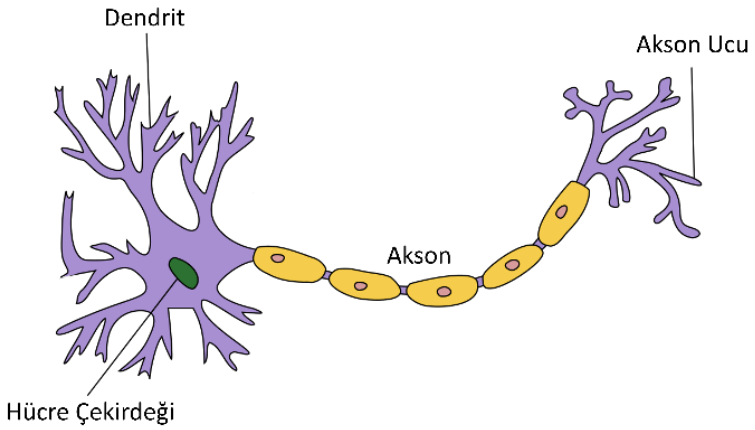
$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n = 0 \tag{2.4}$$

$x \epsilon R^n$ olmak üzere tüm giriş değişkenleri Denklem (2.5)'te kullanılarak LR'nin karar verme sınırları belirlenebilir.

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n = 0 \tag{2.5}$$

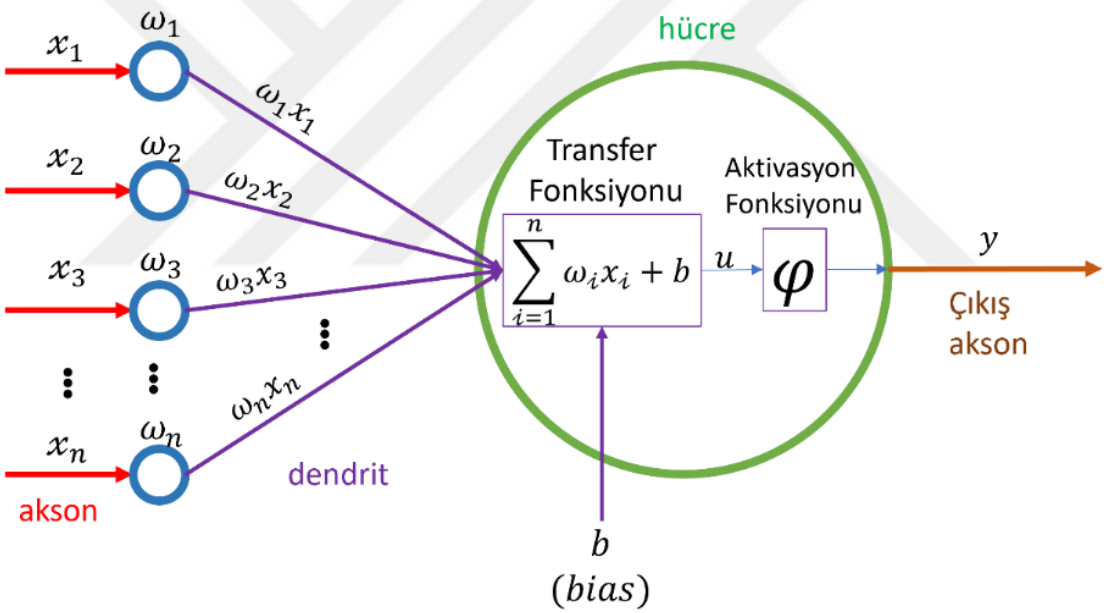
2.4.5. Yapay Sinir Ağları (Artificial Neural Networks - ANNs)

Buluşlara bakıldığında doğadan çok fazla yararlanıldığı görülür. Örneğin uçaklar kuşlardan, radarlar yarasalardan ve robot kollar insan kolundan ilham alınarak bulunmuştur. Yapay sinir ağları da insan beyninin öğrenmesi modellenerek bulunmuştur [26]. Şekil 2.8'de biyolojik nöron görülmektedir. Dendritlerden giren bilgiler çekirdekte ağırlıklandırılarak toplanır ve akson boyunca iletilir. Nöronlar arasında meydana gelen sinaptik bağlantılar gerçekleştirilerek öğrenme gerçekleşir.



Şekil 2.8: Biyolojik Nöron[1]

İnsan beynindeki nöronların modeli olan yapay nöronlar matematiksel olarak gösterimi Şekil 2.9'da şemada gösterilmiştir.

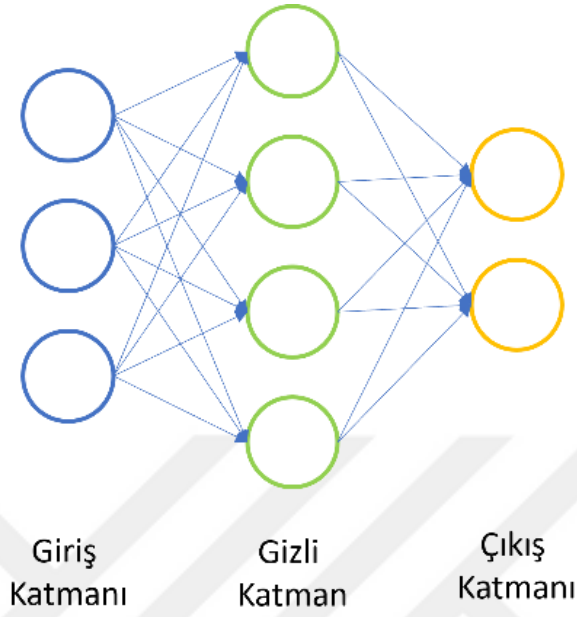


Şekil 2.9: Yapay Nöron Modeli

Aksonlardan gelen bilgi (x_i) dendritteki ağırlık (ω_i) hücreye gönderilir ve hücreden gelen bütün bu değerler çarpılır ve toplanır. Daha sonra bir bias (b) değeri eklenir. Bulunan sonuç aktivasyon fonksiyonuna tabi tutulur. Aktivasyon fonksiyonundan (φ) elde edilen sonuç çıkışa gönderilir [27]. Tek katmanlı algılayıcılar ile doğrusal olan problemler çözebilirken doğrusal

[1] Quasar tarafından oluşturulan resim düzenlenerek alındı. Resmin alındığı internet sayfası, https://tr.wikipedia.org/wiki/Sinir_hücresi

olmayan problemler için çok katmanlı sinir ağları kullanılır. Şekil 2.10'daki şemada tipik bir ANN yapısı görünmektedir.



Şekil 2.10: Yapay Sinir Ağı Örneği

2.5. DERİN ÖĞRENME (DEEP LEARNING - DL)

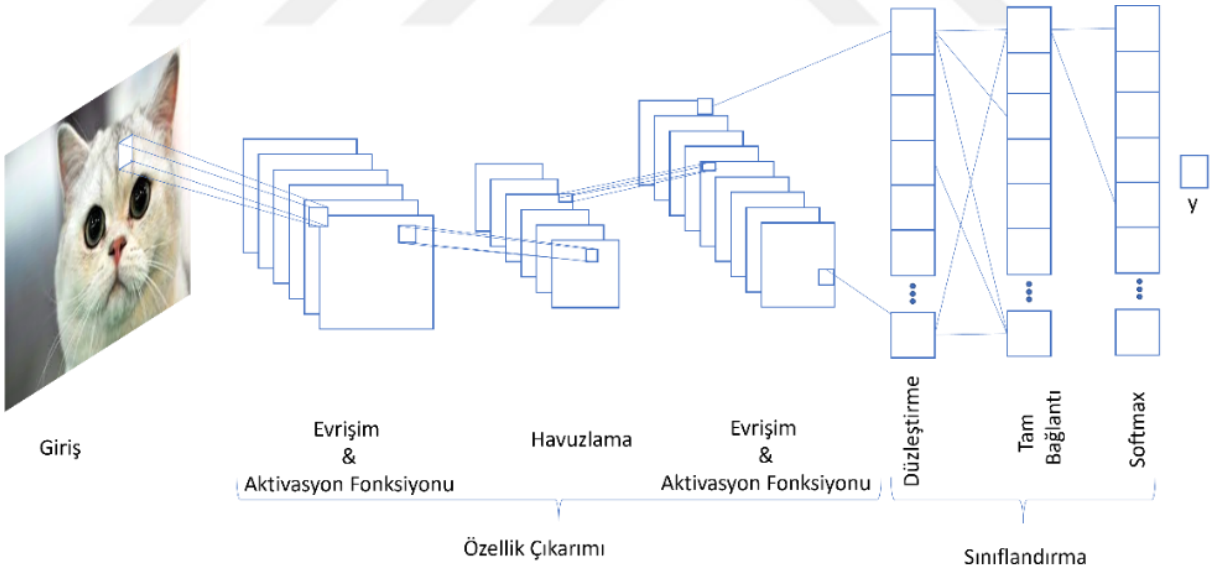
Derin öğrenme, Şekil 2.11'de görüleceği gibi makine öğrenmesinin altında yer alan son zamanlarda en yaygın kullanılan yaklaşımlardan biri haline gelmiştir. Bu sayede birçok bilişsel uygulamada çok iyi sonuçlar elde edilmiştir. DL'nin en büyük faydaları arasında büyük veri ile başa çıkabilmesi vardır. Derin öğrenmenin ML'den en büyük farkı verileri alıp işleme geçmesidir. ML'de öncelikle bazı işlemlerin (özelliklerin tespiti, özelliklerin seçimi vb.) yapılması gerekmektedir. Burada uygulayıcının etkisi büyük olmaktadır. Fakat DL, özelliklerin öğrenilmesi işlemini kendisi yapmaktadır. DL; doğal dil işleme (NLP), görüntü işleme, biyoinformatik, robotik ve kontrol, tıbbi bilgi işleme ve tezin de kapsamında olduğu siber güvenlik alanlarında makine öğrenmesine kıyasla daha iyi sonuçlar vermektedir [28]. DL'nin en büyük zorluklarından birisi ise güçlü donanımlara ihtiyaç duymaktadır. Özellikle ağın derinliği arttıkça hesaplama miktarı artmakta bu durum ise eğitim süresini uzatmaktadır. Uzun bir süre DL yöntemlerinin kullanılmaması daha doğrusu kullanılamamasının sebebi bu durumdur. Günümüzde bilgisayar sistemleri ekran kartları sayesinde çok büyük hesaplama gücüne kavuşmuştur.



Şekil 2.11: Yapay Zekâ, Makine Öğrenmesi ve Derin Öğrenme Arasında İlişki [28]

2.5.1. Evrişimli Sinir Ağları (Convolutional Neural Networks - CNNs)

Evrişimli Sinir Ağları (CNN), yapay sinir ağları sınıfının bir üyesidir. Giriş katmanı, aktivasyon fonksiyonu içeren evrişim katmanları, havuzlama katmanı ve tam bağlı katman gibi çeşitli katmanlardan oluşur [29]. Görüntü işleme, NLP vb. alanlarda başarıyla kullanılmaktadır. Şekil 2.12'de örnek bir CNN yapısı görülmektedir [30].

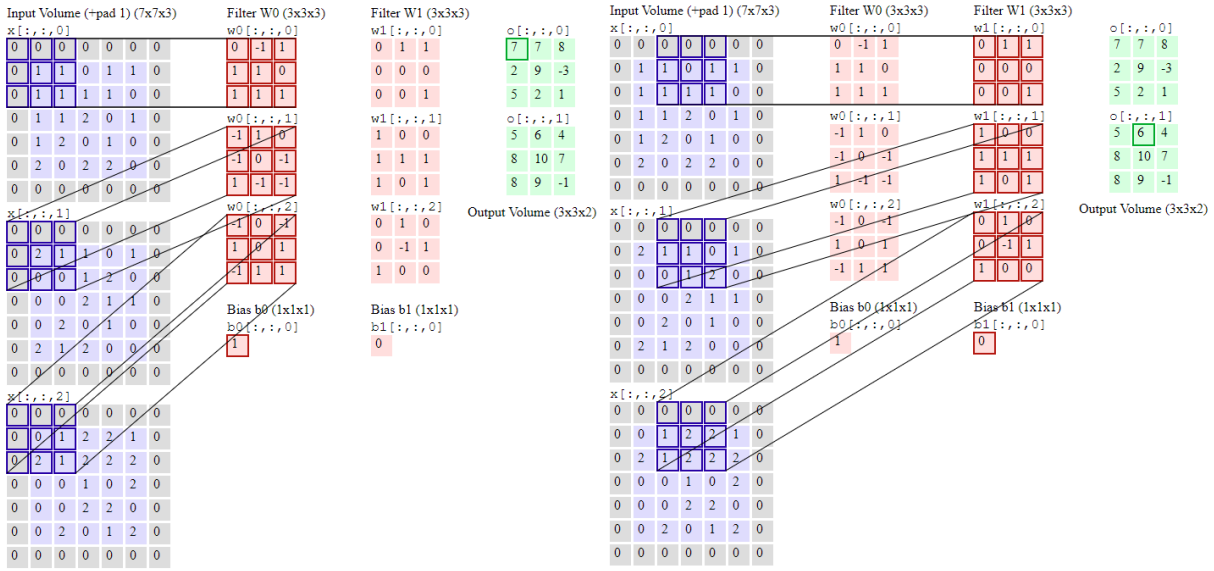


Şekil 2.12: Evrişimli Sinir Ağı Örneği [30]

2.5.1.1. Evrişim Katmanı

Evrişim katmanı, CNN'nin temelidir. Girişler, çekirdek (kernel) ya da diğer adıyla filtreden geçirilerek yeni halini alır. Bu evrişim işlemin nasıl gerçekleştiğini gösteren küçük bir

kesit Şekil 2.13'te verilmiştir. Daha sonra aktivasyon fonksiyonu uygulanarak havuzlama katmanına gönderilir.

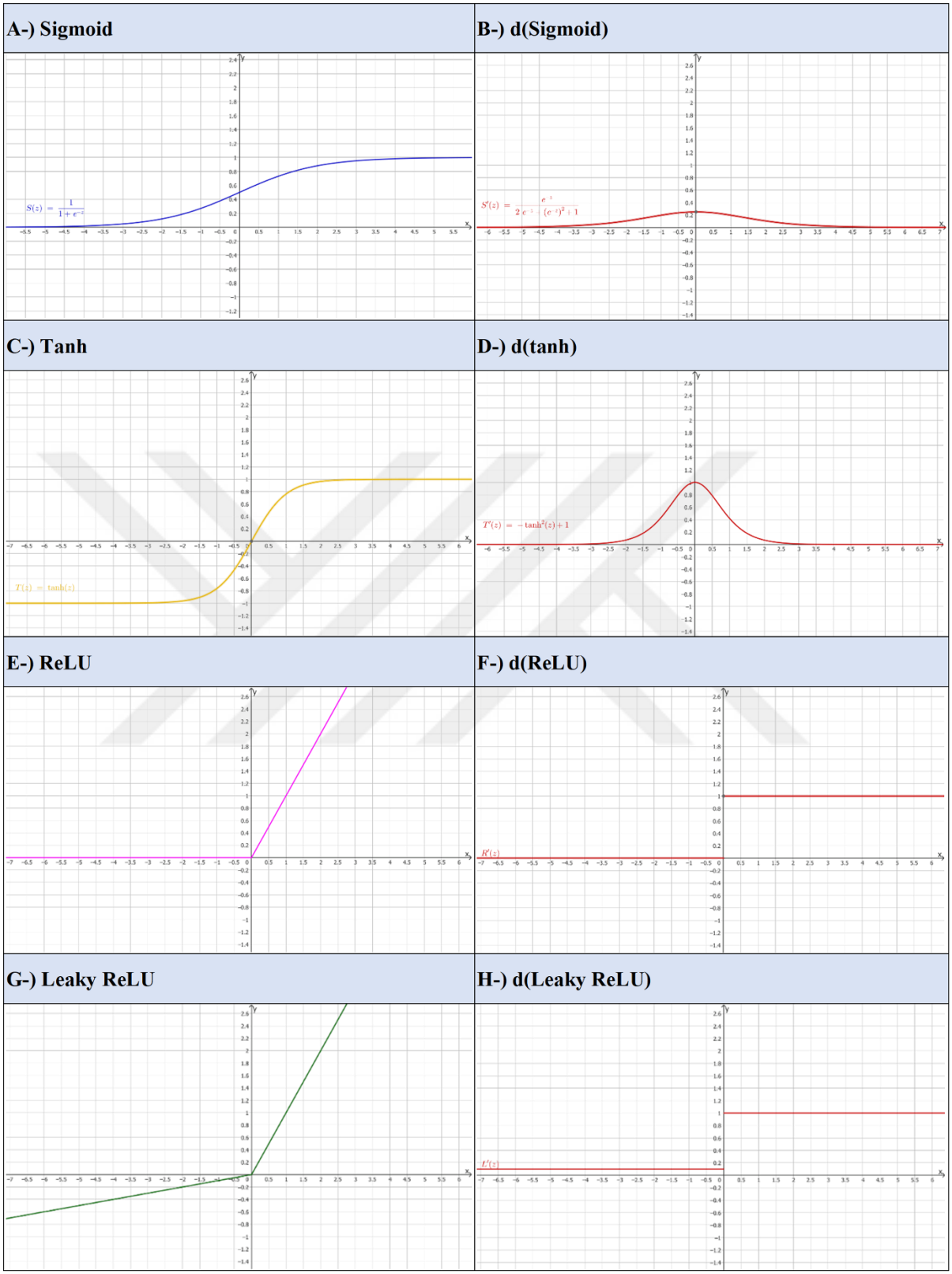


Şekil 2.13: Evrişim Katmanı İşlem Adımı [31]

Giriş matrisinin dış kenarlarına 0 değeri ile doldurma (padding) işlemi uygulanmıştır. Şekil 2.13'te adım olarak 2, 3x3 boyutunda olmak üzere 2 adet filtre seçilmiştir. Böylece $3x3x2$'lik bir çıkış elde edilmiş olur. Doldurma işlemi yapılmadan önce orijinal girişi $5x5$'tir. Eğer adım olarak 1 seçilmiş olsaydı doldurma işleminden dolayı çıkış $5x5$ olacaktır. Filtre ile giriş matrisi bir bir çarpılır ve bias değeri eklenerek çıkış değerleri oluşturulur. Her seferinde seçilen adım kadar filtre giriş üzerinde kaydırılarak çıkış matrisi oluşturulmuş olacaktır. Daha sonra ikinci filtre aynı yöntemle uygulanır ve evrişim işlemi tamamlanmış olur.

2.5.1.2. Aktivasyon Fonksiyonu

Aktivasyon fonksiyonu, doğrusal olmayan girdiyi çıktıya eşlemek için kullanılır. Giriş katmanından gelen veri doğrusal olmayan aktivasyon fonksiyonundan geçirilerek doğrusal olmayan bir çıkış elde edilmiş olur. Kullanılan birçok aktivasyon fonksiyonu vardır. Şekil 2.14'te dört aktivasyon fonksiyonun girişlere uygulanması sonucunda oluşan çıkışların grafikleri görülmektedir [28].

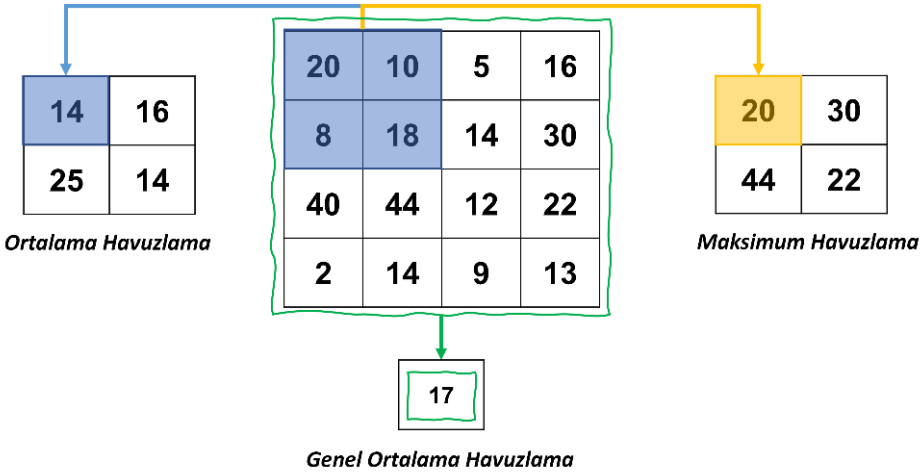


Şekil 2.14: Aktivasyon Fonksiyonları

Bölüm 2.5.1.6'da detaylı bir şekilde anlatılacak olan geri yayılım işleminden dolayı aktivasyon fonksiyonun türev alınabilir bir fonksiyon olması gerekir. Örneğin ReLU işleminde negatif değerlerde sonuç sıfır olduğu için türevde sıfır olacaktır. Eğer negatif değerlerde de öğrenme olması isteniyorsa farklı bir aktivasyon fonksiyonu seçilmelidir.

2.5.1.3. Havuzlama Katmanı

Ortaklama ya da havuzlama katmanı olarak geçen bu katmanda evrişim işlemi uygulanmış girişin çıkış boyutunun azaltılması işlemi yapılır. Bu işlemde baskın olan bilgiler korunmuş olacaktır. Havuzlama yöntemleri; ağaç havuzlama (tree pooling), kapılı havuzlama (gated pooling), minimum havuzlama (min pooling), ortalama havuzlama (average pooling), maksimum havuzlama (max pooling), genel ortalama havuzlama (global average pooling) ve genel maksimum havuzlama (global max pooling) olarak sayılabilir. En çok bilinen minimum, maksimum ve genel ortalama havuzlama yöntemleridir. Bu yöntemlere ait uygulama örneği Şekil 2.15'da gösterilmektedir [28]. Ortalama havuzlamada 20, 10, 8 ve 18 değerlerinin toplamı 54 elde edilmektedir. Ortalaması alındığında ise 14 değeri elde edilmektedir. Maksimum havuzlamada ise 20, 10, 8 ve 18 değerleri içinde en büyük 20 değeri olduğundan 20 değeri alınmaktadır. Genel ortalama havuzlamada ise tüm değerler toplanır ve aritmetik ortalaması alınır. Toplam değeri olan 277 sayısı 16'ya bölündüğünde yaklaşık 17 değeri elde edilmiş olacaktır.



Şekil 2.15: Havuzlama Yöntemlerinden En Çok Kullanılan 3 Yöntemin Örneği [28]

2.5.1.4. Düzleştirme Katmanı

Modelde son sırada olan havuzlama katmanından ya da evrişim katmanından gelen sonuç matrisleri vektörlere çevrilerek tam bağlı katmana sınıflandırma amacıyla gönderilir.

2.5.1.5. Kayıp Fonksiyon

CNN modelinde eğitim sonunda tahmin edilen ile gerçek değer arasındaki hatayı tespit edebilmek için çıktı katmanında kayıp fonksiyonlar kullanılır. Daha sonra CNN buna göre optimize edilir. Bu fonksiyonlar tahmin edilen değer ve gerçek değer olmak üzere iki parametre alırlar.

Softmax ya da Çapraz Entropi: *CNN* modelin performansını ölçmek için kullanılır. Çıktısı olasılıktır. $p \in \{0,1\}$ Genellikle çok sınıflı sınıflandırma problemlerinde kare hata kaybı fonksiyonun yerine kullanılır [28]. Denklem (2.6) çıkış sınıfının olasılığının matematiksel gösterimidir.

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \tag{2.6}$$

Denklem (2.6)'da N, çıktı katmanındaki nöron sayısını belirtir. e^{a_i} ise önceki katmandan gelen çıktıyı göstermektedir.

$$H(p,y) = -\sum_i y_i \, log(p_i) \qquad i \in [1,N] \tag{2.7}$$

Denklem (2.7)'de çapraz entropi kayıp fonksiyonun matematiksel gösterimi yer almaktadır.

Öklid Kayıp Fonksiyonu: Bu fonksiyon, tahmin edilen ile gerçekte olan arasındaki hatayı hesaplamak için genellikle regresyon problemlerinde yaygın olarak kullanılır. Ortalama kare hatası olarak da geçer [28]. Öklid kayıp fonksiyonuna ait matematiksel gösterimi Denklem (2.8)'de verilmiştir.

$$H(p,y) = \frac{1}{2N}\sum_{i=1}^{N}(p_i - y_i)^2 \tag{2.8}$$

2.5.1.6. Geri Yayılım

Geri yayılım(backpropagation) işleminin amacı öğrenme işlemini gerçekleştirmektir. Öğrenme işleminde ana amaç hata değerini en aza indirmektedir. Hata değerini en aza indirmek için optimizasyon algoritmaları kullanılır. Hesaplama sonucunda elde edilen değerle geriye doğru giderek ağırlıklar güncellenir. Bu hesaplama işleminde türev kullanıldığından Bölüm

2.5.1.2'de belirtilen aktivasyon fonksiyonlarından veri setine uygun olanı seçmek gerekir. Çünkü bazı aktivasyon fonksiyonların negatif değerlerinde türev sıfır olacaktır.

2.5.1.7. Gradyan İniş Algoritması (Gradient Descent Algorithm)

Modelleme esnasında amaç eğitim hatasını en aza indirmektir. Bu algoritma, her eğitim döneminde (epoch) ağ parametrelerine göre birinci dereceden türev uygulayarak fonksiyonun gradyanını hesaplar. Daha sonra parametreler gradyanın (eğimin) tersi yönünde güncellenir. Bu güncelleme işlemi her nörondaki gradyanın önceki katmandaki nöronlara geri yayılım (back propagation) yoluyla yapılır. Bu işleme ait matematiksel gösterim Denklem (2.9)'da gösterilmiştir [28].

$$w_{ij^t} = w_{ij^{t-1}} - \Delta w_{ij^t} \quad , \quad \Delta w_{ij^t} = \eta * \frac{\partial E}{\partial w_{ij}} \tag{2.9}$$

Gradyan iniş temelli algoritmalar [32]:

1. ***Toplu Gradyan İniş (Batch Gradient Descent - BGD):*** Toplu gradyan iniş yönteminde eğitim setinin tamamının gradyanı hesaplanır ve parametreler bu gradyana göre bir kez güncellenir. Toplu gradyan iniş her bir parametre için yeniden hesaplama yapacağından dolayı büyük miktarda veri kullanılması sonucunu doğurması sebebiyle büyük verilerde yavaş çalışacaktır. Algoritmanın matematiksel gösterimi Denklem (2.10)'da gösterilmektedir. Denklemde θ güncellenecek parametre vektörünü, η öğrenme katsayısını, $J(\theta)$ maliyet fonksiyonunun gradyanını temsil etmektedir [33].

$$\theta = \theta - \eta * \nabla_\theta J(\theta) \tag{2.10}$$

2. ***Stokastik Gradyan İniş (Stochastic Gradient Descent - SGD):*** Stokastik(Rastgele) gradyan iniş algoritması, her eğitim seti için bir güncelleme yapacağından dolayı toplu gradyan inişe göre daha hızlı çalışacaktır. SGD'nin matematiksel gösterimi Denklem (2.11)'de verilmektedir. Denklemde yer alan $x^{(i)}$ eğitim setindeki veriyi, $y^{(i)}$ise etiketi temsil etmektedir.

$$\theta = \theta - \eta * \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right) \tag{2.11}$$

3. ***Mini Toplu Gradyan İniş (Mini-Batch Gradient Descent)*:** Mini toplu gradyan iniş algoritması hem SGD'nin hem de BGD'nin en iyi özelliklerini almıştır. Bu algoritmada

n eğitim seti için güncelleme yapılacaktır. Böylece daha istikrarlı bir yakınsama sağlanarak güncellemelerin varyansı azaltılır. Ayrıca gradyan hesaplaması verimli hale getirilmiş olur. Batch-size boyutu genellikle 50 ile 256 arasında olur. Fakat bu farklı uygulamalarda farklı değerler alabilir. Denklem (2.12)'de matematiksel formu gösterilmektedir [33].

$$\theta = \theta - \eta * \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{2.12}$$

Yapay sinir ağlarında başarılı sonuçlar elde etmek için maliyet fonksiyonun değeri sıfıra yakın olmasıyla sağlanabilmektedir [32]. Çünkü bu sayede gerçek değer ile tahmin edilen değer arasında farkın az olması sağlanmış olur. Bu da tahmin edilen değerlerin doğru tahminler olduğunu gösterir.

2.5.1.8. Optimizasyon Algoritmaları

Gradyan iniş algoritmalarında bazı zorluklar [33] bulunmaktadır. Bu zorlukları yenmek için birçok optimizasyon algoritmaları geliştirilmiştir. Bunlardan bazıları:

1. ***Momentum***: eğimi aynı yönde olduğu sürece güncelleme hızı artarken eğim ters yönde olduğunda güncelleme hızı yavaşlar. Bunun sonucunda daha hızlı yakınsama ve daha az salınım elde edilmiş olur. Algoritmanın matematiksel formu Denklem (2.13)'te verilmiştir. Denklemde yer alan γ terimi momentum değeridir ve çoğunlukla 0.9 veya benzeri bir değer alınır [33].

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t \tag{2.13}$$

   Momentum önceki parametreleri güncelleme işleminde sadece o iterasyondaki gradyanı değil önceki gradyanları da dâhil eder [32].

2. ***AdaGrad:*** Önceki algoritmalar tüm parametrelerin güncellenmesi aşamasında aynı öğrenme oranı kullanırlar. Önceki algoritmalarda bu oran genellikle 0.01 gibi sabit bir değer olmasından dolayı meydana gelen problem, bu yöntem sayesinde farklı öğrenme oranları kullanarak ortadan kalkmış olmaktadır. Algoritmanın matematiksel gösterimi Denklem (2.14)'te görünmektedir [33].

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{2.14}$$

Denklemdeki G_t; geçmiş gradyan bilgisi saklayan, parametre sayısı ile aynı olan köşegen bir matristir. Bu matristeki her bir eleman ilgili parametrenin önceki gradyanlarının karelerinin toplamını gösterir. ϵ değeri ise paydanın 0 (sıfır) olmasını engellemek için küçük bir sayıdır. Yaklaşık değeri 1e-8'dir [32].

AdaGrad algoritmasının olumsuz yanı, terimlerin pozitif değerlerinde paydadaki gradyanın büyümesinden dolayı öğrenme oranın çok küçülmesidir. Bu da öğrenme oranın düşmesine belli bir noktada ise çok küçük olmasına sebep olacaktır.

3. ***Adadelta:*** AdaGrad algoritmasında zamanla öğrenme katsayısı sıfıra yaklaşmaktadır. Ayrıca başlangıçta öğrenme katsayısı parametresi olmalıdır. Adadelta bu iki durumu da çözmektedir [32].

4. ***Adam (Adaptive Moment Estimation*)**: Derin sinir ağlarını eğitmek için tasarlanmıştır. Hafıza verimliliği ve hesaplama gücünün fazla gerekmemesi algoritmanın en önemli avantajlarındandır [28]. Her parametre için adaptif öğrenme oranlarını hesaplar. Geçmiş kareli gradyanların (v_t) üssel azalan ortalaması tutulur. Adam yöntemi ayrıca geçmiş gradyanların (m_t) üssel azalan ortalamasını saklar. m_t'nin ve v_t'nin hesaplanması Denklem (2.15)'te ve (2.16)'da gösterilmektedir [33].

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.15}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.16}$$

m_t gradyanların birinci momentini (ortalama), v_t ise ikinci momentin (merkezsiz varyans) tahminleridir. Bu iki terim 0 vektörü olarak başlatıldıklarında ilk başlangıç zaman adımlarında ve özellikle sönüm (decay) oranı küçük olduğunda 0 (sıfır)'a doğru eğilimli oldukları belirtilmiştir. Denklem (2.17)'de ve Denklem (2.18)'de hesaplanan düzeltilmiş yanlı birinci ve ikinci moment tahminleri bu eğilime karşı olmaktadır [33].

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.17}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.18}$$

Bu değerler Denklem (2.19)'da parametre güncellemek için kullanılırlar.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.19}$$

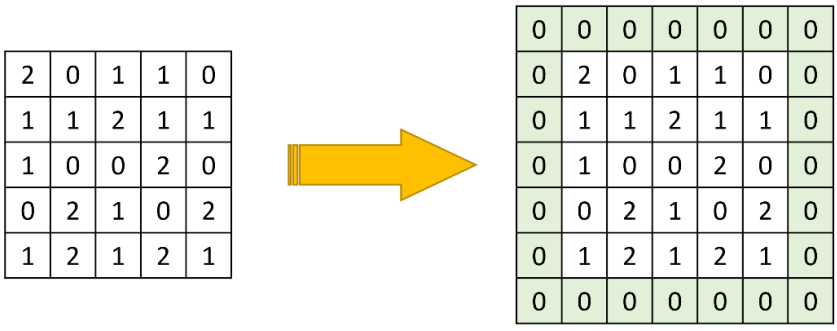
β_1 için 0.9, β_2 için 0,999 ve ϵ değeri için 10^{-8} değerleri önerilmektedir [33].

2.5.1.9. Hiper Parametreler

Hiper parametreler model için ayarlanması gereken, eğitim işleminin başarısını doğrudan etkileyen parametrelerdir. Aşağıda evrişimli sinir ağlarında ayarlanması gereken parametreler yer almaktadır.

- ✓ ***Doldurma (Padding):***

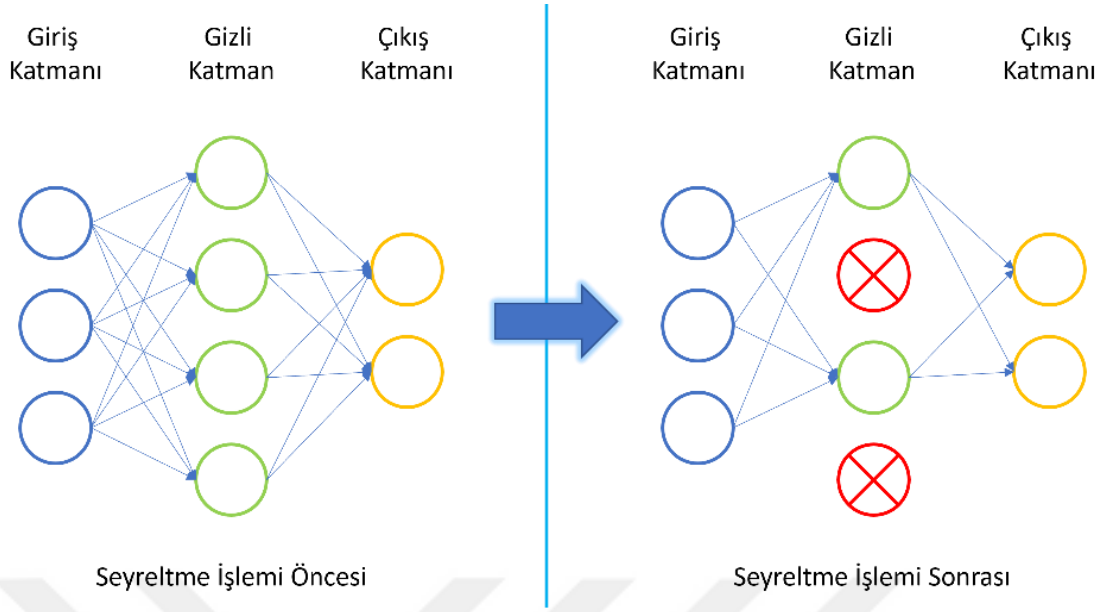
Doldurma, giriş katmanın dış tarafının doldurulması ile ilgilidir. Evrişim katmanından sonra çıkış verilerinde bir miktar azalma olur. Bu bazı verilerin hızlıca yok olması anlamına gelir. Bu problemin çözümü olarak doldurma işlemi kullanılır. Ayrıca bu işlemi yapmak performansı da arttırmaktadır[31]. Doldurma yöntemlerinden biri de 0 eklemektir. Şekil 2.16'de örnek bir doldurma işlemi görülmektedir.



Şekil 2.16: Doldurma İşlemi

- ✓ ***Seyreltme (Dropout):***

Genelleme için kullanılan bir yöntemdir. Her eğitim döneminde verilen oranda nöronlar rastgele atılır. Bu işlem ayrıca modeli farklı bağımsız özellikleri öğrenmeye zorlayacaktır. Eğitim sürecinde seyreltme işlemiyle atılan nöronlar ileri ya da geri yayılımda yer almayacaklardır [28]. Seyreltme öncesinde ve sonrasında katmanlardaki değişikliğin bir parçası Şekil 2.17'de görülmektedir.



Şekil 2.17: Seyreltme İşlemi Öncesi ve Sonrası

✓ ***Çekirdek Boyutu (Kernel Size):***

Çekirdek diğer adıyla filtre, ayrık sayı ya da değerlerden oluşan bir matristir. Bu değerlere çekirdek ağırlığı denir. Çekirdek, giriş matrisinin üzerinde kaydırılarak bazı bilgiler elde edilmeye çalışılır. CNN eğitim başlangıcında çekirdek ağırlıkları rastgele belirlenebilir. Eğitim sürecinde bu ağırlıklar güncellenerek öğrenme gerçekleşir [28].

✓ ***Öğrenme Oranı (Learning Rate):***

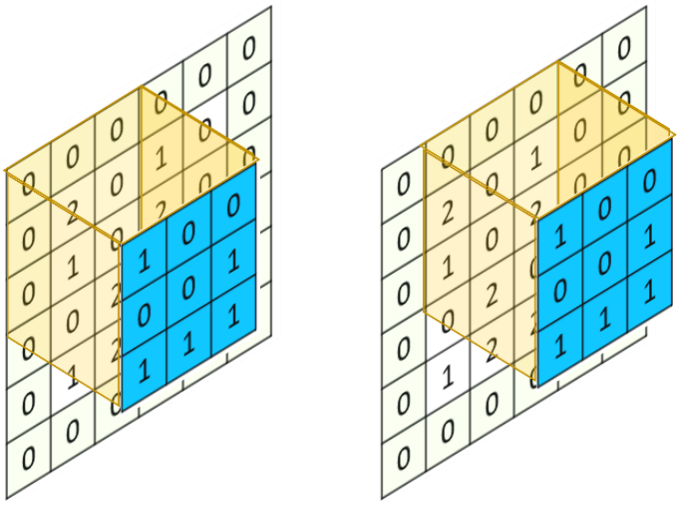
Parametre güncellemesinin adım boyutu olarak tanımlanır. Bu oranın eğitimin başarı düzeyine etkisi olduğundan doğru bir şekilde seçilmesi gerekir.

✓ ***Dönem Sayısı (Epoch):***

Eğitim setindeki tüm verilerin gözden geçirilme sayısıdır. İdeal dönem sayısı belirlenmesi performans ve başarı açısından önemlidir.

✓ ***Kaydırma (Stride):***

Kaydırma işlemi çekirdek (filtre) matrisinin giriş matrisi üzerinde kaydırma miktarı olarak tanımlanabilir. Örnek kaydırma işlemine ait şema Şekil 2.18'da görülmektedir.



Şekil 2.18: 1 Adımlı Kaydırma İşlemi

Kaydırma adımını büyük seçmek çıkış matrisinin boyutunu azaltacaktır. Daha az bilgi çıkarılmasına sebep olacaktır. Kaydırma adımını küçük seçmek ise girişten daha fazla bilgi seçilmesine ve daha derin ağ oluşturulmasını sağlayacaktır. Kaydırma işlemi sonucunda elde edilen çıkışın hesaplanabileceği formül Denklem (2.20)'de verilmiştir.

$$W_2 = \frac{W_1 - F + 2P}{s} + 1$$
$$H_2 = \frac{H_1 - F + 2P}{s} + 1 \qquad (2.20)$$
$$D_2 = K$$

Denklemde yer alan giriş matrisinin genişliği: W_1, yüksekliği: H_1 ve derinliği: D_1 olarak belirtilmiştir. F değeri çekirdeğin (filtrenin) boyutu, P ise doldurma değeri ve K değeri ise çekirdek sayısıdır.

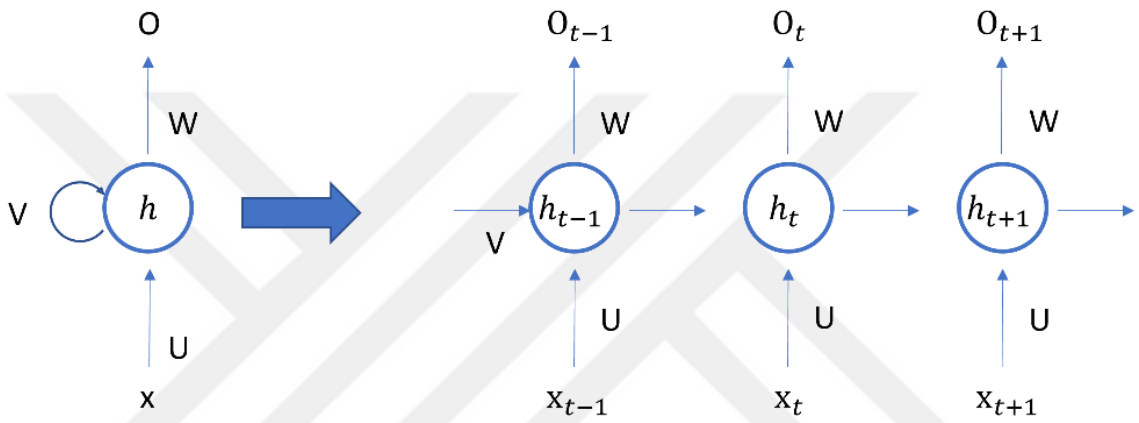
Örnek olarak Şekil 2.13'teki filtreme işlemi dikkate alındığında giriş matrisi (W_1xH_1xD_1) 5x5x3, filtre boyutu (F) 3x3x3, 2 adet filtre (K) uygulanmış, kaydırma (S) olarak 2 seçilmiş ve 1 kez doldurma işlemi (P) yapılmıştır.

$$W_1 = \frac{5 - 3 + 2.1}{2} + 1 = 3 \qquad H_2 = \frac{5 - 3 + 2.1}{2} + 1 = 3 \qquad D_2 = 2$$

Sonuç olarak 3x3x2 boyutlu çıkış matrisi elde edilecektir [31].

2.5.2. Tekrarlayan Sinir Ağları (RNNs)

Tekrarlayan sinir ağları (RNNs), sıralı bilgiyi işlemek üzere oluşturulmuş bir derin öğrenme yöntemidir. Her birimde sıralı bilgi üzerinden işlemler yapılır. Ayrıca önceki birimden gelen bilginin de işlemlere dahil edilerek sonuçlar üzerinde etkisi olmuş olur. Bu yöntemden dolayı tekrarlayan ifadesi kullanılmaktadır. Ayrıca önceki bilgilerin kullanılması durumu bellek olarak ifade edilir. Özetle RNN'ler o anda işlenmekte olan bilgiyi belleğinde yer alan önceki bilgiler ile de işleyerek sonuç üretmeye çalışan bir yöntemdir diyebiliriz [34]. Örnek olarak Şekil 2.19'de basit bir tekrarlayan sinir ağı görülmektedir.



Şekil 2.19: Basit Bir Tekrarlayan Sinir Ağı [34]

2.5.3. Kapsül Ağlar

Bir nesnenin ya da bir nesne parçasının örnekleme parametrelerini temsil eden bir nöron grubu olarak tanımlanmıştır. Kapsül ağlarda bir varlığın olduğunu temsil etmek için aktivite vektörünün uzunluğu ve örnekleme parametrelerini temsil etmek için yönü kullanılır. Aktif kapsüldeki nöronların aktiviteleri, görüntüde bulunan belirli bir varlığın çeşitli özelliklerini temsil ettiği belirtilmiştir. Bu özelliklere örnek olarak yön, konum, hız, ton vb. verilebilir. Kapsül ağlar; giriş, evrişimli katman, öncül kapsüller, digit kapsül ve tam bağlantılı katmandan oluşmaktadır [35].

2.6. MEVCUT ÇALIŞMALAR

Log kayıtlarının makine öğrenmesi yöntemleri ile incelenerek sonuç üretilmesi ile ilgili çok fazla çalışma yer almaktadır. Bu çalışmalarda farklı birçok veri seti kullanılmıştır. Yaygın kullanılan bazı veri setlerine DARPA'98, KDDCUP'99, NSL-KDD, CSIC 2010 HTTP, ISCX 2012, UNSW-NB15, NGIDS-DS 2016 ve CICIDS 2017 veri setleri örnek olarak verilebilir. Veri seti kullanarak üretilen loglar makine öğrenmesi ve/veya derin öğrenme yöntemleri

kullanılarak zararlı davranışlar tespit edilmeye çalışılmaktadır. Log analizi ile normal olmayan girişlerin tespit edilmesiyle ilgili yapılmış birçok çalışma olduğundan dolayı tezde yer alan yöntemleri ve veri setini içeren çalışmalara yer verilmiştir.

Jia ve diğerleri çok kaynaklı günlük kayıtlarında büyük veri analizi gerçekleştirerek izinsiz giriş davranışlarının tespitini yapmışlardır. Makine öğrenmesi, veri madenciliği ve istatistiksel analiz yöntemlerinin entegrasyonu ile bir çerçeve önerilmiş ve etkili olduğu belirtilmiştir [36].

Gharibeh ve diğerleri, CSIC 2010 HTTP veri seti kullanarak web uygulama isteklerinin sınıflandırılması çalışmasını yapmışlardır. SVM, DT, LR ve RF makine öğrenme yöntemleri kullanılmış ve sırasıyla 0,99754, 0,99181, 0,98641 ve 0,98166 doğruluk elde etmişlerdir [37].

Wan ve diğerleri, CSIC 2010 HTTP veri setini kullanarak web erişim günlük kayıtlardan anomali durumların tespitini yapmaya çalışmışlardır. Özellik çıkarmak için Word2Vec yöntemi ve TextCNN modeli kullanılmış son sınıflandırma olarak topluluk öğrenme (Ensemble Learning) kullanılmıştır. ELSV yöntemiyle 0,9933 doğruluk elde edildiği belirtilmiştir [38].

Yan ve Xiong, Web-APT tespitini yapabilmek için makaledeki çalışmayı yapmışlardır. Çalışmasında giriş dizesi ile dikkat tabanlı çeviri sonucu üretilen dizi karşılaştırılarak anomali istekler tespit edilmeye çalışılmıştır. Bu karşılaştırma BLEU skoruna göre yapılmaktadır. Veri seti olarak CSIC 2010 HTTP kullanılmış ve 0,9844 F1 skoru elde edilmiş [39].

Yu ve diğerleri, zararlı web istekleri için genişletilmiş TextCNN yöntemi önermişlerdir. Veri seti olarak CSIC 2010 HTTP kullanılmıştır. Derin öğrenme ve istatistiksel yöntemler birlikte kullanılarak başarının artırılması amaçlanmıştır. Evrişimli sinir ağının son katmanını SVM ile değiştirerek web isteklerinin sınıflandırma işlemi tamamlanır. Süreç sonunda oluşturulan farklı yöntemlerle %98,44 ve %99,33 oranında doğruluk elde edilmiştir [40].

Bhatnagar ve diğerleri, ML yöntemleri kullanarak web uygulamasına izinsiz girişlerin tespitleri ile ilgili bir çalışma yayınladılar. Veri seti olarak CSIC 2010 HTTP ve ECML/PKDD 2007 kullanmışlardır. RF, Çok terimli NB sınıflandırıcı, MLP, DT yöntemleri kullanılmıştır. Tüm modellerin ortalama doğruluğu %99,3 olarak hesaplanmıştır. Modellerin ayrı ayrı doğrulukları sırasıyla %99,85, %99,60, %99,27 ve %98,89 olarak belirtilmiştir [41].

Jemal ve diğerleri LSTM ile CNN'i birleştirerek daha fazla bilgi edinilebileceğini belirtmişler ve hafızalı CNN modeli yani M-CNN'i önermişlerdir. CSIC 2010 HTTP veri seti üzerinde 3 farklı M-CNN yaklaşımı kullanılmış olup %99,248, %97,357 ve %96,961 doğruluk elde etmişlerdir [42].

Seyyar ve diğerleri NLP, BERT ve Derin öğrenme temelli olan MLP ile bir saldırı tespit çerçevesi (framework) önermişlerdir. Veri seti olarak CSIC 2010 HTTP, FWAF ve httpParams veri setlerinin birleştirilmesinden oluşturulmuş özel bir veri seti kullanılmıştır. Önerilen mimarinin %99,98 doğruluğa sahip olduğunu belirtmişlerdir. Ayrıca CNN tabanlı geliştirilen modelin önerdikleri modelden %95,79 gibi bir oranla daha düşük sonuç elde ettiklerini belirtmişlerdir [7].

Das ve diğerleri, NLP ve topluluk öğrenme yöntemini kullanarak NLPIDS adında yeni bir model önermişler ve bu modeli kullanarak izinsiz giriş tespiti yapmaya çalışmışlardır. Veri seti olarak CSIC 2010 HTTP kullanılmıştır. Önişleme sonrasında Doc2vec yöntemi ile veriler işlenmiştir. İşlenen veriler ilk olarak LR, SVM, NB, DT ve NN kullanılarak sınıflandırılmıştır. Daha sonra bu sonuçlar ikinci sınıflandırıcıya girdi olarak verilir. İkinci sınıflandırma modelleri Ens_MV (Çoğunluk Oylama), Ens_LR (Lojistik regresyon), Ens_NB (Naive Bayes), Ens_NN (Yapay Sinir), Ens_DT (Karar Ağaçları) ve Ens_SVM (Destek vektör makinesi) olarak belirlenmiştir. Elde edilen sonuçlar sırasıyla 0,9916, 0,9802, 0,9994, 0,9644, 0,9990, 0,9987, 0,9634, 0,9986, 0,9995, 0,9995, 0,9995 olarak belirtilmiştir. İlk sınıflandırma en düşük DT'de en yüksek doğrulukta NN'de elde edilmiştir. Nihai sınıflandırmada ise en düşük doğruluk Ens_LR'de en iyi sonuç ise Ens_DT'de elde edilmiştir [43].

Niu ve Li, web saldırılarının tespiti için CNN-GRU model tabanlı yüksek performanslı bir çalışma gerçekleştirmişlerdir. Veri arttırma için istatistiksel özellik çıkarım işlemi yapılmıştır ve veri segmentasyonuyla birlikte word2vec'e girdi olarak verilmiştir. Kelime gömme işleminden sonra CNN ve GRU'yu birleştiren modele veriler gönderilmiştir. Çalışmada CSIC 2010 HTTP veri seti kullanılmıştır. Öncelikle geleneksel makine öğrenme yöntemleri ile deneme yapılmış KNN için %91,88, RF için %84,51, GB için %83,02 DT için %81,84, LR için %74,07 ve NB için %66,75 doğruluk elde edilmiştir. Veri artırma işlemi olmadan derin öğrenme yöntemleri kullanıldığında CNN için %93,44, LSTM için %95,70, GRU için %97,17, CNN-LSTM için %97,40 ve önerilen model olan CNN-GRU için %97,56 doğruluk elde

edilmiştir. Veri artırma işlemi sonrasında önerilen model için doğruluk oranı %99,00 olduğu belirtilmiştir [44].

Wu ve Wang, HTTP payload (yük) yani zararlı kodları tespit edebilmek için dikkat tabanlı kodlayıcı-kod çözücü RNN (AEDRAD) modelini önermişlerdir. Önerilen sistem simgeleştirme, kodlama-kod çözme, dikkat mekanizması, anomali tespit ve dikkat görselleştirme adımlarından oluşmaktadır. Veri seti olarak CSIC 2010 HTTP ve Vulnbank kullanılmıştır. Tez kapsamında Vulnbank sonuçları göz ardı edilmiştir. AEADRAD deney sonucunda ROC_AUC değeri 0.9527 ve PR_AUC değeri 0,9157 olarak hesaplanmıştır [45].

Ito ve Iyatomi, karakter seviyeli CNN kullanarak web uygulama güvenlik duvarı yaklaşımı önermişlerdir. Ön işleme adımında URL kod çözme (URL decoding) ve Unicode kodlama işlemi gerçekleştirilmiş. Daha sonra her bir isteğin 1000 karakter içerdiği varsayılmış. Yetersiz olan isteklerde 0 (sıfır) olarak doldurulmuştur. Her bir karakter gömme işleminde 128 boyutlu bir vektöre dönüştürülür böylece CNN'e gönderilecek giriş matrisi 128 x 1000 olmuş olur. 2 tür CNN modeli önerilmiştir. A modeli %98,8 ve B modeli %98,2 doğruluk elde edilmiştir [46].

Işıker ve Soğukpınar, Makine öğrenmesi tabanlı bir web uygulama güvenlik duvarı önermişlerdir. Karakter n-gram ve lineer destek vektör makinesi (SVM) kullanarak oluşturulan model en iyi sonucu 1-6 n-gramda verdiği belirtilmiştir. Kelime n-gram'da %95,08 ve önerilen model olan karakter n-gram'da %99,53 oranında bir doğruluk elde edildiği sunulmuştur [47].

Tong ve diğerleri, içerik tabanlı spam tespiti için uzun kısa dikkatli bir kapsül ağı önermişlerdir. TextCNN, LSTM ve BERT yöntemlerinden daha iyi sonuç aldıkları belirtilmişlerdir. Dengesiz veri setinde %98,72 dengeli veri setinde %99,30 gibi bir doğruluk elde etmişlerdir [48].

Ouyang ve diğerleri, kapsül ağ tabanlı tekrar saldırılarını tespit edecek bir sistem öneriyorlar. Çalışmanın sonunda tekrar saldırılarını başarıyla tespit ettiklerini belirtmişlerdir [49].

Huang ve diğerleri kapsül tabanlı sinir ağı kullanarak oltalama url tespiti yapmaya çalışmışlardır. Alexa, Phishtank ve Openphish'ten toplanan veriler ile bir veri seti oluşturulmuş. Deney sonunda %99,63'lük bir başarı elde edilmiştir [50].

İlgili çalışmalar incelendiğinde web isteklerinin olduğu günlük kayıtlarının Kapsül Ağ yaklaşımıyla ilgili herhangi bir çalışma olmadığı tespit edilmiştir [51] [52]. Tez kapsamında web günlük kayıtlarının kapsül ağ yaklaşımıyla analiz edilerek anomali tespiti yapılacaktır.

Tablo 2.2: Veri Seti ya da Kapsül Ağ ile ilgili çalışmalar

K. NO	YAZARLAR	YÖNTEM	YIL	VERİ SETİ	BAŞARIM
37	Gharibeh ve diğerleri	SVM,DT,LR,RF	2020	CSIC 2010 HTTP	0.99754, 0.99181, 0.98641 ve 0.98166
38	Wan ve diğerleri	ELSV	2021	CSIC 2010 HTTP	0,9933
39	Yan ve Xiong	BLEU	2020	CSIC 2010 HTTP	0,9844
40	Yu ve diğerleri	TextCNN+SVM	2020	CSIC 2010 HTTP	0,9933
41	Bhatnagar ve diğerleri	RF, NB,MLP,DT	2022	CSIC 2010 HTTP ve ECML/PKDD 2007	0,9985, 0,9960, 0,9927, 0,9889
42	Jemal ve diğerleri LSTM	CNN,LSTM, MCNN	2020	CSIC 2010 HTTP	0,9812, 0,89821, 0,99248
7	Seyyar ve diğerleri	CNN, BERT+MLP	2022	CSIC 2010 HTTP, FWAF ve httpParams	0,9579, 0,9998
43	Das ve diğerleri	LR, SVM, NB, DT, NN, Ens_MV, Ens_LR, Ens_NB, Ens_NN, Ens_DT ve Ens_SVM	2020	CSIC 2010 HTTP	0,9916, 0,9802, 0,9994, 0,9644, 0,9990, 0,9987, 0,9634, 0,9986, 0,9995, 0,9995, 0,9995
44	Niu ve Li	KNN, RF, GB, DT, LR, NB, CNN, LSTM, GRU, CNN-LSTM, CNN-GRU CNN-GRU(Veri Arttırma)	2020	CSIC 2010 HTTP	0,9188, 0,8451, 0,8302, 0,8184, 0,7407, 0,6675, 0,9344, 0,9570, 0,9717, 0,9740, 0,9756, 0,9900
45	Wu ve Wang	RNN (AEDRAD)	2021	CSIC 2010 HTTP ve Vulnbank	0,9527
46	Ito ve Iyatomi	CNN(A model), CNN(B model)	2018	CSIC 2010 HTTP	0,988, 0,982
47	Işıker ve Soğukpınar	SVM(n gram), SVM(1-6 n gram)	2021	CSIC 2010 HTTP	0,9508, 0,9953
48	Tong ve diğerleri	CapsNet-LSA	2021	Chinese spam Dataset (Trec06)	0,9872
50	Huang ve diğerleri	URLNET, Kapsül Ağ	2019	Alexa, Phishtank ve Openphish	0,9407, 0,9966

3. YÖNTEM

Bu bölümde bu tez kapsamında gerçekleştirilen çalışmada kullanılan yöntemlere yer verilmiştir. Daha sonra uygulama gerçekleştirilmesi ile ilgili bilgi verilecek olup sonraki adımda bulgular verilip değerlendirilecektir. Derin öğrenme yöntemlerinden biri olan kapsül ağlar, derin öğrenme yöntemlerinden biri olması, daha önce HTTP log analizinde kullanılmamış olması, nasıl bir performans sergileyeceğinin görülmek istenmesi gibi durumlarından dolayı öğrenme yöntemi olarak seçilmiştir. Eğitim ve model oluşturma aşaması Şekil 3.1'de görülmektedir.



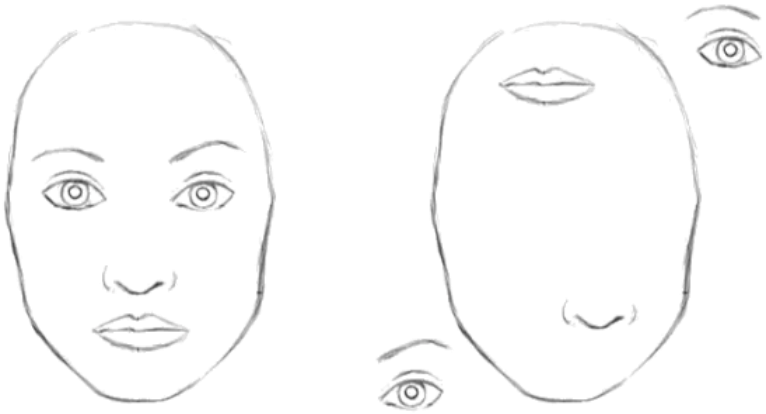
Şekil 3.1: Eğitim Aşaması

3.1. KAPSÜL AĞLAR

Kapsül ağlar ilk olarak 2011 yılında Hinton ve diğerleri tarafından "Transforming Auto-Encoders" konulu makalede önerilmiştir [53]. Daha sonra 2017 yılında "Dynamic Routing Between Capsules" konulu makalede ikinci türü yayınlanmıştır [35]. Son olarak 2018 yılında "Matrix capsules with EM routing" konulu makale yayınlanmıştır [54]. Patrick ve diğerlerine göre kapsül ağların en popüler versiyonu anlaşmalı yönlendirmedir (routing by agreement) [52].

3.1.1. Temel Yapı

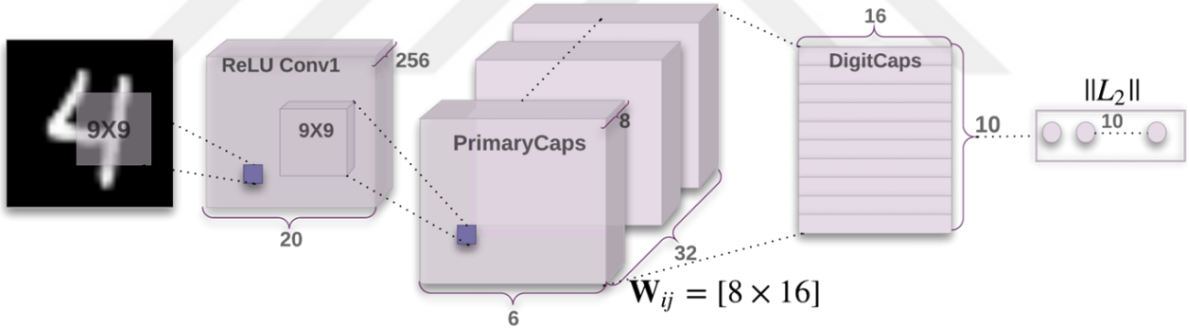
Derin öğrenme yöntemlerinden biri olan kapsül ağ, CNN yönteminin sıkıntılarını çözmek için bulunmuş yeni bir yöntemdir. CNN yönteminde maksimum havuzlama aşamasında bazı özelliklerin kaybolduğu bilinmektedir. Ayrıca CNN'de yön bilgisi bulunmadığından dolayı olması gerekenden fazla iyimser davranmaktadır. Örneğin Şekil 3.2'de görülebileceği gibi bir insan yüzünde göz, burun vb. yer değiştirdiğinde artık bir yüzden bahsedilemez. Fakat CNN bunu yüz olarak algılayacaktır. Çünkü nesnelerin konumları dikkate alınmamaktadır. Bu sebeple Hinton CNN'de kullanılan havuzlama katmanını eleştirmiştir [55].



Şekil 3.2: CNN için benzer olan iki resim [55]

Yukarıdaki sorunun çözümü için Hinton, sınıflandırma ve nesne tanıma işleminin doğru bir şekilde yapılabilmesi için yön, konum vb. diğer özelliklerin saklanmasını savunduğu belirtilmiştir [55]. Bu savunma kapsül ağlarının temelini oluşturmaktadır.

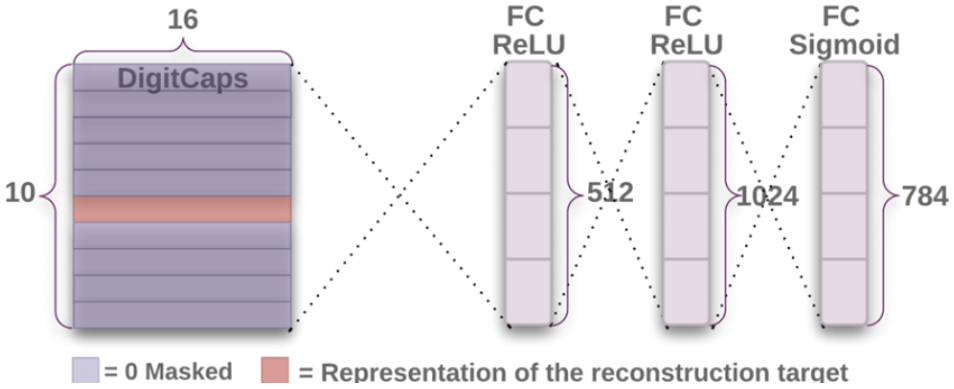
Kapsül ağlar Şekil 3.3'te belirtildiği gibi temelde; Evrişim Katmanı, Öncül Kapsül (Primary Capsule), Digit Kapsül ve Tam Bağlantılı Katman'dan oluşmaktadır [35].



Şekil 3.3: Temel Kapsül Ağ [35]

Kapsül, nöronlar grubudur. Bir kapsülün girdisi evrişimli ağdan gelen çıktıdır. Kapsül çıktı olarak nesnedeki bir özelliğin olma olasılığının ve genellikle örnekleme parametreleri (Instantiation Parameters) olarak adlandırılan vektör dizisidir [52]. Örnekleme parametreleri; poz, doku, konum, ölçek gibi özellikleri temsil eder [56]. İlk 3 katman kodlayıcı diğer 3 katman ise kod çözücü olarak görev yapmaktadır. Özellik çıkarma işlemi evrişimli katman tarafından yapılır. Çıkarılan özelliklere ait tensör öncül kapsül katmanına (Primary Caps Layer) girdi olarak gönderilir. Öncül kapsül katmanı çıkarılan temel özellikleri alır ve onların kombinasyonlarını üretmek için işlemler yapar. Sonuç olarak bir tensör çıkarılır ve DigitCaps katmanına gönderilir. Daha sonra DigitCaps tarafından işlenen tensör kod çözücü olarak

adlandırılan tam bağlantılı katmana gönderilir. Temel yapıda DigitCaps'ten çıkış 16x10'luk bir matristir. Tam bağlantılı katmanlar Şekil 3.4'te görülmektedir [52] [55].



Şekil 3.4: CapsNet Tam Bağlantılı Katman [35]

3.1.2. Ezme İşlemi

Kapsül ağlarda ezme (squash) önemli işlemlerden biridir. Ezme fonksiyonu ile kapsül çıktısının 0 ile 1 arasında olması sağlanmaktadır. Denklem (3.1)'de, bir alt katmandaki kapsülün çıkışı olan u_i ile ağırlık matrisi olan W_{ij}'nin çarpılmasıyla öncül kapsülün tahmin vektörü $\hat{u}_{j|i}$ elde edilmiş olur.

$$\hat{u}_{j|i} = W_{ij} u_i \tag{3.1}$$

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{3.2}$$

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \tag{3.3}$$

Denklem (3.2)'de iteratif dinamik yönlendirme işlemi tarafından belirlenen eşleme katsayısının $\hat{u}_{j|i}$ ile çarpımının toplamından s_j elde edilir. Denklem (3.3)'te ise ezme fonksiyonun denklemi görülmektedir. Denklemde ilk kesir ek ölçekleme yaparken ikinci kesir işlemi birim ölçekleme yapar. Burada s_j kapsülün toplam girdisi iken v_j kapsülün çıktısı olmaktadır. Doğrusal olmayan ezme işlemiyle bir kapsülün çıkış uzunluğunun, kapsül tarafından temsil edilen varlığın o anki girdide olma olasılığını temsil edilmesi sağlanmaktadır. Uzunluğu kısa olan vektörler sıfıra, uzun olan vektörler ise 1'e çok yakın olacak şekilde olması sağlanır [35] [52] [55].

3.1.3. Kayıp Fonksiyon

Bir kapsülün nesnesinin var olma olasılığını temsil etmek için örnekleme vektörünün uzunluğu kullanılmış. Bir varlık, üst düzey DigitCaps'te var olması durumunda sonucunun 1 olması istenir. Denetimli öğrenmede her bir değerin etiketi olduğu için ilgili nesne varsa 1 ve diğer vektörler varsa 0 olacaktır. Toplam kayıp, tüm basamaklardaki kayıpların toplamı olacaktır. Kayıp fonksiyona ait Denklem (3.4)'te verilmiştir.

$$L_k = T_k \ max(0, m^+ - \|v_k\|)^2 + \lambda(1 - T_k)\, max(0, \|v_k\| - m^-)^2 \tag{3.4}$$

Denklemde; eğer doğru etiket k. kapsülde varsa $T_k = 1$, $m^+ = 0.9, m^- = 0.1$ olarak alınır. Kayıp sıfır olduğu durumda ise eğitimin kesintiye uğramaması için λ değerinden yararlanılır ve 0,5 olarak alınır [35].

3.2. VERİ SETİ

Benzer çalışmalarda sıklıkla kullanılan bir veri seti olan "HTTP Dataset CSIC 2010" bu tez kapsamında veri seti olarak kullanılacaktır. Eğitim seti güncel olmamasına karşın temelde yapılmak istenen yöntem için geçerli olduğu düşünülmektedir ve hâlâ kullanılan [56] bir veri seti olduğu görülmektedir. Ayrıca birçok makalede de kullanılan ve rahatça karşılaştırma yapılmasına imkân sağlayan bir veri setidir. "Spanish Research National Council (CISC) Information Security Institute" tarafından 2010 yılında üretilmiş bir veri setidir. Veri seti, [58] adresinden indirilebilir durumda olan 3 adet txt dosyasından oluşmaktadır. "normalTrafficTraining.txt" dosyası, normal isteklerin olduğu eğitim veri setidir. "normalTrafficTest.txt" dosyası, normal isteklerin olduğu test veri setidir. Son olarak "anomalousTrafficTest.txt" dosyası ise anomali yani zararlı isteklerin olduğu test dosyasıdır.

Veri seti dosyaları incelenerek çıkarılmış olan temel bilgiler Tablo 3.1'de verilmiştir. Dosyanın içeriği hakkında net bilgi edinebilmek için değerler çıkarılırken veri seti bütünlüğünde değil temel txt dosyası olarak değerlendirilmiş ve tablodaki değerler hesaplanırken dosyadaki her satır bağımsız olarak hesaplanmıştır.

Tablo 3.1: Veri Seti Dosya Temel Bilgileri

DOSYA ÖZELLİKLERİ	ANOMALİ TRAFİK (TEST)	NORMAL TRAFİK(EĞİTİM)	NORMAL TRAFİK (TEST)	TOPLAM
Karakter Sayısı	15.378.747	19.656.988	19.659.204	54.694.939
Dosya Boyutu	15,3 MB	19,6 MB	19,6 MB	54,6 MB
Kelime Sayısı	2.687.981	3.475.712	3.476.086	9.639.779
Satır Sayısı	355.777	492.001	492.001	1.339.779

Veri seti özelinde bakıldığında 36.000 normal eğitim verisi, 36.000 test eğitim verisi ve değişik saldırı türleri içeren 25.065 anomali veri içermektedir.

Veri setinin tam olarak ne içerdiğinin daha fazla anlaşılabilmesi amacıyla veri setinden seçilmiş 4 adet örnek Şekil 3.5, Şekil 3.6, Şekil 3.7 ve Şekil 3.8'de görülmektedir. Veri setindeki örnekler aktarılırken görsel açıdan rahat bir şekilde ayırt edilebilmesi amacıyla veriler ve öznitelikler sekme ile hizalanmıştır ve öznitelikler kalın olarak biçimlendirilmiştir. Orijinal veri seti incelendiğinde bu şekilde olmadığı görülecektir.

```
GET http://localhost:8080/tienda1/publico/anadir.jsp?id=1&nombre=Jam%F3n+Ib%E9rico&precio=39 &cantidad=41
    &B1=A%F1adir+al+carrito HTTP/1.1
User-Agent      : Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma          : no-cache
Cache-control   : no-cache
Accept          : text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain; q=0.8,image/png,*/*;q=0.5
Accept-Encoding : x-gzip, x-deflate, gzip, deflate
Accept-Charset  : utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language : en
Host: localhost : 8080
Cookie          : JSESSIONID=54E25FF4B7F0E4E855B112F882E9EEA5
Connection      : close
```

Şekil 3.5: Zararsız Bir Get İsteği

```
GET http://localhost:8080/tienda1/publico/anadir.jsp?id=2&nombre=Jam%F3n+Ib%E9rico&precio=85
    &cantidad=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos+WHERE+nombre+LIKE+%27%25
    &B1=A%F1adir+al+carrito HTTP/1.1
User-Agent      : Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma          : no-cache
Cache-control   : no-cache
Accept          : text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding : x-gzip, x-deflate, gzip, deflate
Accept-Charset  : utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language : en
Host: localhost : 8080
Cookie          : JSESSIONID=B92A8B48B9008CD29F622A994E0F650D
Connection      : close
```

Şekil 3.6: Zararlı Bir Get İsteği

```
                POST http://localhost:8080/tienda1/publico/anadir.jsp HTTP/1.1
User-Agent      : Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma          : no-cache
Cache-control   : no-cache
Accept          : text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding : x-gzip, x-deflate, gzip, deflate
Accept-Charset  : utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language : en
Host            : localhost:8080
Cookie          : JSESSIONID=788887A0F479749C4CEEA1E268B4A501
Content-Type    : application/x-www-form-urlencoded
Connection      : close
Content-Length  : 74

                id=1&nombre=Jam%F3n+Ib%E9rico&precio=39&cantidad=41&B1=A%F1adir+al+carrito
```

Şekil 3.7: Zararsız Bir Post İsteği

```
                POST http://localhost:8080/tienda1/publico/anadir.jsp HTTP/1.1
User-Agent      : Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma          : no-cache
Cache-control   : no-cache
Accept          : text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding : x-gzip, x-deflate, gzip, deflate
Accept-Charset  : utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language : en
Host            : localhost:8080
Cookie          : JSESSIONID=AE29AEEBDE479D5E1A18B4108C8E3CE0
Content-Type    : application/x-www-form-urlencodedConnection: close
Content-Length  : 146

                id=2&nombre=Jam%F3n+Ib%E9rico&precio=85&cantidad=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+
                FROM+datos+WHERE+nombre+LIKE+%27%25&B1=A%F1adir+al+carrito
```

Şekil 3.8: Zararsız Bir Post İsteği

Veri setinde HTTP isteklerinden oluştuğu için bir HTTP isteğinin yapısını bilmek gerekmektedir. HTTP istekleri 4 bölümden oluşmaktadır. İlk satır istek satırıdır. İstek satırı; istek metodu, istek yapılan sayfa, parametreler ve HTTP versiyonundan oluşur. İkinci bölüm istek başlığıdır. Burada çeşitli bilgiler bulunmaktadır. Üçüncü bölüm boş bir satırdan oluşmaktadır. Son bölüm ise istek verileri yer almaktadır. Bu bölümlerin tamamının olduğu örnek bir istek Şekil 3.7'de yer alan "Post" isteğinde görülebilir.

Veri setine ait istatistiksel bilgiler Tablo 3.2'de çıkarılmıştır. Tablo oluşturulurken zararsız isteklerin olduğu 2 normal dosya birleştirilmiş zararsız olarak tek dosya haline getirilmiştir. Anomali isteklerin yer aldığı dosya ise zararlı olarak isimlendirilmiştir. Ayrıca "Accept" değerlerinde *strip fonksiyonu* kullanılmıştır. "Content_Length" değerinde uzunluk değeri asgari olarak 1000 alınmıştır.

Tablo 3.2: Veri Setinin İstatistiksel Değerlendirmesi

MADDE	ZARARSIZ VERİ SETİ	ZARARLI VERİ SETİ	TOPLAM
Toplam veri miktarı	72.000 (%74,18)	25.065 (%25,82)	97.065 (%100)
Method	56.000 GET 16.000 POST 0 PUT	15.088 GET 9.580 POST 397 PUT	71.088 GET (%73,24) 25.580 POST (%26,35) 397 PUT (% 0,41)
User_Agent	72.000 Mozilla/5.0…	25.065 Mozilla/5.0…	97.065 Mozilla/5.0…
Pragma	72.000 no-cache	25.065 no-cache	97.065 no-cache
Cache_control	72.000 no-cache	25.065 no-cache	97.065 no-cache
Accept (*strip)	72.000 text/xml…	25.065 text/xml…	97.065 text/xml…
Accept_Encoding	72.000 x-gzip…	25.065 x-gzip…	97.065 x-gzip…
Accept_Charset	72.000 utf-8…	25.065 utf-8…	97.065 utf-8…
Accept_Language	72.000 en	25.065 en	97.065 en
Host	72.000 localhost:8080	24.668 localhost:8080 397 localhost:9090	96.668 localhost:8080 397 localhost:9090
Cookie	72.000 (benzersiz)	25.065 (benzersiz)	97.065 (benzersiz)
Content-type	56.000 boş 16.000 application…	15.088 boş 9.977 application…	71.088 boş 25.977 application…
Connection	72.000 close	25.065 close	97.065 close
Content_Length (*limit: 1000)	56.000 boş 2.000 (4) 2.000 (17) 2.000 (33) 14.000 diğer	15.088 boş 9.977 diğer	71.088 boş 2.057 (4) 2.046 (17) 2.054 (33) 19.820 diğer

3.3. VERİ ÖN İŞLEME

Veri ön işleme bölümünde veri setinin ham halinden alınıp kapsül ağda eğitilebilecek duruma getirme işlemleri yer almaktadır. Yapay sinir ağlarının eğitim maliyeti yüksektir. Bu yüzden gereksiz verilerin silinmesi eğitim süresini düşürmektedir. Bölüm 3.2'de yer alan bilgiler ışığında veri setinde incelemeler yapılmıştır. Zararlı ve zararsız olarak gruplanan isteklerin her ikisinde de aynı değere sahip özellikler bulunmaktadır. Örneğin "Accept_Language" özelliği hem zararlı hem de zararsız isteklerde aynı değere sahiptir. Bu değerlerin tespitinde Tablo 3.2'den faydalanılmıştır. Bunun gibi aynı değere sahip özellikler

veri setinden çıkarılmıştır. Daha sonra veri url isteği olduğu için içinde url kodlama işlemi vardır. Saldırganlar ayrıca url kodlama işleminden faydalanarak zararlı olan kodları url kodlama ile gizlemeye çalışmaktadırlar. Örneğin %25 değeri url kodlamada % karakterine karşılık gelir. SQLi açığı olan girişlerde ' karakteri SQL deyiminde alana ait değeri sarmalamak için kullanılır. Örnek olarak *kullanici-adi='yonetici'*. Bu aşamada URL üzerinden veri gelirken ' karakteri gelmesi beklenmez. Eğer bu karakter gelirse SQL sorgusunda ilgili alana veri girişinin bittiği söylenmiş olur. Açıktan faydalanmak için ' karakterinden sonra elde edilmek istenen veriye göre saldırganın gönderdiği SQL kodları gelir. Bunu engellemek için ' karakterine karşılık gelen %27 kodlaması kontrol edilebilmektedir. Bunu kontrolü atlatmak için saldırganlar %2527 şeklinde bir veri gönderir. Bu tarz durumların veri temizleme aşamasında silinmesi veri kaybı olarak nitelendirilmiş özellikle bu tarz önemli zararlı olduğu düşünülen kodlamalar temizlenmeden ve kod çözme yapmadan önce özel olarak işaretlenmiş ardından gerekli işlemler yapılmıştır.

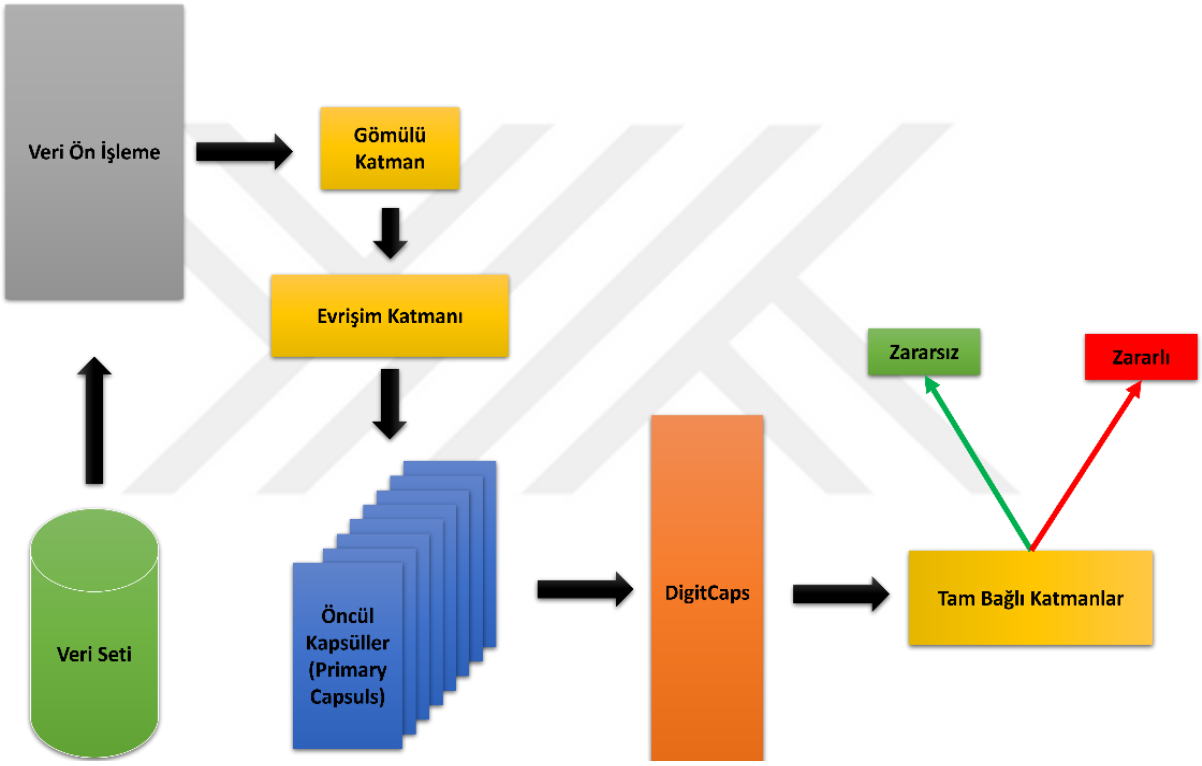
Duygu analizinde ya da metin üzerinde yapılan farklı değerlendirmelerde noktalama işaretleri genellikle önemsizdir. Bu yüzden genellikle bu karakterler silinir. Fakat bu karakterler bilgisayar sistemleri için önemlidir. Bu yüzden karakterler veri setinden atılmadan önce veri tabanı için anlamlı olan noktalama işaretleri ayrıca işaretlenmiştir. Daha sonraki aşamada ise url kod çözme işlemiyle veri seti sade hale getirilir. Devamında ise gereksiz karakterler atılmıştır. Veri seti artık arındırılmış durumda olduğundan bir sonraki adımda kayıtlar vektör haline getirilmiştir. Böylece derin öğrenme modelimiz kayıtlar üzerinden öğrenebilecek duruma gelmiştir.

3.4. MODEL EĞİTİMİ

Veri setindeki kayıtlar sayısal olarak hazırlandıktan sonra eğitim seti, doğrulama seti ve test seti olarak ayrılmaktadır. Genellikle eğitim seti %80 test seti %20 olarak seçilmektedir. Veri setindeki kayıtlar çok olduğu durumda eğitim seti değerinin yüksek seçilmesi problem oluşturmazken kayıt sayısının az olduğu veri setlerinde eğitim seti oranın yüksek seçilmesi doğru bir uygulama olmayacaktır [59]. Örneğin 500 kayıt içeren bir eğitim setinde eğitim setinin %80, doğrulama setinin %10 ve test setinin %10 seçilmesi durumunda 400 kayıt eğitim için kullanılırken 50 kayıt test amaçlı kullanılacaktır. Bu da test verilerinin gerçek durumu ne kadar az temsil ettiğini göstermektedir. Çok kullanılan değerlerden biri de %60, %20 ve %20 şeklindedir. Amazon ML ise varsayılan olarak eğitim %70 test %30 olacak şekilde böler [60].

Burada eğitim, doğrulama ve test seti olarak %70, %15, %15 oranında ayırma işlemi yapılmıştır.

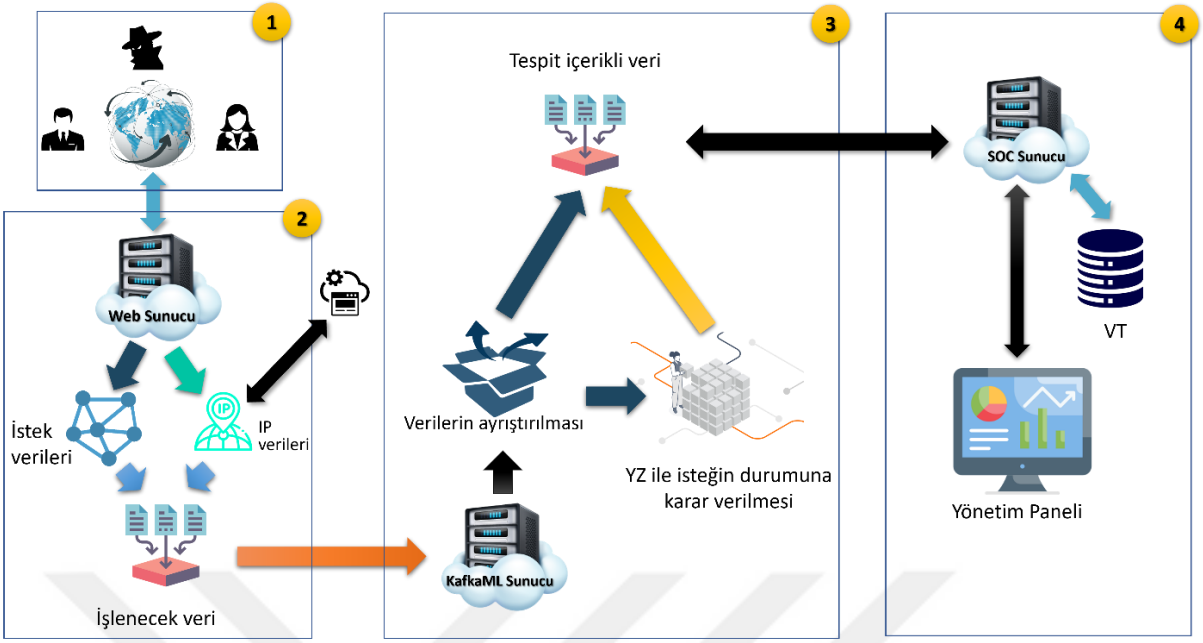
Kapsül ağımız bir evrişim katmanı, 8 öncül kapsülden ve DigitCaps katmanından oluşmaktadır. Verilerin doğrultma işlemi için tam bağlantılı katman bulunmaktadır. Evrişim katmanımızda çekirdek 6, kaydırma 1 adım olarak atanmıştır. Dolgu işlemi için 1 seçilmiş ve sıfırla doldurulmuştur. Öncül katmanımızda çekirdek değeri 9 seçilmiş ve kaydırma olarak 2 belirlenmiştir. Eğitim test sürecine ait yapı Şekil 3.9'da görülmektedir.



Şekil 3.9: Eğitim Test Süreci Adımları

3.5. UYGULAMANIN GERÇEKLEŞTİRİLMESİ

Eğitim aşamasında elde edilen en iyi model serileştirme yöntemi ile kaydedilmektedir. Daha sonra bu model kullanılarak web sitesine yapılan anlık isteklerin zararlı olup olmadığı tespit edilecektir. Senaryoda yukarıdaki bölümlerde değinilmeyen bulut hizmetler ve Apache Kafka kullanılacaktır. Uygulanan sistemin akış şeması Şekil 3.10'da görülmektedir.



Şekil 3.10: Uygulanan Sistemin Akış Şeması

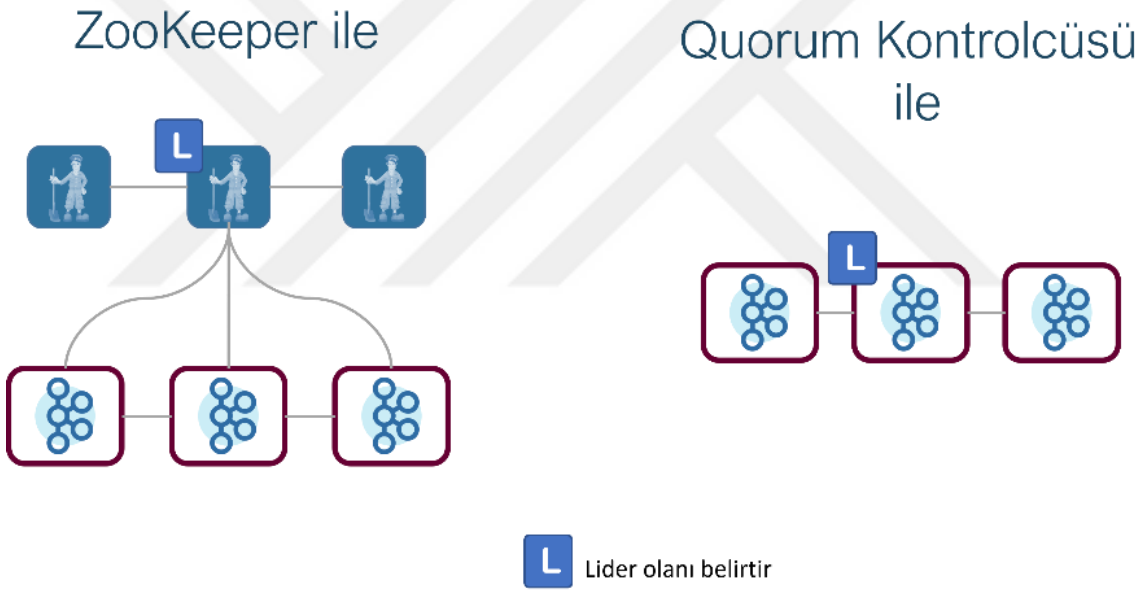
3.5.1. Bulut Hizmetler

Bulut hizmeti, fiziksel olarak bilişim alt yapısı satın almadan kiralama yöntemiyle sunucu kurma, depolama alanı ayrıma, web sistemlerini sunma, yapay zekâ çalışmaları yapma vb. işlerin ve hizmetlerin gerçekleştirilmesini sağlar. Böylece şirketler fiziksel olarak kurması gereken donanım ve alt yapı maliyetinin yanında bakımının ve güvenliğinin sağlanması gibi maliyetli işlemlerden de kurtulmuş olmaktadır. Günümüzde birçok bulut hizmeti veren firma bulunmaktadır. Microsoft Azure platformu da bunlardan sadece biridir. Öğrenciler için ücretsiz öğrenci hesabı kullanım imkanı sebebiyle Microsoft Azure tercih edilmiştir.

3.5.2. Apache Kafka ve KRaft

Apache Kafka, olay akış sistemi olarak geçen dağıtık olarak çalışan bir mesajlaşma platformudur. Apache sitesinde yer alan bilgiye göre Fortune dergisinin ilk 100 şirketinin %80'i Kafka kullanmaktadır. Hızlı, ölçeklenebilir ve güvenli bir platform olan Kafka birçok sektörde halihazırda kullanılmaktadır. Kafka'nın kullanıldığı sektörlere borsa, banka, lojistik, IoT sistemleri, hastaneler vb. örnek olarak verilebilir. Kafka'nın broker, consumer, topic, partition gibi nesneleri bulunmaktadır. Kısaca broker, mesajı oluşturur ve gönderir. Consumer, mesajları okur. Topic, yani konu mesajların yazıldığı diğer bir değişle verilerin saklandığı yerdir. Partition yani bölümler, konuların içindeki mesajların bölümleridir [61]. Zookeeper, Kafka'nın çalışması için zorunlu bir hizmettir. Zookeeper yapılandırma bilgilerini korumak, dağıtık senkronizasyonu sağlamak, grup hizmeti sunmak gibi görevleri olan merkezi bir hizmettir [62].

KRaft, Zookeeper olmadan çalışan Apache Kafka'nın yeni bir versiyonudur. Zookeeper'ın olmadığı versiyonu olan Apache Kafka, meta verilerin yönetimi için Zookeeper'ın kullanılması yerine bu işlemin Kafka'nın içine dahil edilmesi fikrine dayanmaktadır. Böylece iki farklı sistem kullanmak yerine sadece KRaft (Kafka Raft) kullanarak işlemler basit bir şekilde gerçekleştirilebilmektedir. Bunu Raft Uzlaşı Protokolünün (Raft Consensus Protocol) olay temelli bir versiyonunu kullanarak başarmışlardır. KRaft henüz geliştirme aşamasındadır. Bu sebeple KRaft modu sadece geliştirme ve test ortamları için kullanılması gerektiği belirtilir. KRaft'ın Zookeeper bağımlı Kafka'dan daha hızlı olduğu grafiklerle gösterilmiştir. Zookeeper bağımlı Kafka ile KRaft platformlarının yapısı Şekil 3.11'de görülmektedir. KRaft, Apache Kafka'nın 2.8 sürümünden itibaren erken erişim olarak sunulmaktadır [63]. KRaft kullanabilmek için kraft modunun aktifleştirilmesi yeterlidir.

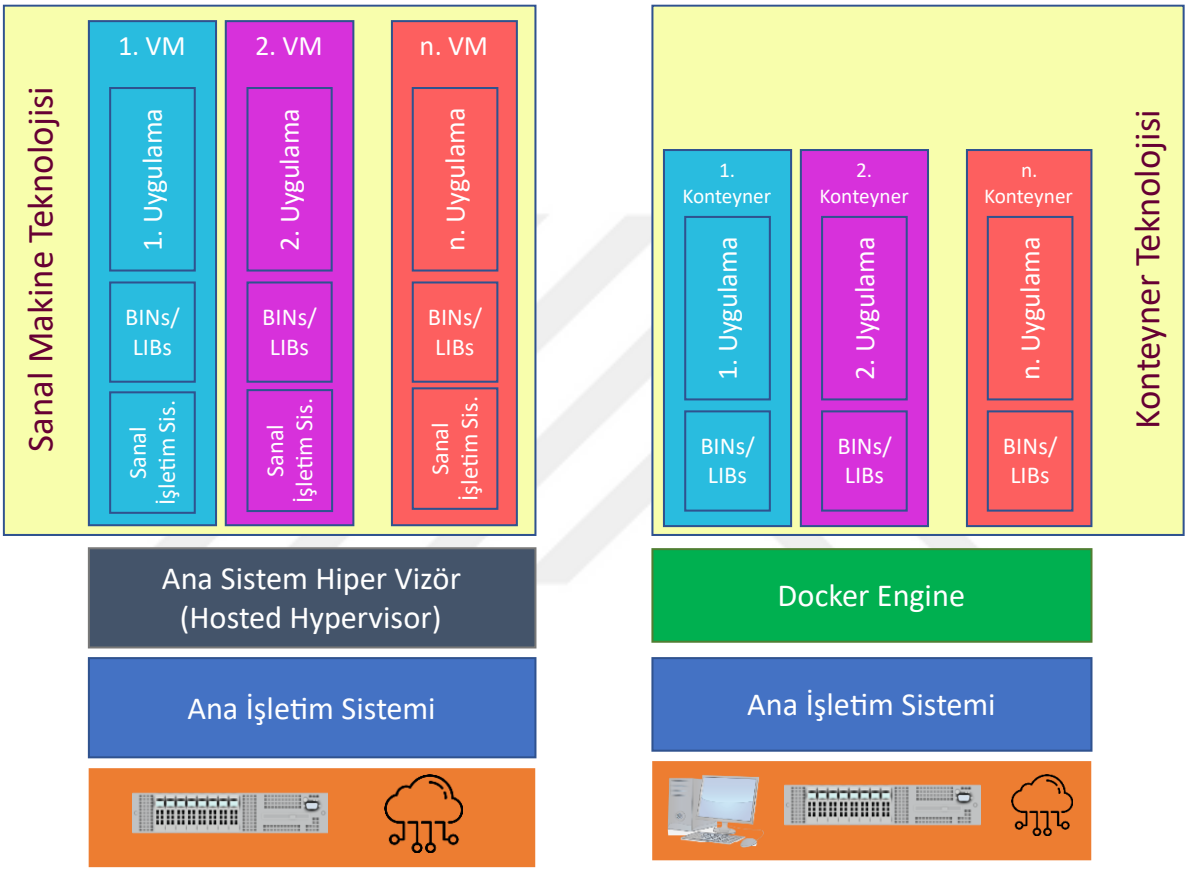


Şekil 3.11: Zookeeper ve Kafka ile KRaft Yapısı [61]

3.5.3. Docker

Uygulama geliştirilirken uygulamamızın birçok kütüphaneye bağımlılığı olabilir. Fiziksel makinede geliştirme yaptığımız zaman farklı kütüphanelere ya da aynı kütüphanenin farklı versiyonlarına ihtiyaç olduğunda uygulamalar problem çıkarabilir. Bu uygulamaların birbirinden bağımsız ortamda çalışması için konteyner yapısı kullanılmaktadır. Docker bu alandaki açık kaynak ücretsiz bir sanallaştırma teknolojisidir. Docker, sanal makine çözümüne benzemektedir. Sanal makine sisteminde bir işletim sistemi olur. Ana işletim sisteminin bakım ve sıkılaştırma işlemlerinin yanına bir de sanal makinedeki işletim sistemlerinin bakım ve sıkılaştırılması işlemleri gelecektir. Bu maliyetli bir çözüm olmaktadır. Ayrıca her sanal işletim

sistemi aynı zamanda depolama alanının gereksiz tüketilmesine sebep olacaktır. Docker mimarisi gereği ana işletim sistemin kaynaklarını kullanmaktadır. Böylece hafif, taşınabilir olmasının yanında az yer kaplar. Aynı makinede istenildiği kadar konteyner oluşturabilir. Bu konteynerler birbirinden bağımsız bir şekilde çalışırken Docker ortamının sunduğu hizmetlerden faydalanarak kendi aralarında ve dış dünya ile rahatlıkla haberleşebilmektedir. Konteyner ile sanal makine teknolojisi arasındaki yapı farkı Şekil 3.12'de görülmektedir.

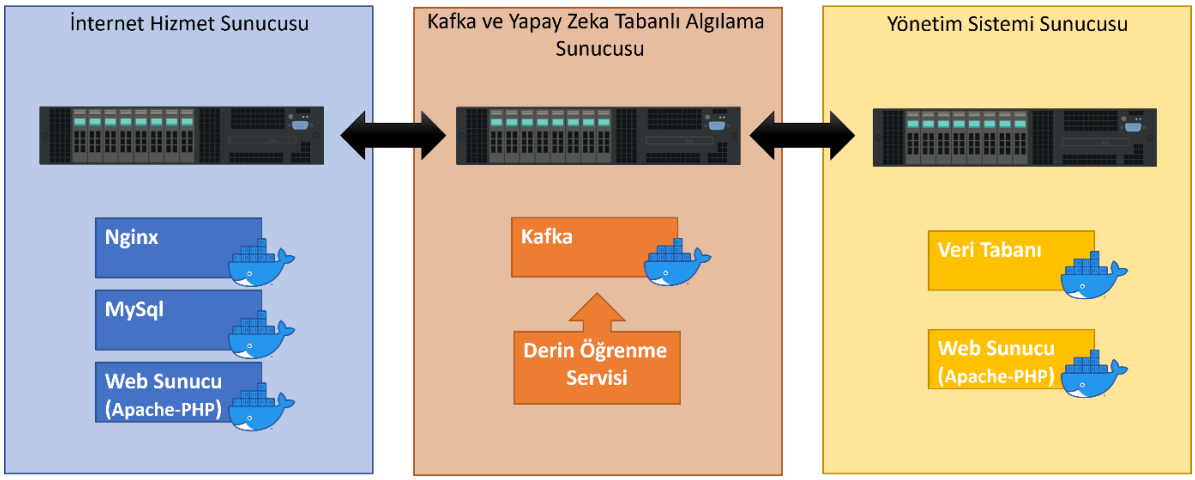


Şekil 3.12: Sanal Makine ve Konteyner Yapısı[2]

3.5.4. Ortamın Hazırlanması

Microsoft Azure üzerinde 3 adet bulut sunucu oluşturulmuştur. Değişik amaçlar için kullanılacak bu 3 sunucu benzer özelliklere sahiptir. Birinci sunucu ile ikinci sunucu Kafka mesajlaşma sistemi ile haberleşmektedir. İkinci ve üçüncü sunucu, üçüncü sunucuda çalışmakta olan RESTful servisi ile veri aktarımı yapmaktadır. Uygulama ortamı Şekil 3.13'te görülmektedir.

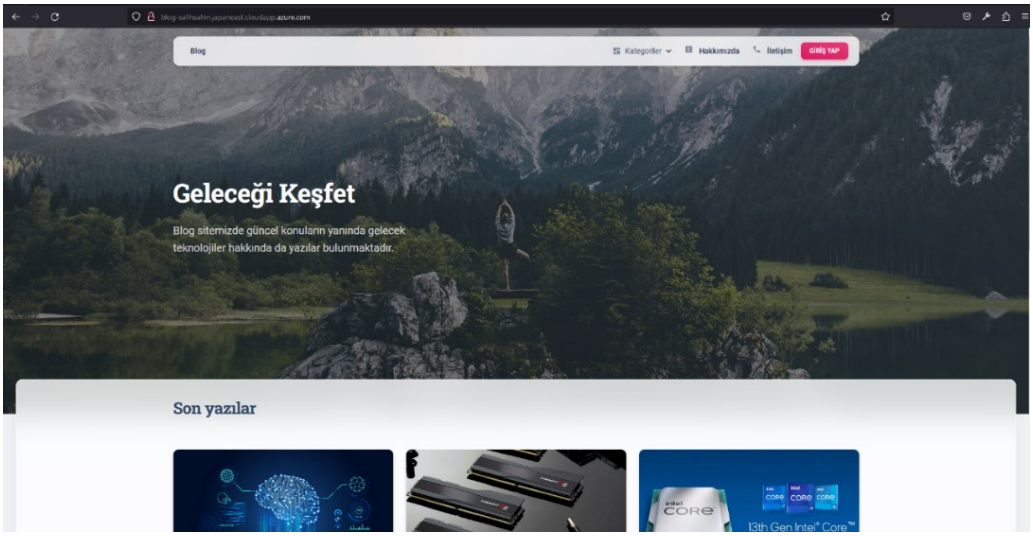
[2] *Binary files* kavramı için *Bins* kısaltması kullanılmış olup ikili dosyalar anlamına, *Library files* için de *Libs* kısaltması kullanılmış olup uygulamanın ihtiyaç duyduğu kütüphane dosyaları anlamına gelmektedir.



Şekil 3.13: Uygulama Ortamı

3.5.4.1. Web Hizmeti

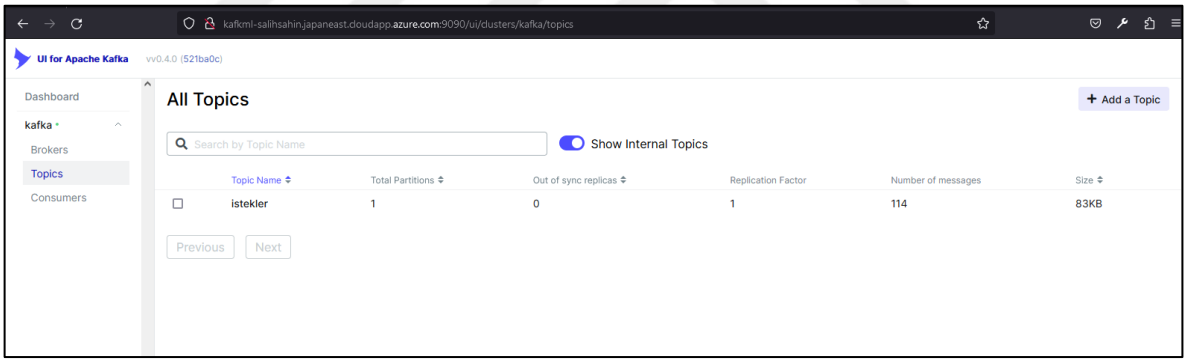
Kullanıcıların hizmetine sunulacak olan web sitesi konteyner yapısıyla bir sunucuda çalışacaktır. Bu sunucu üzerinde Docker kuruludur. Docker stack kullanılarak ters vekil sunucusu (reverse proxy) olan Nginx, web sunucusu ve veri tabanı sunucusu hizmeti verilmektedir. Şekil 3.14'te Azure üzerinde çalışan web sunucumuzun sunduğu web sayfası görülmektedir. Burada veri tabanı için farklı bir fiziksel sunucu seçilebilirdi. Fakat asıl konunun dışında olduğu için bu durum göz ardı edilmiştir. Bu web sitesine yapılan istekle birlikte bazı veriler arka planda toplanmakta ve JSON verisine dönüştürülmektedir. Kullanıcının kullandığı tarayıcı bilgisi, IP adresi, IP adresi konum bilgileri, istekler URL ve gönderilen parametreler vb. bilgiler toplanmaktadır. Konum bilgisi farklı bir API sağlayıcıdan alınmaktadır. Oluşturulan JSON verisi daha sonra Kafka ML olarak isimlendirilen başka bir bulut sunucudaki Apache Kafka'ya gönderilmektedir.



Şekil 3.14: Web Uygulamasının Çalışır Durumdaki Ekran Görüntüsü

3.5.4.2. Kafka ve Yapay Zekâ Karar Destek Sistemi

KafkaML sunucusundaki Apache Kafka'ya gelen mesajlar okunacak ve aynı sunucuda çalışan tespit sistemimize iletilecektir. Apache Kafka olarak KRaft kullanılacaktır. KRaft, Bölüm 3.5.2'de de belirtildiği gibi hâlen erken erişim durumundadır. Fakat kurulan yapı ticari odaklı olmadığı için kullanmakta herhangi bir sakınca yoktur. KafkaML sunucumuzda Python ile yazılmış bir servis programımız çalışmaktadır. Programımızda bir Kafka tüketici (consumer) modülü, yapay zekâ karar destek modülü ve diğer sunucuya veri aktarımı yapacak veri tabanı kayıt modülü bulunmaktadır. Bu sunucumuzda çalışan tüketici gelen mesajları okuyacaktır. Daha sonra mesajdaki verilerden kullanılacak olan veriler ayrıştırılacak ve eğitilmiş olan modelle test edilerek isteğin zararlı olup olmadığına karar verilecektir. Bir sonraki aşamada önceden okunan mesaj, isteğin zararlı olup olmadığını belirten karar ve yönetim merkezinde çalışan RESTful servisinden elde edilen token tekrar JSON formatına çevrilecektir. Yeni oluşturulmuş olan JSON verisi veri tabanına kaydedilmek üzere üçüncü sunucuya yani SOC (Security Operations Center) ismi verilen sunucuya gönderilmektedir. KRaft'ın çalışmasına ait görüntü Şekil 3.15'te görülebilmektedir.

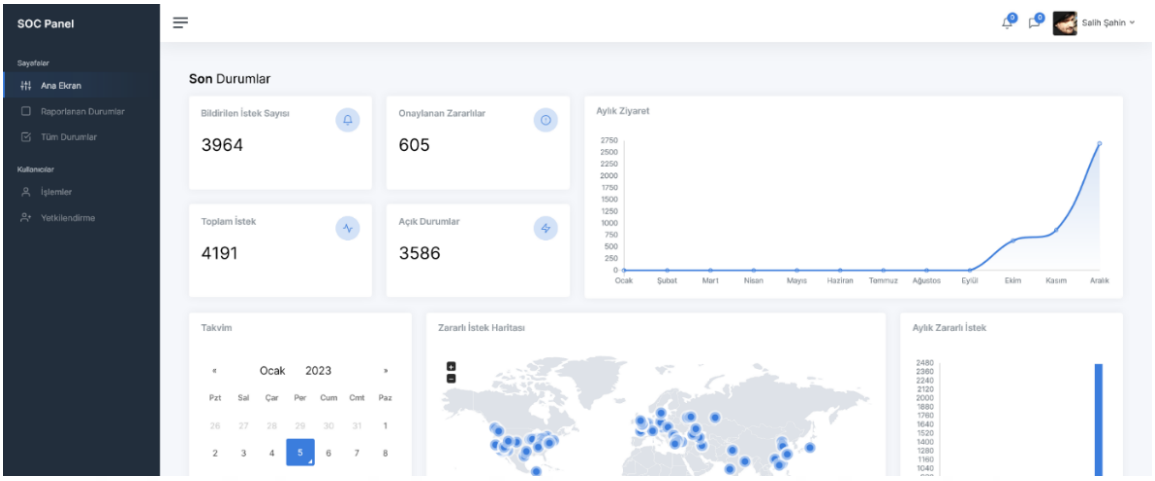


Şekil 3.15: KRaft Kullanıcı Ara Birimi

3.5.4.3. Güvenlik Yönetim Merkezi Paneli

KafkaML sunucusundan gelen veriler bu sunucuda çalışan veri tabanına kaydedilmektedir. Böylelikle hem gelen ham veri burada saklanacak hem de makine öğrenmesinden gelen tespit SOC ekibine yardımcı olacaktır. Gelen veriler API servisi tarafından karşılanmaktadır. Gelen veri çözümlenir ve gerekli kimlik doğrulama işlemleri yapılır. Ardından veriler doğrulanır. Eğer gelen verilerde herhangi bir sorun tespit edilmezse veri tabanının ilgili bölümüne kaydedilir. Bu sunucu dıştan erişime kapalı durumdadır. Sunucu sadece iç ağdan ve belirlenen IP adreslerinden gelen isteklere cevap verecek şekilde tasarlanmıştır. Yetkili kullanıcılar giriş bilgilerini panele girdikten sonra oturum açarlar ve gerekli ön bilgileri ana sayfadan görebilirler. Ayrıca raporlanan durumları teyit etmek için

raporlanan durumlar sayfasını kullanabilirler. Bu özellikle etiketleme için kullanılabilecek sayfadır. Ayrıca burada listelenen istekler Excel veya CSV dosyası olarak indirilebilmektedir. Böylece yeni oluşan durumlar kullanılarak model eğitiminin tekrardan yapılabilir ve böylece model geliştirilebilmektedir. Ana sayfada ayrıca sonra 1 dakika içinde belirlenen sayıda ihlal durumu oluşursa bu durum bildirim bölümünde listelenmektedir. Ayrıca belirli durumlarda da sistem otomatik mesaj oluşturacaktır. Böylece yapılan güvenlik ihlalleri kayıtlı ve görülebilir durumda olacaktır. Güvenlik olay merkezinin yönetim için kullanacağı panel Şekil 3.16'da görülmektedir.



Şekil 3.16: Güvenlik Olay Merkezi (SOC) Paneli

4. BULGULAR

Kapsül ağlar konusu ile ilgili yapılan çalışmada elde edilen sonuçlar bu bölümde verilecektir. Düzenli olmayan veri setleri ile yapılan çalışmalarda veri ön işleme ve model eğitimi aşaması olmaktadır. Günlük yaşamda da genellikle gelen veriler istenen özelliklerde değildir. Eğitim işleminde Intel i5 11800, 16 GB DDR4 Ram, NVIDIA GeForce RTX 3050 4GB GPU ve 1TB NVMe M.2 SSD özelliklerine sahip dizüstü bilgisayar kullanılmıştır. Genel performans metriklerinden olan *doğruluk*, *F1 puanı*, *kesinlik* ve *duyarlılık* metrikleri incelenecektir. Ayrıca doğru negatif(TN), doğru pozitif(TP), yanlış negatif(FN) ve yanlış pozitif(FP) değerleri hesaplanacaktır. Çalışan sistem üzerinde yapılan denemelerde yine bu metriklerin yanında ortalama tespit ve kayıt süresi ölçülerek sistemin doğru çalışmasının yanında hızlı yanıt süresi de belirlenmiş olacaktır. Modelin ilk kısmında gömülü (embedding) katman gelmektedir. Giriş olarak sözlük uzunluğu ve çıkış olacak olan bir boyut verisi verilir. Sözlük boyutumuz 798 ve çıkış boyutumuz ise 100 olarak seçilmiştir. Ardından sıra evrişim katmanına gelir. Giriş boyutu 100, çıkış olarak 256 seçilmiştir. Çekirdek boyutu (Kernel size) 6, kaydırma (stride) olarak 1 seçilmiş ve doldurma işlemi yapılacaktır. Evrişim katmanından sonra birincil ya da diğer bir değişle öncül katman (primary caps) gelmektedir. 8 adet kapsül bulunmaktadır. Giriş boyutu 256 ve çıkış boyutu 32'dir. Çekirdek boyutu 9 ve kaydırma 2 olarak seçilmiştir. Kapsül ağının son katmanı olan DigitCaps katmanı 16x10 boyutundadır. Eğitim için ayrı ayrı dizüstü bilgisayarda GPU desteği olmadan ve GPU desteğiyle, Google Colab Ücretsiz ve Google Colab Pro versiyonunda denenecektir. Google Colab Ücretsiz versiyonunda 12.68 GB Tesla T4 16 GB GDDR6 ekran kartı kullanılmaktadır. Google Colab Pro versiyonunda 83 GB Sistem belleği ve Nvidia A100 SXM4 40 GB HBM2e ekran kartı bulunmaktadır. Diğer sistemlerde GPU belleği yeterli olmadığından batch değeri hata almamak için 10 seçilmiştir. Google Colab Pro'da batch değeri 64 seçilmiştir. Google Colab Pro'da işlem birimi kullanılarak hizmet kullanılabilmektedir.

4.1. VERİ ÖN İŞLEME

Eğitimde kullanılan veri seti ile ilgili detaylı bilgiler Bölüm 3.2'de detaylı bir şekilde anlatılmıştır. Veri setinde kayıt silme işlemi yapılmamıştır. Okunan her satırda veriler kontrol

edilmiş ve gereksiz öznitelikler dahil edilmemiştir. Toplam 97.065 adet kayıt bulunmaktadır. Veri setinden veri okuma, veri temizleme ve kelime sözlüğü (Vocabulary) oluşturma işlemi ortalama 1 dakikada tamamlanmıştır. Kelime sözlüğüne ait kelime bulutu Şekil 4.1'de görülmektedir. Sözlüğümüzde 798 adet kelime bulunmaktadır.



Şekil 4.1: Veri setine ait kelime bulutu

4.2. EĞİTİM BULGULARI

Eğitim için derin öğrenme yöntemlerinden biri olan kapsül ağlar kullanılmış ve eğitim sonucunda ikili sonuç üretilmek hedeflenmiştir. Girdinin işlenip zararlı (1) ya da zararsız (0) olup olmadığı çıktı olarak verilecektir.

4.2.1. Klasik Makine Öğrenmesi Algoritmaları Bulguları

Bölüm 2.4'te verilen makine öğrenmesi algoritmalarından olan SVM, DT, RF ve LR ile ayrı ayrı modeller eğitilmiştir. Eğitim işlemlerinde diz üstü bilgisayar kullanılmıştır. Makine öğrenmesi kullanarak model oluşturmak için değişken seçimi ve veri ön hazırlama işlemleri yapılmak zorunda olduğundan öncelikle bu işlemler yapılmıştır. Daha önceden de belirtildiği gibi veri setinden aynı değerlere sahip öznitelikler bulunmaktadır. Bu öznitelikler veri setinden çıkarılmıştır. Daha sonra veri temizleme işlemi yapılarak eğitime hazır olan Pandas veri çerçeve (DataFrame) yapısı oluşturulmuştur. Son aşama olarak oluşturulan veriler makine öğrenmesi algoritmalarında kullanılarak modellerin eğitilmesi tamamlanmıştır.

Eğitim aşamasında model 10 katlı çapraz doğrulama (k-fold Cross-Validation) yöntemi ile elde edilen en iyi parametreler seçilerek oluşturulmuştur. Modellere ilişkin bulgular tablo halinde Tablo 4.1'de verilmektedir.

Tablo 4.1: Makine Öğrenmesi Yöntemlerine Ait Sonuçlar

YÖNTEM	DOĞRULUK	F1 SKORU
Destek Vektör Makinesi (SVM)	0,9532	0,9004
Karar Ağaçları (DT)	0,9644	0,9260
Rastgele Orman (RF)	0,9666	0,9305
Lojistik Regresyon (LR)	0,9567	0,9016

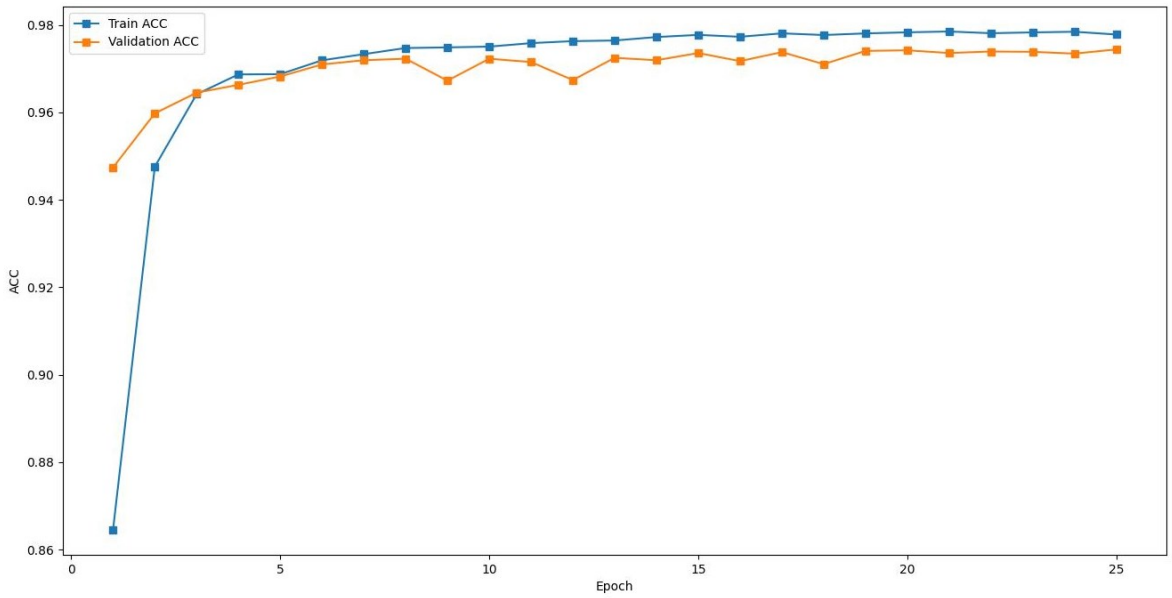
4.2.2. Kapsül Ağlar Bulguları

Dizüstü bilgisayarda GPU desteği olmadan eğitim işlemi her bir eğitim dönemi (epoch) yaklaşık olarak 2 saatten fazla sürmektedir. 25 epoch için yaklaşık 50 saatten fazla zaman gerekecektir. Bu sebeple GPU desteği olmadan çok uzun süreceği ve hatalar alınması muhtemel olacağından dolayı bu eğitim yöntemi göz ardı edilmiştir.25 eğitim dönemi işlemi dizüstü bilgisayarda GPU desteği kullanılarak yaklaşık olarak 5 saatte tamamlanmaktadır. Google Colab ücretsiz olan platformda eğitim yaklaşık olarak 2,5 saatte ve Google Colab Pro versiyonunda ise eğitim yaklaşık olarak 30 dakikada tamamlanmaktadır. Eğitim esnasında %97,43 doğruluk elde edilmiştir. Ölçüm sonuçları Tablo 4.2'de verilmiştir.

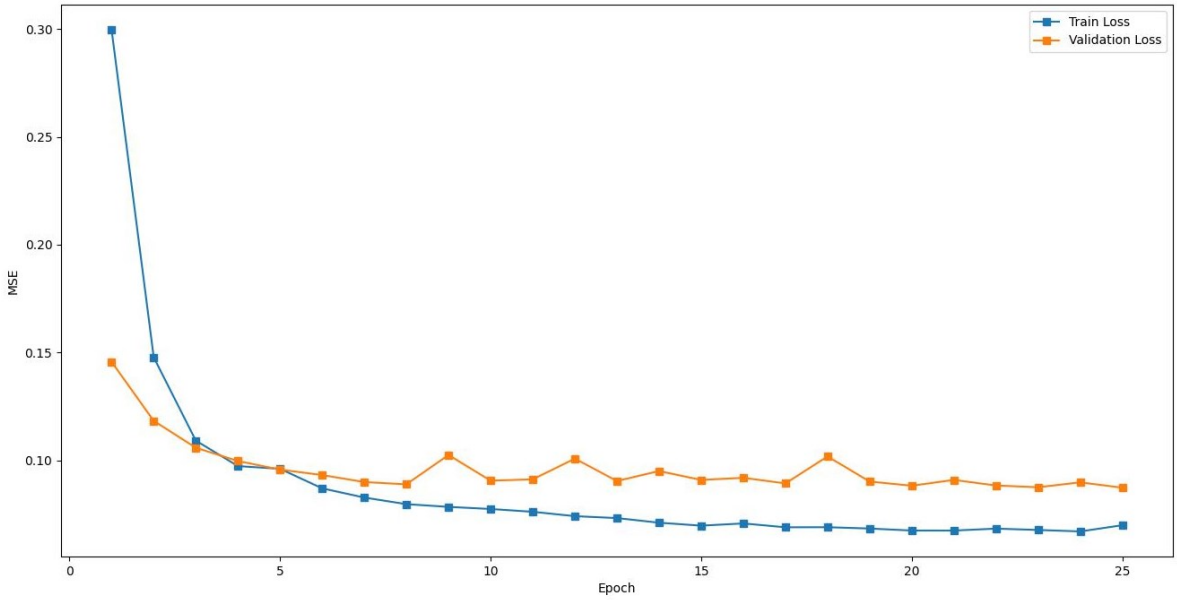
Tablo 4.2: Eğitim Ortamları ve Ölçüm Sonuçları

ORTAM	SÜRE (YAKLAŞIK)	DOĞRULUK
Diz üstü bilgisayar (GPU desteği olmadan)	50 saat (tahmini)	–
Diz üstü bilgisayar (GPU desteği olmadan)	5 saat 10 dakika	%97,21
Google Colab Ücretsiz	2 Saat 30 dakika	%97,35
Google Colab Pro	30 dakika	%97,43

Eğitime dair doğruluk ile kayıp grafiği Şekil 4.2'de ve Şekil 4.3'te verilmiştir.

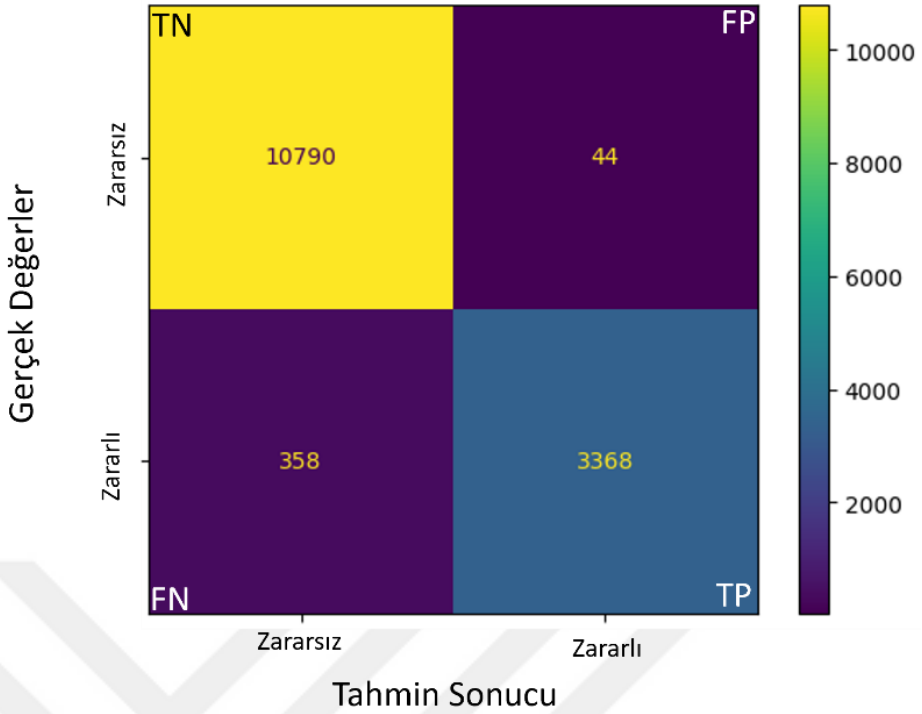


Şekil 4.2: Doğruluk Grafiği



Şekil 4.3: Kayıp Grafiği

Eğitim sonunda elde edilen modelle yapılan test işlemi sonunda ortaya çıkan karmaşıklık matrisi Şekil 4.4'te görülmektedir.



Şekil 4.4: Test Karmaşıklık Matrisi

Karmaşıklık matrisi incelendiğinde TP yani doğru pozitif değeri 3.726 zararlı isteğin 3.368 tanesi zararlı olarak tespit edildiğini göstermektedir. Doğru negatif yani TN değeri 10.834 adet zararsız istekten 10.790 tanesi doğru bir şekilde zararsız olarak tespit edildiğini belirtmektedir. Yanlış negatif değeri olan FN, 3.726 zararlı istekten 358 tanesi zararsız olarak yanlış bir şekilde tahmin edildiğini göstermektedir ve son olarak yanlış pozitif değeri FP, 44 olarak ölçülmüştür. Başka bir değişle 44 tane zararsız istek zararlı olarak tahmin edilmiştir.

Doğruluk, kesinlik, duyarlılık ve F1 skorlarına ait ölçümler “Sklearn Metrics” ile hesaplanmış ve Tablo 4.3’te verilmiştir.

Tablo 4.3: Sklearn Ölçüm Metrikleri

Özellikler	Keskinlik (Precision)	Duyarlılık (Recall)	F1 Skoru	Kayıt Sayısı
0	0,97	1,00	0,98	10.834
1	0,99	0,90	0,94	3.726
Doğruluk	–	–	0,97	14.560
Makro Ortalama	0,98	0,95	0,96	14.560
Ağırlıklı Ortalama	0,97	0,97	0,97	14.560

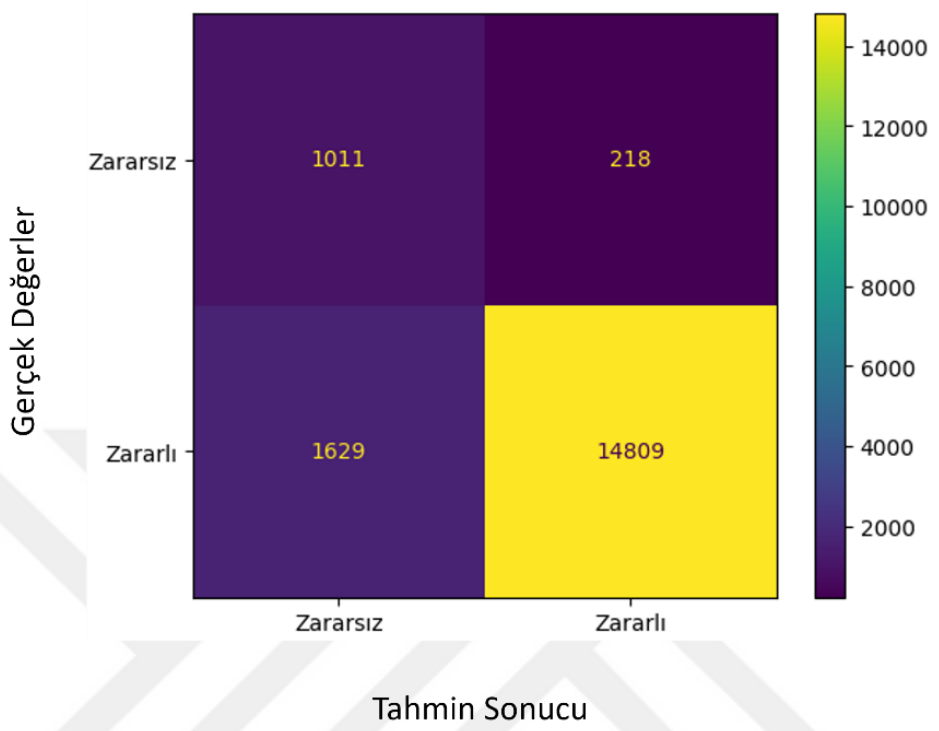
Doğru pozitiflerin, doğru pozitif ve yanlış negatiflere oranına duyarlılık denilmektedir. Duyarlılık sonucu ortalama 0,95 olarak bulunmuştur. Kesinlik, doğru pozitiflerin doğru pozitif ve yanlış pozitiflere oranı olarak hesaplanır ve ortalama 0,98 olarak hesaplanmıştır. F1 skoru ise ortalama 0,96 olarak belirlenmiştir. Genel ağırlıklı ortalama bakıldığında ise 0,97 olarak başarı elde edildiği görülmektedir. Bu ölçümlerden elde edilen sonuca bakıldığında modelin zararlı olan istekleri ise %90,39 oranında ve zararlı olmayan istekleri %99,59 gibi çok iyi bir skorla tespit edebildiği görülmektedir.

4.3. UYGULAMA SONUÇLARI

İnternet sitesine yapılan istekler Kafka sunucusuna en fazla 1 sn içinde gönderilmektedir. Kafka sunucusuna gelen istek işlenmekte ve işlenen isteğin daha sonra derin öğrenme yöntemi ile zararlı olup olmadığına bakılmaktadır. Elde edilen sonuç ile gelen istek, yönetim sunucusundaki veri tabanına kaydediliyor. Bu işlem için yaklaşık 1 saniye harcanmaktadır. İlgili kullanıcılar, yönetim panelinden gelen isteklere ve isteklerin zararlı olup olmadığına bakabiliyorlar. Raporlanan isteklerin doğru raporlanıp raporlanmadığı etiketlenebilmektedir. Denemeler sonucunda isteklerin başarılı şekilde tespit edildiği görülmektedir. Ekranlara ait görüntüler Ekler bölümünde verilmiştir. Henüz yeni kurulan bir sistem olduğu için az bir istek verisi bulunmaktadır ve bu az veri ile eğitilen model Kafka ML sunucusunda kullanılmaktadır. Yeni model oluşturulurken başlangıçta uygulamanın çalışmasından 1.277 adet zararlı istek ve 55 adet zararsız istek alınarak veri seti oluşturulmuştur. Bu veri seti ile model eğitilmiş ve bu modelle tespit işlemleri gerçekleştirilmiştir.

Uygulamanın başarı oranını görebilmek için internet sitesine zararlı ve normal istekler gönderilmesi gerekmektedir. Bunun için normal istekler için site belli aralıklarla ziyaret edilmiştir. Ayrıca zararlı isteklerin yapılacağı uygulamalarda keşif amaçlı zararsız istekler göndermektedir. Zararlı istekler için sızma testlerinde tercih edilen Linux dağıtımlarından biri olan Kali Linux dağıtımı kullanılmıştır. Kullanılan araçlar sqlmap, skipfish ve nikto kullanılmıştır. Ayrıca Burpsuite Enterprise programından da faydalanılmıştır. Tarama ve açık bulma araçları ile istenilen istekler oluşturulmuş olmaktadır. Toplamda 17.667 istek yapılmıştır. İstekler KafkaML sucusuna geldiğinde zararlı olup olmadığı tahmin edilip sonuçlar etiketlenerek yönetim sunucusuna gönderilmiştir. Yönetim sunucusuna gelen istekler indirilmiş

ve tek tek etiketlenerek gerçek etiket değerlerine de ulaşılmıştır. Tüm bu işlemlerin sonucunda elde edilen karmaşıklık matrisi Şekil 4.5'te görülmektedir.



Şekil 4.5: Uygulama Sonucu Oluşan Karmaşıklık Matrisi

Uygulama sonucunda elde edilen karmaşıklık matrisinden faydalanarak doğruluk duyarlılık kesinlik ve F1 skorları hesaplanarak Tablo 4.4'e aktarılmıştır.

Tablo 4.4: Uygulama Sonuç Metrikleri

KESKİNLİK (PRECİSİON)	DUYARLILIK (RECALL)	F1 SKORU	DOĞRULUK (ACCURACY)
0,9855	0,9009	0,9413	0,8955

5. TARTIŞMA

Kapsül ağlar ilk çıktığı andan günümüze kadar çok fazla ilgi görmüş, üzerine birçok çalışma yapılmış ve yapılmaya devam etmektedir. Tez kapsamında temel bir kapsül ağ hazırlanmış ve anomali tespiti yapılmaya çalışılmıştır. Model oluşturma sürecinde %97,43 gibi kabul edilebilir bir sonuç elde edilmiştir. Aynı veri seti ile yapılan çalışmalarla karşılaştırma yapıldığında aşağıdaki sonuçlar elde edilir.

Niu ve Li yapmış oldukları çalışmada KNN'de %91,88, RF'de %84,51, GB'de 83,02, DT'de %81,84, LR'de %74,07 ve NB'de %66,75 doğruluk elde etmişlerdir. Çalışmaların devamında derin öğrenme yöntemlerini kullanmışlar ve CNN'de %93,44, LSTM'de %95,70 ve GRU'da %97,17 başarı elde etmişlerdir. Tezde kullanılan yapıda elde edilen başarı az önce belirtilen yöntemlerden daha fazla başarı elde etmiştir. Niu ve Li çalışmalarının devamında hibrit derin öğrenme yöntemi kullanarak CNN+GRU'da %97,56 ve veri seti arttırma işlemleri ile %99,00 başarı elde etmişlerdir. Tezdeki modelimizde hibrit öğrenme yöntemi kullanılmamış ayrıca veri setinde herhangi bir arttırma işlemi yapılmadan orijinal haliyle kullanılmıştır.

Jemal ve diğerleri LSTM ve CNN modelini birleştirerek hibrit bir öğrenme modeli geliştirmişlerdir. M-CNN olarak adlandırılan bu modeli 3 farklı türde kurgulamışlar ve doğruluk oranlarını %96,96, %97,36 ve %99,25 olarak elde etmişlerdir. Bu çalışmada önerilen model, M-CNN modeline ait üç türün ikisinden daha fazla başarı göstermiştir.

Tez olarak gerçekleştirilen çalışmada elde edilen sonuçların özet hali Tablo 5.1'de verilmiştir. Önerilen modelin diğer çalışmaların bir kısmından daha iyi sonuç elde etmesi umut verici olarak görülmektedir. Çalışmada ayrıca çalışan bir sistem kurgulanmış ve elde edilen model bu sistemde çalışır duruma getirilmiştir. Aktif olarak bir bulut sistem üzerinde model tehditleri algılamaya çalışmış ve elde edilen sonuçların incelenebilmesi için sonuçlar farklı bir bulut sunucuda saklanmıştır.

Tablo 5.1: Model Sonuçlarının Karşılaştırma Tablosu

ÇALIŞMAYI GERÇEKLEŞTİREN	VERİ SETİ	YÖNTEM	DOĞRULUK ORANI
Niu ve Li	CSIC 2010 HTTP	NB	%66,75
Niu ve Li	CSIC 2010 HTTP	LR	%74,07
Niu ve Li	CSIC 2010 HTTP	DT	%81,84
Niu ve Li	CSIC 2010 HTTP	GB	%83,02
Niu ve Li	CSIC 2010 HTTP	RF	%84,51
Niu ve Li	CSIC 2010 HTTP	KNN	%91,88
Niu ve Li	CSIC 2010 HTTP	CNN	%93,44
Tez_ML_Model	**CSIC 2010 HTTP**	**SVM**	**%95,32**
Tez_ML_Model	**CSIC 2010 HTTP**	**LR**	**%95,67**
Niu ve Li	CSIC 2010 HTTP	LSTM	%95,70
Tez_ML_Model	**CSIC 2010 HTTP**	**DT**	**%96,44**
Tez_ML_Model	**CSIC 2010 HTTP**	**RF**	**%96,66**
Jemal ve diğerleri	CSIC 2010 HTTP	MCNN Tip1	%96,96
Niu ve Li	CSIC 2010 HTTP	GRU	%97,17
Jemal ve diğerleri	CSIC 2010 HTTP	MCNN Tip2	%97,36
Önerilen Model	**CSIC 2010 HTTP**	**Kapsül Ağ**	**%97,43**
Niu ve Li	CSIC 2010 HTTP	CNN+GRU	%97,56
Niu ve Li	CSIC 2010 HTTP	CNN+GRU (Veri arttırma)	%99,00
Jemal ve diğerleri	CSIC 2010 HTTP	MCNN Tip3	%99,25

Bulut sistem üzerinde çalışan web uygulamasında elde edilen sonuçlar oluşturulan modelle doğrudan bağlantılıdır. DL'de en önemli hususlardan biri model eğitiminde kullanılan veri setlerinin büyük boyutlu olmasıdır. Uygulama için oluşturulan model mevcut olan az veri ile eğitilmiştir. Buna karşın elde edilen sonuçlar umut vericidir. Yaklaşık 0,99 kesinlik ve 0,94 F1 skoru elde edilmiştir

6. SONUÇ VE ÖNERİLER

Kapsül ağlar kullanılarak eğitim yapılmış ve eğitim sonunda bir model çıkarılmıştır. Bulgular bölümünde verilen sonuçlara göre tüm istek türlerinde %97,43 oranında başarıyla sınıflandırma yapabilen bir model üretilmiştir. Ayrıca model zararsız istekleri %99,59 oranı gibi çok iyi bir oranda doğru bir şekilde sınıflandırdığı görülmektedir. Bu model daha sonra bir bulut sistemin parçası olarak kullanılmıştır. Bunun için bir ortam hazırlanmıştır. Hazırlanan bu ortam bulut sunuculara yüklenmiş ve orada çalışması sağlanmıştır. Web sitesine gelen isteklerden günlük kayıtları oluşturulmuş ve bu günlük kayıtları web projesi için yeni eğitilen modelle test edilmiştir. Bu aşamadan sonra günlük kaydı ve elde edilen sonuç başka bir sisteme aktarılmıştır. Senaryoda yönetim ekibi ilk iki sunucudaki gerçekleşen işlemleri konsol ekranından görmek yerine kendileri için oluşturulan yönetim panelinden görebilmektedir. Yönetim panelinden istenirse istekler CSV veya Excel formatında indirilebilmektedir. İndirilen bu dosyalar ile daha önceden eğitilen modele yeni veriler sağlanarak modelin daha iyi öğrenmesi sağlanabilmektedir.

Kapsül ağlar ile HTTP günlükleri analiz edilerek bir sisteme entegrasyonun yapılması konulu bilindiği kadarıyla daha önce herhangi bir çalışma olmadığından tez kapsamında kapsül ağlarla HTTP günlük veri setinin analizi yapılmış ve bu derin öğrenme yönteminin umut verici sonuçlar ürettiği görülmüştür.

Tez kapsamında kapsül ağlar kullanılarak eğitilen modelin web sayfası isteklerinin zararlı olup olmadığının tespit edilmesi gibi işlemlerin yerine getirebilmesi için modelin sisteme entegrasyonu yapılmış ve temel fonksiyonları yerine getirmesi sağlanmıştır. Eğitim aşaması ile ilgili iyileştirilmesi gereken noktalar vardır. Verilerin işlenmesi ve veri setinin sayısallaştırılması aşaması ile ilgili farklı yaklaşımlar denenebilir. Ayrıca kapsül ağlarının farklı varyasyonları bulunmaktadır. Model eğitim aşamasında diğer kapsül ağlar kullanılarak ya da hibrit çalışma yöntemi uygulanarak birden fazla makine öğrenmesi yöntemi kullanılarak daha iyi sonuçlar elde edilebileceği düşünülmektedir.

KAYNAKLAR

[1] Han, C., *"Enhanced cybersecurity for safe smart world," 2021 International Conference on Computational Science and Computational Intelligence (CSCI),* IEEE 2021, pp. 1886-1890, doi: 10.1109/CSCI54926.2021.00078.

[2] Nwosu, C. and Sykes, P., 2022, *"Visualizing The 50 Biggest Data Breaches From 2004–2021",* https://www.visualcapitalist.com/cp/visualizing-the-50-biggest-data-breaches-from-2004-2021/, [Ziyaret Tarihi: 07 Kasım 2022].

[3] IBM, 2021, *"Cost of a Data Breach Report 2021",* https://www.ibm.com/tr-tr/security/data-breach, [Ziyaret Tarihi: 07 Kasım 2022].

[4] Demir, B., 2015, *"Yazılım Güvenliği Saldırı ve Savunma",* 2. Baskı, Dikeyeksen, İstanbul, ISBN:9786058666085.

[5] Farooq, H. M. and Otaibi, N. M., *"Optimal Machine Learning Algorithms for Cyber Threat Detection",* 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim), 2018, pp. 32-37, doi: 10.1109/UKSim.2018.00018.

[6] Khraisat, A., Gondal, I., Vamplew, P. et al., *"Survey of intrusion detection systems: techniques, datasets and challenges",* Cybersecur 2, 20 (2019). https://doi.org/10.1186/s42400-019-0038-7.

[7] Seyyar, Y. E., Yavuz, A. G. and Ünver, H. M., *"An Attack Detection Framework Based on BERT and Deep Learning",* in IEEE Access, vol. 10, pp. 68633-68644, 2022, doi: 10.1109/ACCESS.2022.3185748.

[8] Ghani, I., Choi, E. and Min, D., *"Environment-Aware Communication Model in the Perspective of Web Ecosystems,",* 2008 International Conference on Convergence and Hybrid Information Technology, 2008, pp. 311-318, doi: 10.1109/ICHIT.2008.262.

[9] Nizam, Dr. A., 2011, *"Veri Tabanı Tasarımı İlişkisel veri Modeli ve Uygulamaları", 1. Basım,,* Papatya, İstanbul, ISBN:9786054220281.

[10] NTT, 2022, *"2022 Global Threat Intelligence Report",* https://www.security.ntt/global-threat-intelligence-report-2022, [Ziyaret Tarihi: 13 Kasım 2022].

[11] OWASP, 2021, *"OWASP Top 10 – 2021",* https://owasp.org/Top10/, [Ziyaret Tarihi: 13 Kasım 2022].

[12] OWASP, 2021, *"A01:2021 – Broken Access Control",* https://owasp.org/Top10/A01_2021-Broken_Access_Control/, [Ziyaret Tarihi: 13 Kasım 2022].

[13] OWASP, 2021, *"A02:2021 – Cryptographic Failures",* https://owasp.org/Top10/A02_2021- Cryptographic_Failures/, [Ziyaret Tarihi: 13 Kasım 2022].

[14] OWASP, 2021, *"A03:2021 – Injection",* https://owasp.org/Top10/A03_2021-Injection/, [Ziyaret Tarihi: 13 Kasım 2022].

[15] OWASP, 2021, *"A04:2021 – Insecure Design",* https://owasp.org/Top10/A04_2021-Insecure_Design/, [Ziyaret Tarihi: 13 Kasım 2022].

[16] OWASP, 2021, *"A05:2021 – Security Misconfiguration",* https://owasp.org/Top10/A05_2021-Security_Misconfiguration/, [Ziyaret Tarihi: 13 Kasım 2022].

[17] OWASP, 2021, *"A06:2021 – Vulnerable and Outdated Components",* https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/, [Ziyaret Tarihi: 13 Kasım 2022].

[18] OWASP, 2021, *"A07:2021 – Identification and Authentication Failures",* https://owasp.org/ Top10/A07_2021-Identification_and_Authentication_Failures/, [Ziyaret Tarihi: 13 Kasım 2022].

[19] OWASP, 2021, *"A08:2021 – Software and Data Integrity Failures",* https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/, [Ziyaret Tarihi: 13 Kasım 2022].

[20] OWASP, 2021, *"A09:2021 – Security Logging and Monitoring Failures",* https://owasp.org/ Top10/A09_2021-Security_Logging_and_Monitoring_Failures/, [Ziyaret Tarihi: 13 Kasım 2022].

[21] OWASP, 2021, *"A10:2021 – Server-Side Request Forgery (SSRF)",* https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_(SSRF)/, [Ziyaret Tarihi: 13 Kasım 2022].

[22] Nilsson, N. J., Çeviri Doğan, M., 2019, *"Yapay Zekâ Geçmişi ve Geleceği",* 2. Baskı, Boğaziçi Üniversitesi Yayınevi, İstanbul, ISBN: 9786056842511.

[23] Nabiyev, Prof. Dr. V. V., 2016, *"Yapay Zekâ: İnsan – Bilgisayar Etkileşimi",* 5. Baskı, Seçkin Yayıncılık, Ankara, ISBN:9789750237270.

[24] Bilgin, Dr. M., 2018, *"Makine Öğrenmesi",* 2. Basım, Papatya Yayıncılık Eğitim, İstanbul, ISBN: 9786059594257.

[25] Wibowo, V. V. P., Rustam, Z., Laeli, A. R. and Sa'id, A. A., *"Logistic Regression and Logistic Regression-Genetic Algorithm for Classification of Liver Cancer Data",* 2021 International Conference on Decision Aid Sciences and Application (DASA), 2021, pp. doi: 10.1109/DASA53625.2021.9682242.

[26] Géron, A. ,2022, *"Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow",* 3. Baskı, O'Reilly Media, Inc., Kanada, ISBN: 9781492032649.

[27] Kızrak, M. A., 2018, *"ŞU KARA KUTUYU AÇALIM: Yapay Sinir Ağları",* https://ayyucekizrak.medium.com/şu-kara-kutuyu-açalim-yapay-sinir-ağları-7b65c6a5264a, [Ziyaret Tarihi: 17 Kasım 2022].

[28] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al., *"Review of deep learning: concepts, CNN architectures, challenges, applications, future directions.",* J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8.

[29] Yadav, A., Vishwakarma, D.K., *"Sentiment analysis using deep learning architectures: a review.",* Artif Intell Rev 53, 4335–4385 (2020). https://doi.org/10.1007/s10462-019-09794-5.

[30] Berman, D., Buczak, A., Chavis, J., and Corbett, C., *"A Survey of Deep Learning Methods for Cyber Security",* Information, vol. 10, no. 4, p. 122, Apr. 2019, doi: 10.3390/info10040122.

[31] Stanford Üniversitesi, *"CS231n Convolutional Neural Networks for Visual Recognition",* https://cs231n.github.io/convolutional-networks/, [Ziyaret Tarihi: 18 Kasım 2022].

[32] Yazan, E. and Talu, M. F., *"Comparison of the stochastic gradient descent based optimization techniques,",* 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), 2017, pp. 1-5, doi: 10.1109/IDAP.2017.8090299.

[33] Ruder, S., 2016, *"An overview of gradient descent optimization algorithms.",* ArXiv, https://doi.org/10.48550/arXiv.1609.04747.

[34] Young, T., Hazarika, D., Poria, S. and Cambria, E., *"Recent Trends in Deep Learning Based Natural Language Processing [Review Article]",* in IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55-75, Aug. 2018, doi: 10.1109/MCI.2018.2840738.

[35] Sabour, S., Frosst, N., Hinton, G. E. ,2017, *"Dynamic Routing Between Capsules",* ArXiv, https://doi.org/10.48550/arXiv.1710.09829.

[36] Jia, Z., Shen, C., Yi, X., Chen, Y., Yu , T. and Guan, X., *"Big-data analysis of multi-source logs for anomaly detection on network-based system,",* 2017 13th IEEE Conference on Automation Science and Engineering (CASE), 2017, pp. 1136-1141, doi: 10.1109/COASE.2017.8256257.

[37] Gharibeh, S., Melhem , S. and Najadat, H., *"Classification on Web Application Requests,",* 2020 11th International Conference on Information and Communication Systems (ICICS), 2020, pp. 1-5, doi: 10.1109/ICICS49469.2020.239537.

[38] Wan, W., Shi, X., Wei, J., Zhao, J. and Long, C., *"ELSV: An Effective Anomaly Detection System from Web Access Logs,",* 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), 2021, pp. 1-6, doi: 10.1109/IPCCC51483.2021.9679413.

[39] Yan , L. and Xiong, J., *"Web-APT-Detect: A Framework For Web-Based Advanced Persistent Threat Detection Using Self-Translation Machine With Attention",* in IEEE

Letters of the Computer Society, vol. 3, no. 2, pp. 66-69, 1 July-Dec. 2020, doi: 10.1109/LOCS.2020.2998185.

[40] Yu, L., Chen, L., Dong, J., Li, M., Liu, L., Zhao, B., Zhang, C., *"Detecting Malicious Web Requests Using an Enhanced TextCNN,",* 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), 2020, pp. 768-777, doi: 10.1109/COMPSAC48688.2020.0-167.

[41] Bhatnagar, M., Rozinaj, G. and Yadav, P. K., *"Web Intrusion Classification System using Machine Learning Approaches",* 2022 International Symposium ELMAR, 2022, pp. 57-60, doi: 10.1109/ELMAR55880.2022.9899790.

[42] Jemal, I., M. Haddar, A., Cheikhrouhou, O. and Mahfoudhi, A., *"M-CNN: A New Hybrid Deep Learning Model for Web Security",* 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA), 2020, pp. 1-7, doi: 10.1109/AICCSA50499.2020.9316508.

[43] Das, S., Ashrafuzzaman, M., Sheldon, F. T. and Shiva, S., *"Network Intrusion Detection using Natural Language Processing and Ensemble Machine Learning",* 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2020, pp. 829-835, doi: 10.1109/SSCI47803.2020.9308268.

[44] Niu, Q. and Li, X., *"A High-performance Web Attack Detection Method based on CNN-GRU Model",* 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2020, pp. 804-808, doi: 10.1109/ITNEC48623.2020.9085028.

[45] Wu, S. and Wang, Y., *"Attention-based Encoder-Decoder Recurrent Neural Networks for HTTP Payload Anomaly Detection",* 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications: Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2021, pp. 1452-1459, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00196.

[46] Ito, M. and Iyatomi, H., *"Web application firewall using character-level convolutional neural network",* 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), 2018, pp. 103-106, doi: 10.1109/CSPA.2018.8368694.

[47] Işıker, B. and Sogukpinar, I., *"Machine Learning Based Web Application Firewall",,* 2021 2nd International Informatics and Software Engineering Conference (IISEC), 2021, pp. 1-6, doi: 10.1109/IISEC54230.2021.9672335.

[48] Tong, X., Wang, J., Zhang, C., Wang, R., Ge, Z., Liu, W., Zhao, Z., *"A Content-Based Chinese Spam Detection Method Using a Capsule Network With Long-Short Attention",* in IEEE Sensors Journal, vol. 21, no. 22, pp. 25409-25420, 15 Nov.15, 2021, doi: 10.1109/JSEN.2021.3092728.

[49] Ouyang, M., Das, R. K., Yang , J. and Li, H. , *"Capsule Network based End-to-end System for Detection of Replay Attacks,",* 2021 12th International Symposium on Chinese Spoken

Language Processing (ISCSLP), 2021, pp. 1-5, doi: 10.1109/ISCSLP49672.2021. 9362111.

[50] Huang, Y., Qin , J. and Wen, W. , *"Phishing URL Detection Via Capsule-Based Neural Network",* 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), 2019, pp. 22-26, doi: 10.1109/ICASID.2019.8925000.

[51] Li, J., Zhao, Q., Li, N., Ma, L., Xia, X., Zhang, X., Ding, N., Li, N., *"A Survey on Capsule Networks: Evolution, Application, and Future Development",* 2021 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS), 2021, pp. 177-185, doi: 10.1109/HPBDIS53214.2021.9658349.

[52] Patrick, M. K., Adekoya, A. F., Mighty, A. A. and Edward, B. Y., *"Capsule Networks – A survey",* Journal of King Saud University - Computer and Information Sciences,Volume 34, Issue 1,2022,Pages 1295-1310,ISSN 1319-1578,https://doi.org/10.1016/j.jksuci.2019.09.014.

[53] Hinton, G.E., Krizhevsky, A., Wang, S.D., 2011, *Transforming Auto-Encoders. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds) Artificial Neural Networks and Machine Learning – ICANN 2011,* Lecture Notes in Computer Science, vol 6791. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21735-7_6.

[54] Hinton, G., Sabour, S. and Frosst, N., 2018, *Matrix capsules with EM routing,* OpenrReview.Net, https://openreview.net/pdf?id=HJWLfGWRb.

[55] Pechyonkin, M., 2017, *"Understanding Hinton's Capsule Networks. Part I: Intuition",* https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b, [Ziyaret Tarihi: 23 Kasım 2022].

[56] Kızrak, M. A., 2018, *"Yapay Zekanın Yeni ve Çekici Mimarisi Kapsül Ağına Uygulamalı Bir Bakış",* https://ayyucekizrak.medium.com/ef7310e3d847 [Ziyaret Tarihi: 27 Kasım 2022].

[57] Tran, A. -T., Luong, T. D. , Pham , X. S.,and Tran, T. L. , *"Deep Models with Differential Privacy for Distributed Web Attack Detection",* 2022 14th International Conference on Knowledge and Systems Engineering (KSE), 2022, pp. 1-6, doi: 10.1109/KSE56063.2022.9953788.

[58] Giménez, C. T., Villegas, A. P., Marañón, G. Á., *"HTTP DATASET CSIC 2010",* 2010, Information Security Institute of CSIC (Spanish Research National Council), https://www.tic.itefi.csic.es/dataset/ [Ziyaret Tarihi: 16 Ekim 2022].

[59] Kızrak, M. A., *Python ile Derin Öğrenme,* Udemy, https://www.udemy.com/course/derin-ogrenmeye-giris/learn/lecture/11263250, [Ziyaret Tarihi: 07.12.2022].

[60] Amazon, *Splitting the Data into Training and Evaluation Data,* https://docs.aws.amazon.com/machine-learning/latest/dg/splitting-the-data-into-training-and-evaluation-data.html, [Erişim Tarihi: 08.01.2023].

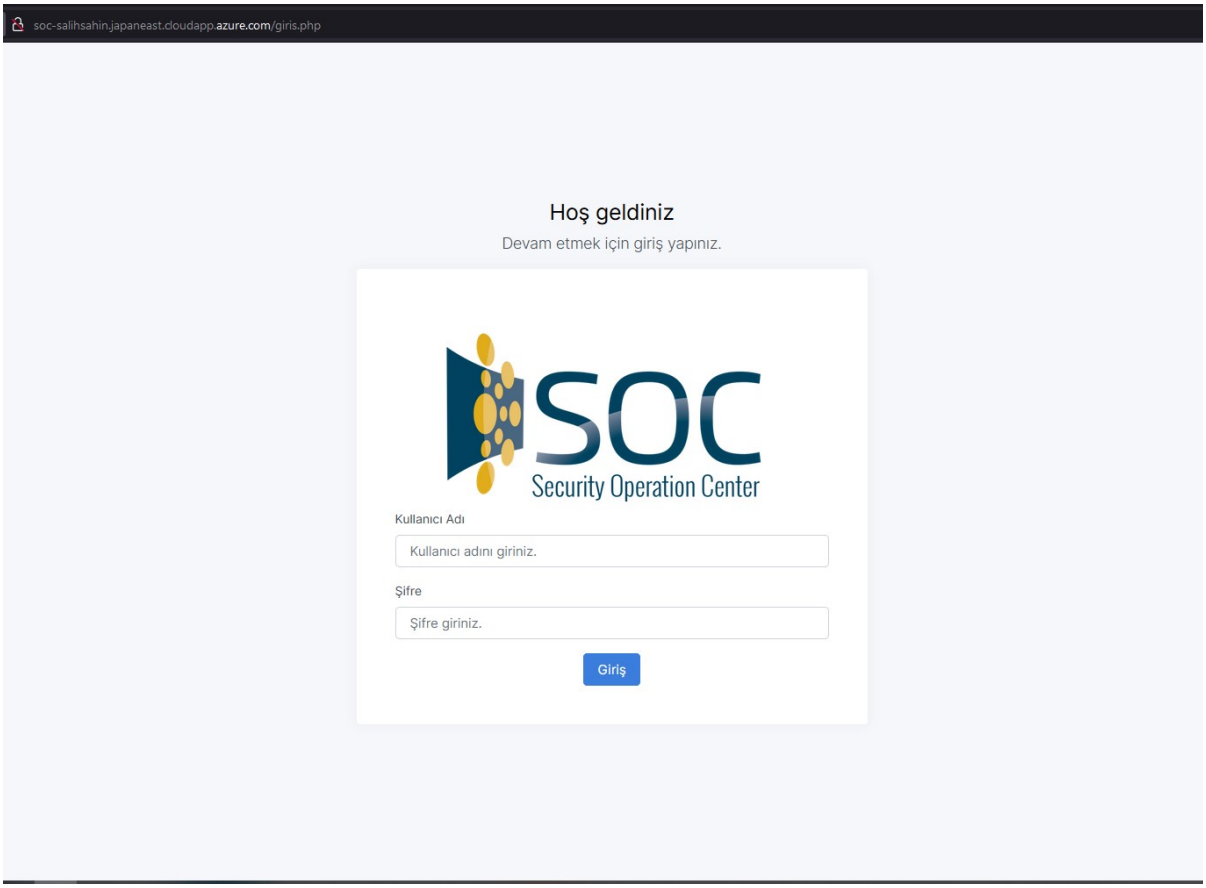
[61] Apache Software Foundation, *"Apache Kafka",* "https://kafka.apache.org/" [Erişim Tarihi: 01.12.2022].

[62] Apache Software Foundation, *"Welcome to Apache ZooKeeper™",* "https://zookeeper.apache.org/", [Erişim Tarihi: 01.12.2022].

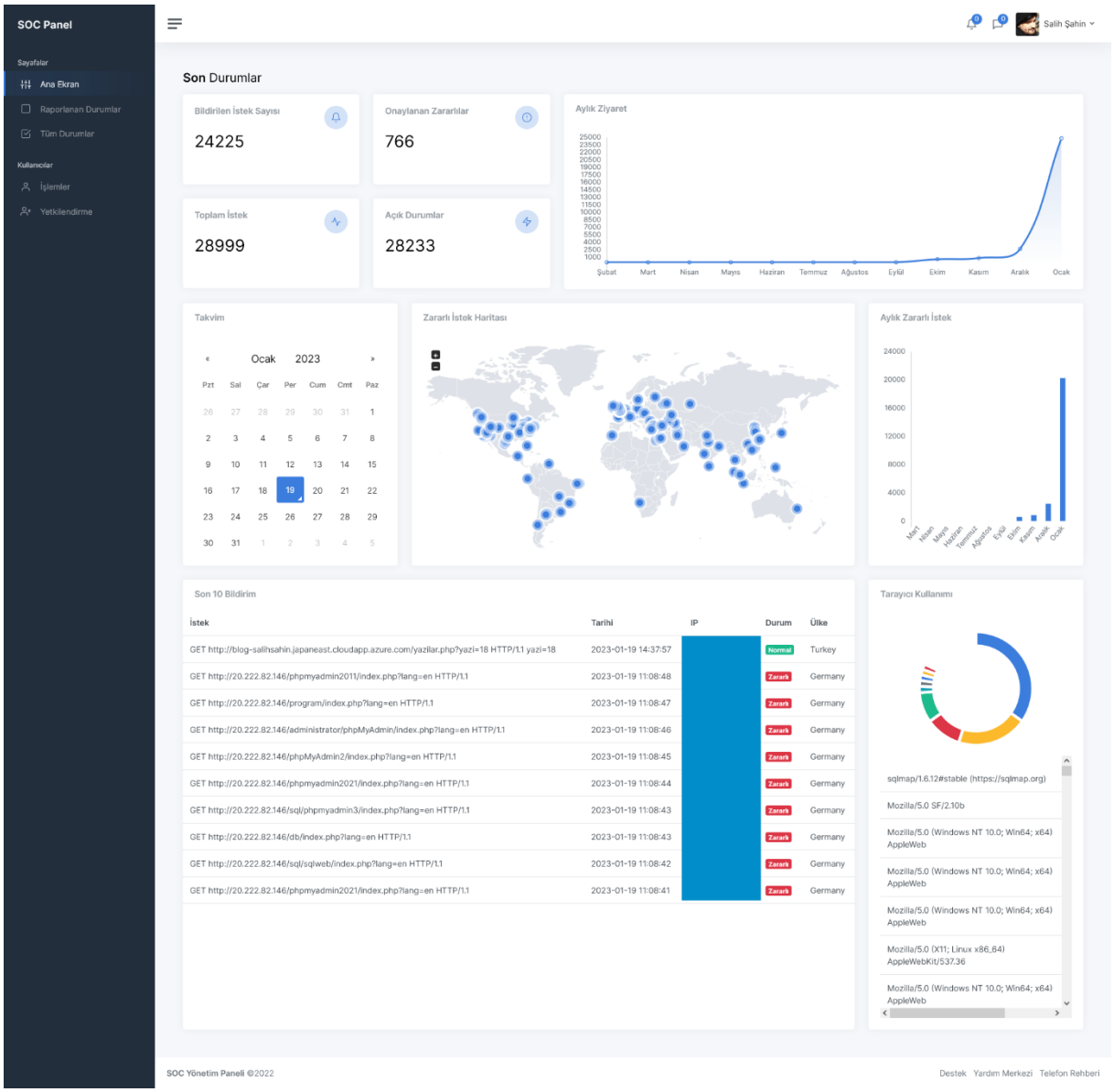
[63] Confluent Inc., *"KRaft: Apache Kafka Without ZooKeeper",* "https://developer.confluent.io/learn/kraft/", [Erişim Tarihi: 01.12.2022].

EKLER

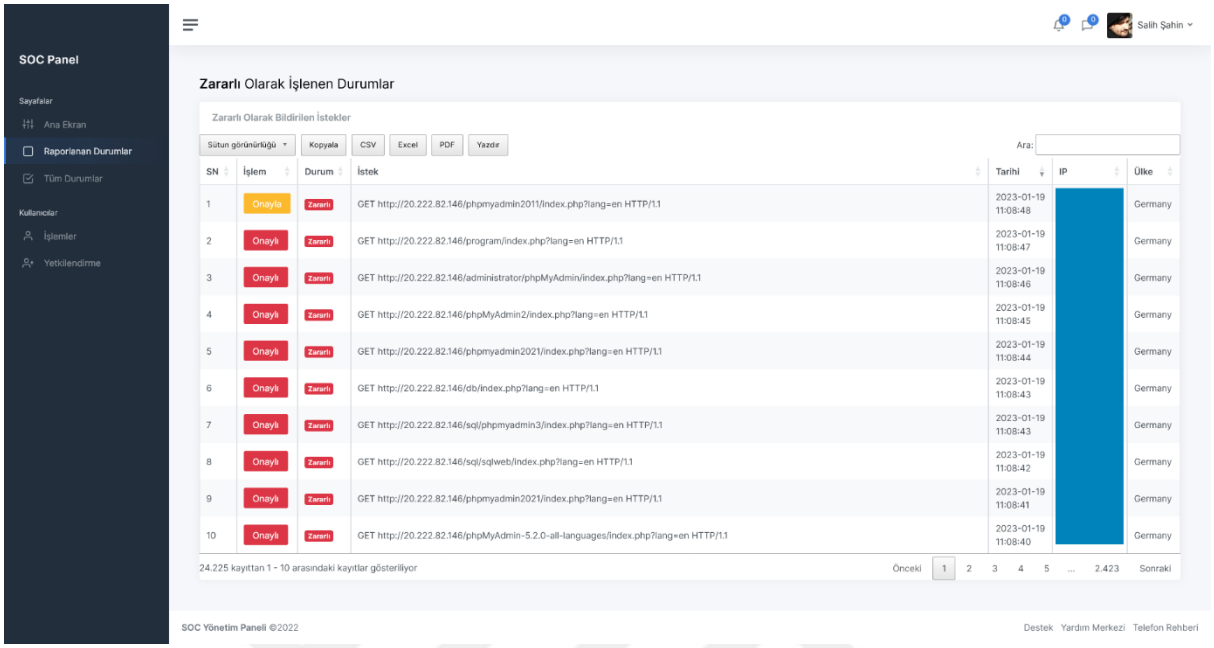
EK 1. Yönetim Paneli Giriş Ekranı



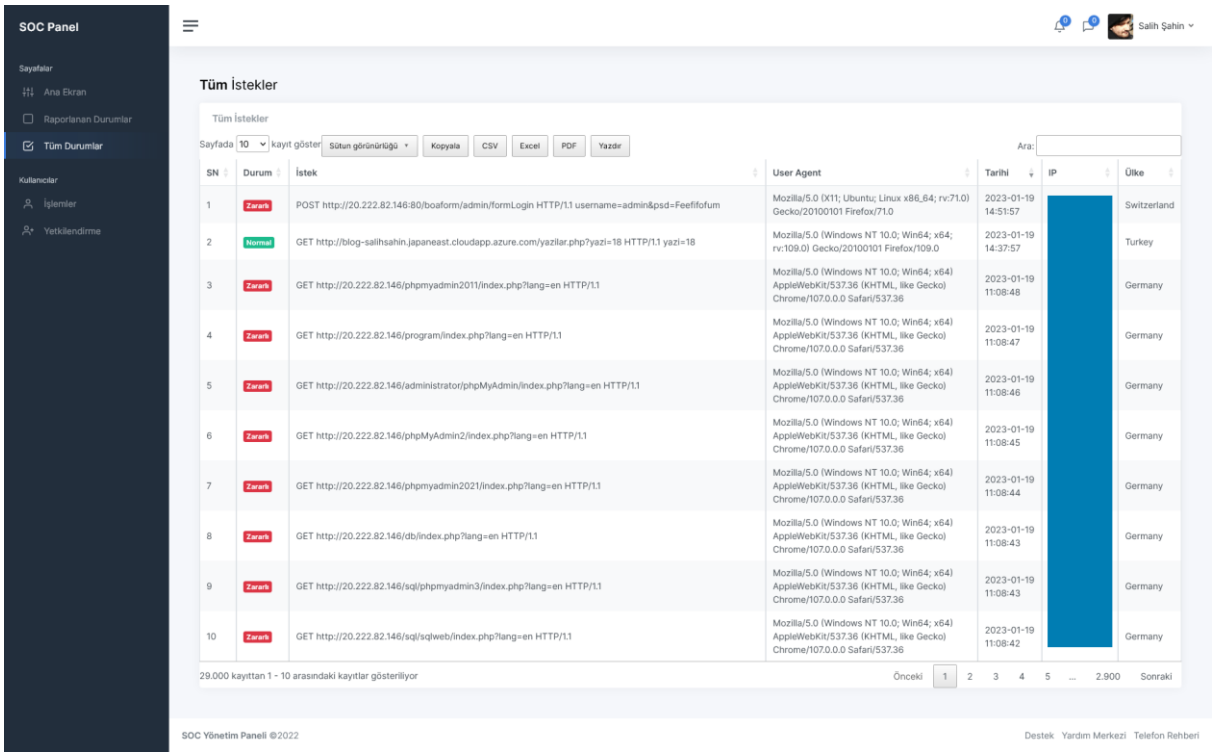
EK 2. Yönetim Paneli Ana Sayfa



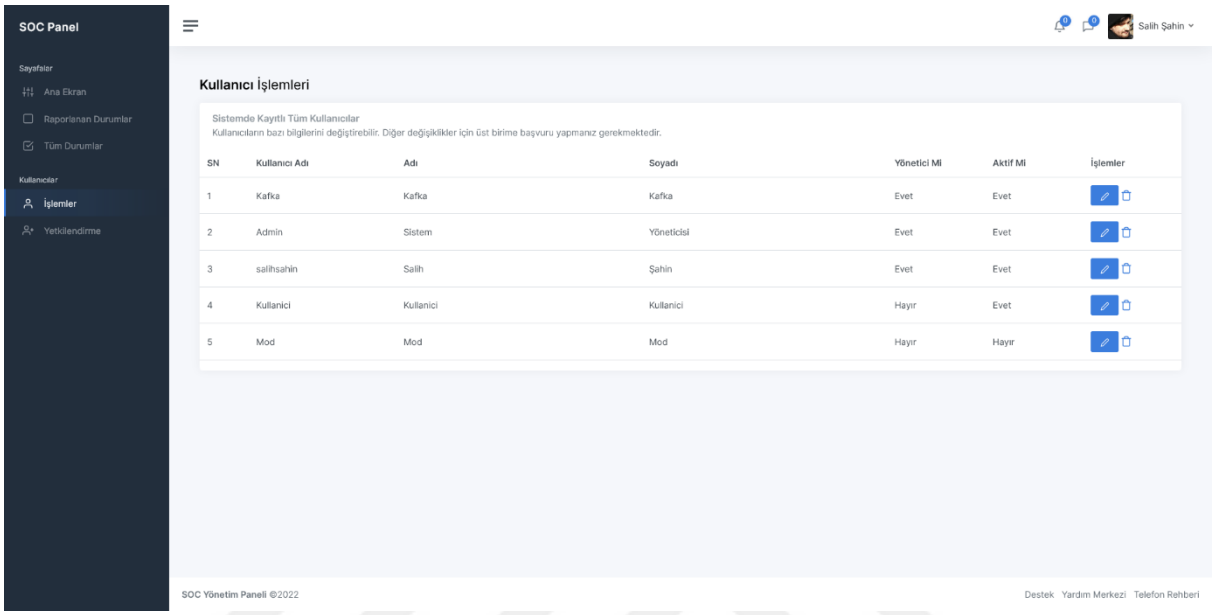
EK 3. Zararlı Olarak Tespit Edilen İsteklerin İşlem Sayfası



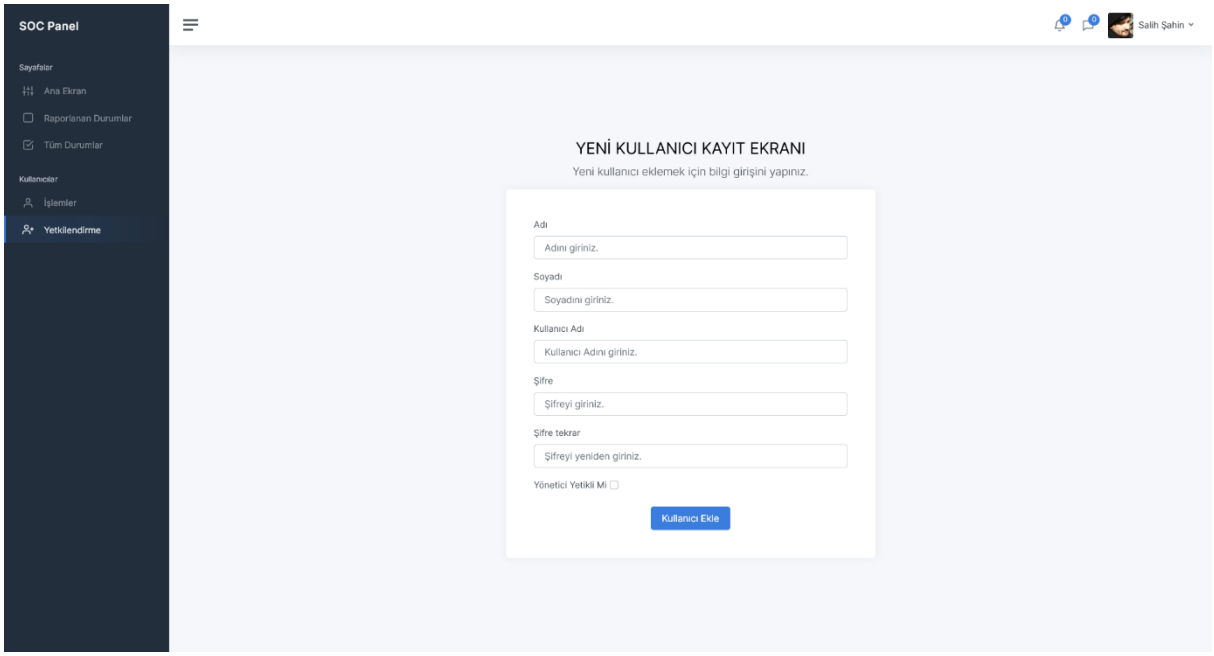
EK 4. Tüm İsteklerin Listelendiği Sayfa



EK 5. Yönetici Yetkili Kullanıcıların Kullanıcı İşlemlerini Yaptığı Sayfa



EK 6. Yönetici Yetkili Kullanıcıların Yeni Kullanıcı Eklediği Sayfa



EK 6. KafkaML Sunucusu Servis Durum Kontrolü

