MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TOPLULUK ÖĞRENMESİ KULLANILARAK ZARARLI ALAN ADLARININ SINIFLANDIRILMASI

AHMET AYDIN

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgi Güvenliği Mühendisliği Yüksek Lisans Programı

DANIŞMAN
Doç. Dr. Kazım YILDIZ

İSTANBUL, 2023

MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TOPLULUK ÖĞRENMESİ KULLANILARAK ZARARLI ALAN ADLARININ SINIFLANDIRILMASI

AHMET AYDIN

527322606

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgi Güvenliği Mühendisliği Yüksek Lisans Programı

DANIŞMAN
Doç. Dr. Kazım YILDIZ

İSTANBUL, 2023

ÖNSÖZ

Tez çalışmamın araştırılmasında ve yürütülmesinde her daim destek olan, her fırsatta yardımlarıyla yol gösteren değerli danışmanım Doç. Dr. Kazım YILDIZ'a,

Tez çalışmam boyunca verdikleri destek için kıymetli eşime, bugünlere gelmemde en büyük pay sahibi olan değerli anneme, babama ve ağabeylerime

Teşekkürlerimi sunarım.

Aralık, 2023 **Ahmet AYDIN**

İÇİNDEKİLER

	SAYFA
ÖNSÖZ	**i**
İÇİNDEKİLER	**ii**
ÖZET	**iv**
ABSTRACT	**v**
SEMBOLLER	**vi**
KISALTMALAR	**vii**
ŞEKİL LİSTESİ	**viii**
TABLO LİSTESİ	**x**
1. GİRİŞ	**1**
1.1. GENEL BAKIŞ	**3**
1.1.1. Problemin tanımı	3
1.1.2. Amaç ve hedef	4
1.1.3. Ana katkılar	4
1.1.4. Tez organizasyonu	4
1.2. YAPAY ZEKA	**5**
1.2.1. Makine öğrenmesi	5
1.2.2. Bagging algoritması	13
1.2.3. Random Forest algoritması	14
1.2.4. Extra Trees algoritması	15
1.2.5. Extreme Gradient Boosting algoritması	15
1.2.6. Adaptive Boosting algoritması	16
1.2.7. SMOTE	17
1.2.8. NearMiss	18
1.2.9. Yapay sinir ağları	19
1.2.10. Derin öğrenme	23
1.3. LİTERATÜR ÇALIŞMALARI	**26**
2. MATERYAL VE YÖNTEM	**31**
2.1. VERİ KÜMESİ	**31**
2.2. SİSTEM MİMARİSİ	**33**
2.3. MODEL YAPISI	**35**

3. BULGULAR VE TARTIŞMA **37**
3.1. DENEY ORTAMI **37**
3.2. PERFORMANS ÖLÇÜTLERİ **37**
3.2.1. Doğruluk 37
3.2.2. Kesinlik 37
3.2.3. Duyarlılık 38
3.2.4. F1 değeri 38
3.2.5. Test süresi 38
3.3. DENEY SONUÇLARI **38**
3.3.1. Topluluk öğrenmesi yöntemleri ile elde edilen sonuçlar 38
3.3.2. Derin topluluk öğrenmesi yöntemleri ile elde edilen sonuçlar 45
3.4. DİĞER SONUÇLARLA KARŞILAŞTIRMA **46**
4. SONUÇLAR **48**

ÖZET

TOPLULUK ÖĞRENMESİ KULLANILARAK ZARARLI ALAN ADLARININ SINIFLANDIRILMASI

İnternet kullanımı her geçen gün daha da yaygınlaşmakta ve insan hayatındaki varlığını giderek artırmaktadır. Kullanımın artmasıyla birlikte siber saldırılar da her geçen gün artmakta ve hem kullanıcıların hem de sistemlerin güvenliği tehdit altında kalmaktadır. Doğrudan sistemleri hedef alan siber saldırılar mevcut olmakla birlikte kullanıcıları zararlı web sitelerine yönlendirerek çeşitli çıkarlar elde etmeye çalışan siber saldırılar da mevcuttur. Böyle bir ortamda alan adlarının zararlı olup olmadığının tespiti büyük önem arz etmektedir. Bu alandaki ilk çalışmalar zararlı alan adlarından oluşan bir kara liste oluşturarak erişilmek istenen web sitesinin kara listede olup olmadığını tespit etmeye yönelmiştir. Ancak bu yaklaşım, kullanıcıları sadece daha önce zararlı olduğu tespit edilen web sitelerinden korumaktadır. İlk kez kullanılacak zararlı web sitelerini tespit edememektedir. Daha sonra makine öğrenmesi yöntemlerinin de gelişimiyle kötücül web sitelerinin yapay zeka ile tespit edilip sınıflandırılması çalışmaları hız kazanmıştır. Bu alanda makine öğrenmesi ve derin öğrenme yöntemleriyle çeşitli çalışmalar yapılmakta ve kayda değer başarılar elde edilmektedir.

Bu çalışma kapsamında topluluk öğrenmesi yöntemleri kullanılarak zararlı alan adı tespiti ve sınıflandırması yapılmıştır. Literatürde bu alanda makine öğrenmesi ve derin öğrenme teknikleriyle araştırmalar olsa da topluluk öğrenmesi yöntemleri kullanılan bir çalışma henüz bulunmamaktadır. Ayrıca mevcut çalışmaların birçoğunda sınıflandırma sürelerine yer verilmemiştir. Ancak operasyonel anlamda işletilebilir bir sistem kurmak için modelin doğruluğu kadar çalışma hızı da önemlidir. Bu sebeple çeşitli topluluk öğrenmesi yöntemleriyle sınıflandırma performanslarını doğruluk oranı ve sınıflandırma süresi açısından kıyaslayan bir çalışma yapılmıştır. Çalışma kapsamında Canadian Institute for Cybersecurity tarafından sağlanan CIC-Bell-DNS 2021 veri kümesi kullanılarak 4 farklı sınıf (phishing, malware, spam, benign) için sınıflandırma yapılmıştır.

ARALIK, 2023 **Ahmet AYDIN**

ABSTRACT

MALICIOUS DOMAIN NAME CLASSIFICATION USING ENSEMBLE LEARNING

Internet usage is becoming increasingly prevalent, exerting a growing influence on people's daily lives. As usage surges, cyberattacks are on the rise, endangering the security of both users and systems. While there are direct cyberattacks targeting systems, there are also those that attempt to lead users to harmful websites to pursue various interests. In this context, it's vital to determine the malicious nature of domain names. Initial efforts in this domain focused on establishing a blacklist of malicious domain names to identify whether a desired website was on that list. However, this approach only safeguards users from known malicious websites, failing to detect newly used malicious sites. With the advancement of machine learning, artificial intelligence-based methods have gained momentum in identifying and categorizing malicious websites.

This study employs ensemble learning methods for malicious domain detection and classification. While research on machine learning and deep learning techniques exists in the literature, there has been no study involving ensemble learning methods in this context. Additionally, most existing studies do not address classification times. However, for the establishment of an operational system, the classification speed of the model is as crucial as the model's accuracy. Hence, a study comparing classification performance in terms of accuracy and classification time using various ensemble learning methods was conducted. Within this study, domain names were classified into four distinct categories (phishing, malware, spam, benign) using the CIC-Bell-DNS 2021 dataset provided by the Canadian Institute for Cybersecurity.

DECEMBER, 2023 **Ahmet AYDIN**

SEMBOLLER

i	:	Gizli katman sayısı
j	:	Nöron sayısı
x	:	Girdi vektörü
y	:	Çıktı vektörü
w_n	:	Nöronun ağırlık vektörü
z	:	Nöronun ön yargı değeri
g	:	Aktivasyon fonksiyonu
e	:	Euler sayısı
Σ	:	Toplam
f	:	Karar fonksiyonu
t	:	Adım sayısı
u	:	Aktivasyon potansiyeli
ϵ	:	Çok küçük bir değer
Wt	:	t anındaki ağırlık vektörü
θ	:	Bias
σ	:	Unutma kapısı
s_t	:	t anındaki gizli durum
o_t	:	t anındaki çıktı
x_t	:	t anındaki girdi vektörü
U	:	Girdi ile gizli katman arasındaki ağırlık matrisi
W	:	İki gizli katman arasındaki ağırlık matrisi
V	:	Gizli katman ile çıktı arasındaki ağırlık matrisi

KISALTMALAR

MLP	:	Multilayer Perceptron
LSTM	:	Long Short Term Memory
IP	:	Internet Protocol
IPS	:	Intrusion Prevention System
KNN	:	K-Nearest Neighbors
IDS	:	Intrusion Detection System
DNS	:	Domain Name System
TCP	:	Transmission Control Protocol
CNN	:	Convolutional Neural Network
RNN	:	Recurrent Neural Network
SMOTE	:	Synthetic Minority Oversampling Technique
XGBoost	:	eXtreme Gradient Boosting
AdaBoost	:	Adaptive Boosting
URL	:	Uniform Resource Locator
TP	:	True Positive
TN	:	True Negative
FP	:	False Positive
FN	:	False Negative
MP	:	McCulloch-Pitts
SVM	:	Support Vector Machine
EFW-LS-SVM	:	Fuzzy-Weighted Least Squares Support Vector Machine
ANN	:	Artificial Neural Network
CIC	:	Canadian Institute for Cybersecurity
MÖ	:	Milattan Önce
TLD	:	Top Level Domain
SLD	:	Second Level Domain
ASN	:	Autonomous System Number
TTL	:	Time to Live

ŞEKİL LİSTESİ

SAYFA

Şekil 1.1. Makine öğrenmesi akış diyagramı. 7

Şekil 1.2. Denetimli öğrenme 8

Şekil 1.3. k-ortalama algoritması ile veri kümeleme 9

Şekil 1.4. Denetimsiz öğrenme 9

Şekil 1.5. Yarı denetimli öğrenme akış diyagramı 10

Şekil 1.6. Pekiştirmeli öğrenmede kontrol döngüsü 11

Şekil 1.7. Topluluk öğrenmesi yöntemleriyle değişkenliğin azaltılması 12

Şekil 1.8. Bagging algoritması çalışma prensibi. 14

Şekil 1.9. Random Forest algoritması. 15

Şekil 1.10. XGBoost algoritması. 16

Şekil 1.11. Adaboost algoritması çalışma prensibi. 17

Şekil 1.12. SMOTE 18

Şekil 1.13. NearMiss tekniği çeşitleri. 19

Şekil 1.14. İnsan beynindeki nöronun yapısı 20

Şekil 1.15. MP Nöron 20

Şekil 1.16. Yapay sinir ağı. 22

Şekil 1.17. Yaygın kullanılan aktivasyon fonksiyonları 23

Şekil 1.18. İki gizli katmandan oluşan bir ileri beslemeli sinir ağı yapısı 24

Şekil 1.19. Yinelemeli sinir ağı mimarisi 25

Şekil 1.20. LSTM mimarisi A. mimariye genel bakış, B. LSTM hücre yapısı 26

Şekil 1.21. DNS paketi alma 27

Şekil 1.22. EFW-LS-SVM modeli akış diyagramı..... 29

Şekil 2.1. Çalışma kapsamında uygulanan akış diyagramı. 34

Şekil 2.2. Önerilen sistem mimarisi. 34

Şekil 2.3. Çalışmada kullanılan derin öğrenme mimarisi. 36

Şekil 3.1. Topluluk öğrenmesi ile ikili sınıflandırma performans özeti..... 41

Şekil 3.2. Topluluk öğrenmesi ile çoklu sınıflandırma performans özeti 44

TABLO LİSTESİ

SAYFA

Tablo 1.1. 165 veri kümesinde 14 algoritmanın üstünlük karşılaştırması 13

Tablo 2.1. Mahdavifar ve arkadaşlarının veri kümesi için topladığı özellikler. 31

Tablo 2.2. Tez kapsamında kullanılan öznitelikler. 33

Tablo 2.3. Algoritmaların parametre değerleri. 35

Tablo 3.1. Topluluk öğrenmesi yöntemleri ile kısmen dengeli veri kümesi üzerinde ikili sınıflandırma sonuçları. 39

Tablo 3.2. Aşağı örnekleme yapılmış veri kümesi üzerinde topluluk öğrenmesi yöntemleri ile ikili sınıflandırma sonuçları. 39

Tablo 3.3. Yukarı örnekleme yapılmış veri kümesi üzerinde topluluk öğrenmesi yöntemleri ile ikili sınıflandırma sonuçları. 40

Tablo 3.4. Kısmen dengeli veri kümesi üzerinde topluluk öğrenmesiyle çoklu sınıflandırma sonuçları. 42

Tablo 3.5. Aşağı örneklemeyle dengelenmiş veri kümesi üzerinde topluluk öğrenmesiyle çoklu sınıflandırma sonuçları. 42

Tablo 3.6. Yukarı örneklemeyle dengelenmiş veri kümesi üzerinde topluluk öğrenmesiyle çoklu sınıflandırma sonuçları. 42

Tablo 3.7. Derin topluluk öğrenmesi modeliyle yukarı örneklenmiş veri kümesi üzerinde yapılan ikili sınıflandırma sonucu. 45

Tablo 3.8. Derin topluluk öğrenmesi modeliyle yukarı örneklenmiş veri kümesi üzerinde yapılan çoklu sınıflandırma sonucu. 45

Tablo 3.9. Literatürle karşılaştırma. 47

1. GİRİŞ

İnsanlık tarihinin başından beri insanoğlunun en temel ihtiyaçlarından biri de iletişim olmuştur. Bu ihtiyaç zaman içerisinde insanların farklı çözümler geliştirmesine ön ayak olmuş ve dumanla haberleşmeden mektup göndermeye ve güvercinle iletişime kadar pek çok farklı iletişim aracının doğuşuna yol açmıştır. Teknolojinin gelişmesiyle birlikte iletişim araçları da günden güne bu gelişimde kendine yer bulmayı başarmış ve pek çok farklı şekilde kendisini göstermiştir. İlerleyen zamanlarda telgraf ve telefonla kurulan iletişim yıllar içerisinde çok daha hızlı şekilde kurulabilir olmuştur. Bilgisayarın icadı sonrasında kişisel kullanım alanında yaygınlaşmasıyla bilgisayarların da birbiri ile iletişimini sağlamak ve dış dünyayla haberleşebilmek iletişim alanında bir çığır açmıştır. İlk olarak 1960 yılında araştırmacıların kendi arasındaki iletişimi sağlamak için icat edilen internet ilk zamanlarda sadece kapalı ağlarda kullanılmaktaydı. Ancak 1963 yılında Transfer Control Protocol/Internet Protocol (TCP/IP)'un belirlenmesiyle birlikte tüm bilgisayarların ve ağların birbiriyle ortak bir dil aracılığıyla iletişim kurabilmesi sağlandı. Böylelikle günümüzdeki internet keşfedilmiş oldu [1].

İnternet ilk olarak araştırmacılar arasındaki iletişimi kurmak için oluşturulmuş olsa da zaman içerisinde bu amacının çok ötesine ulaşmıştır. İnternet, bilgiye ve hizmete ulaşımı saniyeler içerisinde mümkün kılabildiği için kendisine çok yaygın bir yer bulabilmiştir. Öyle ki günümüzde artık internet hemen hemen tüm sektör ve hizmetlerde yaygın şekilde kullanılmaktadır. Kurumlar müşterilerine daha hızlı ve kaliteli hizmet verebilmek için hizmetlerini internete taşımaktadırlar. Öyle ki resmi kurumlar dahi artık pek çok sayıda hizmeti internetten vermektedir. Bazı ülkelerde başkanlık seçimleri dahi internet hizmetleri üzerinden elektronik ortamda yapılabilmektedir.

Kullanım alanının bu denli artışı birtakım güvenlik risklerini de beraberinde getirmiştir. Tarih boyunca iyi niyetli kişiler olduğu kadar maalesef kötü niyetli kişilerin de varlığı yadsınamaz bir gerçek olmuştur. Ne yazık ki bu durum internet kullanılan sahalarda da kendini göstermiş ve zaman içerisinde internet dünyasında pek çok siber saldırı boy göstermiştir. Öyle ki bu saldırılar arasında Stuxnet saldırısında olduğu gibi devletlerin nükleer programlarını hedef alan siber saldırılar dahi olmuştur [2].

Siber saldırılar kimi zaman doğrudan kullanıcı ya da sistemleri hedef alırken kimi zaman da zararlı içerik içeren web siteleri şeklinde kendini gösterebilmektedir. Bu sebeple

kullanıcıları zararlı içerik içeren web sitelerinden korumak için zararlı alan adlarını tespit etmek önem arz etmektedir. Tüm bu saldırılar hem çok ciddi maddi kayıplara hem de manevi kayıplara yol açabilmektedir. Yapılan bir araştırmaya göre siber suçlar her yıl dünya üzerinde yaklaşık 400 milyar dolar değerinde bir zarara yol açmaktadır [3]. Risklerin bu denli büyük boyutta oluşu beraberinde güvenlik tedbirlerine olan ihtiyacı doğurmuş ve siber güvenlik kavramı ortaya çıkmıştır. Sanal ortamlardaki risklere karşı sistemleri ve kullanıcıları korumak üzere gelişen bu kavram günümüzde hayatımızın bir parçası haline gelmiştir. Bu kapsamda siber güvenliğin sağlanması için çeşitli güvenlik yazılımları ve donanımlar geliştirilmiştir. Kimi zaman bunlar güvenlik duvarı vb. cihazlar olarak karşımıza çıkmakta kimi zaman ise antivirüs yazılımları ya da saldırı tespit sistemleri olarak kendini göstermektedir.

Güvenlik yazılımları önceleri zararlı yazılımların özet değerlerini, kendi virüs imza veri tabanlarında tutup herhangi bir yazılımın zararlı olup olmadığını bu veri tabanındaki verilerle karşılaştırarak tespit ederlerdi. Ancak gün geçtikçe saldırganlar da kendilerini geliştirerek zararlı yazılımın davranışını değiştirmeden sadece tanımlayıcıların farklı olmasını sağlayacak şekilde yöntemler geliştirmişlerdir ve bu çalışma metoduna sahip güvenlik yazılımlarını atlatmayı başarmışlardır. Saldırganların aldıkları bu tedbir ise güvenlik yazılımı geliştiren tarafları karşı tedbir almaya itmiş ve zararlı yazılımları yapay zekâ kullanarak tespit etmeye yarayan yöntemler geliştirmelerine teşvik etmiştir. Yapay zekanın siber güvenlik alanında kullanılmasına yapılan yatırımlar her yıl artış göstermektedir ve bir araştırmaya göre 2032 yılına kadar dünya üzerinde yapay zekanın siber güvenlik pazarındaki payının yaklaşık 91,7 milyar dolara ulaşacağı tahmin edilmektedir [4].

Siber güvenlik alanında yapay zekanın kullanımı ile saldırı tespit işlemleri insan etkileşimli sistemlere göre daha hızlı işletilebilir duruma gelmiştir. Bu alandaki teknolojik gelişmelerle birlikte bir internet trafiğinin saldırı olup olmadığını tespit etmenin yanı sıra saldırı türünün de tespiti mümkün olmaktadır.

Bu çalışma kapsamında internet kullanıcılarını zararlı içerik içeren web sitelerinden korumak için yapay zekanın bir alt dalı olan topluluk öğrenmesi ile zararlı alan adı tespiti ve sınıflandırması üzerine çalışma yapılmıştır.

1.1. Genel Bakış

Dünya üzerindeki internet kullanım oranı her gün biraz daha artış göstermektedir. Bu kullanım artışı birtakım siber güvenlik risklerini de beraberinde getirmektedir. Yıllar içinde küresel çapta, bireysel kullanıcılara, şirketlere ve hatta ülkelere çok sayıda siber saldırı yapılmış ve çok büyük miktarlarda zararlara yol açılmıştır.

Siber saldırılar çok farklı yöntemlerle yapılabilmekte ve gün geçtikçe yeni saldırı tipleri ortaya çıkmaktadır. Söz konusu saldırı tiplerinin en önemlilerinden biri de zararlı yazılım bulaştırmak, dolandırıcılık vb. amaçlarla kullanıcıları zararlı web sitelerine yönlendirmektir.

Günümüzde alan adı satın almak son derece kolaydır. Siber suçlular satın aldıkları bu alan adları ile oltalama(phishing) saldırıları yapma, zararlı yazılım(malware) indirtme, istenmeyen e-posta(spam) gönderme, kimlik avı web siteleri oluşturma ya da botnet ağı kurma gibi çok sayıda amaç için kullanmaktadırlar [5].

EfficientIP'nin 2022 yılında yayınladığı Küresel DNS Tehdit Raporu'na göre şirketlerin %88'i en az 1 siber saldırıya maruz kaldı. Raporun yayınlanmasından önceki 12 ay içerisinde şirket başına ortalama 7 siber saldırı yapıldı. Saldırıların %51'i oltalama saldırısı kurbanı oldu. Aynı şekilde bu saldırılar neticesinde firmaların %70'i uygulamalarında hizmet kesintisi yaşadı ve %24'ü veri sızdırılmasının kurbanı oldu. Rapora göre her bir saldırının ortalama maliyetinin $942.000 olduğu tespit edildi [6].

Alan adı tabanlı bu saldırılar beraberinde zararlı alan adı tespitine yönelik tedbir alınması gerekliliğini doğurmuştur. Zararlı alan adlarına karşı önceleri bu alan adlarından oluşan kara listeler oluşturularak tedbirler alınmaya çalışılmıştır. Ancak bu tedbir sadece halihazırda zararlı olduğu bilinen web sitelerine karşı güvenlik sağladığı için yeni oluşturulan web sitelerine karşı güvenlik sağlayamamıştır. Ancak makine öğrenmesi yöntemlerinin gelişmesi ile birlikte henüz zararlı olduğu bilinmeyen web sitelerinin tespiti de mümkün olmuştur.

1.1.1. Problemin tanımı

Zararlı alan adları ile yapılan saldırıların kara liste yönteminin yanı sıra makine öğrenmesi yöntemleri ile yapılması daha önce zararlı olduğu bilinmeyen alan adlarının tespiti açısından çok önemlidir. İnternetin olağan akışı içerisinde trafiği sekteye uğratmadan alan

adının zararlı olup olmadığını tespit edebilmek için kullanılacak makine öğrenmesi algoritmalarının hızı da doğruluk ve kesinlik performansları kadar önemlidir. Bu doğrultuda hem doğruluk ve kesinlik açısından yüksek performanslı hem de hızlı çalışan bir algoritma kullanan sistemler tasarlanmalıdır.

1.1.2. Amaç ve hedef

Bu tez kapsamında da makine öğrenmesi yöntemlerinden topluluk öğrenmesi yöntemleri kullanılarak zararlı alan adlarının sınıflandırılması üzerine çalışma yapılmıştır. Çalışma kapsamında başarı oranları kadar sınıflandırma sürelerine de odaklanılmış ve operasyonel anlamda işletilebilir bir sınıflandırma süresinde önemli bir başarı oranı elde edilmesi hedeflenmiştir. Topluluk öğrenmesi algoritmaları arasından XgBoost, Bagging, Adaboost, Random Forest vb. algoritmalar kullanılarak yüksek doğruluk oranı ile sınıflandırma yapabilmek ve algoritmaların çalışma sürelerini kıyaslayarak optimum algoritmayı seçmek amaçlanmıştır. Çalışma kapsamında Canadian Institute for Cybersecurity tarafından sağlanan CIC Bell DNS 2021 veri kümesi kullanılarak alan adlarının oltalama amaçlı alan adı, kötücül yazılım indirtme amaçlı alan adı, istenmeyen e-posta gönderim amaçlı alan adı ve iyi huylu (benign) alan adından oluşan 4 farklı sınıf için operasyonel olarak işletilebilir sürede yüksek doğruluk oranı ile sınıflandırılması hedeflenmiştir.

1.1.3. Ana katkılar

Literatürdeki mevcut yöntemler çok büyük ölçüde başarı oranına odaklanmıştır. Ancak söz konusu başarı oranına ulaşmak için geçen süreye odaklanmamıştır. Oysaki internetin sürekli hızlandığı bir ortamda işlem süresini göz ardı etmemek gerekir. Bu tez kapsamında, topluluk öğrenmesi algoritmaları ve derin öğrenme kullanılarak başarı oranları ve işlem süreleri hesaplanıp kıyaslanmıştır. Çalışma neticesinde işlem süresi ve doğruluk oranı bakımından en etkin performans gösteren model önerilmiştir.

1.1.4. Tez organizasyonu

Tez dört ana bölümden oluşmaktadır. Giriş bölümünde yapay zeka ve siber saldırılar ile ilgili genel bilgiler, makine öğrenmesi ve derin öğrenme ile ilgili birtakım bilgiler ve literatür çalışmaları yer almaktadır. İkinci bölüm olan “Materyal ve Yöntem” bölümünde çalışma kapsamında kullanılan veri kümesi üzerinde yapılan ön işlemelere ve

sınıflandırma için kullanılan yöntemlerin detaylarına yer verilmiştir. Üçüncü bölüm olan "Bulgular ve Tartışma" bölümünde deney ortamı, performans ölçütleri, deney sonuçları ve deney sonuçlarının literatür çalışmaları ile karşılaştırma yapılmıştır. Son bölüm olan "Sonuçlar" bölümündeyse çalışma sonuçları değerlendirilmiştir.

1.2. Yapay Zeka

Tarih boyunca insanoğlunun emeğini gerektiren işler otomatize edilmeye çalışılmıştır. Bu süreçte bedensel iş yükleri kadar mental iş yüklerinin de otomatize edilmesi hedeflenmiştir. İnsanoğlunu mental iş yüklerinden kurtarma çabası durmaksızın devam etmiştir [7]. Bu süreçte Antik Yunan'da Aristo'nun (MÖ 384-322) ilk tümdengelimli akıl yürütme sistemini geliştirmesinden günümüzdeki kuantum bilgisayarlara kadar sayısız gelişme yaşanmıştır. Bu gelişmelerin önemli parçalarından biri de yapay zekadır. Yapay zeka, insan zekasının yapabileceği görevleri yerine getirebilmek için bilgisayar sistemi geliştirmeyi amaçlayan bir bilim dalıdır.

Alan Turing, 1950 yılında yayınladığı "Computing Machinery and Intelligence" [8] isimli makalesinde makinelerin düşünebilip düşünemeyeceğini sorgulamış ve bunu değerlendirebilmek için literatüre Turing testi olarak geçen testi önermiştir. Aynı makalesinde artık makine ve düşünme ifadelerinin bir arada kullanılmaya başlamasını önermiştir. O günden bugüne pek çok farklı gelişmenin yaşandığı yapay zeka alanı makine öğrenmesi, doğal dil işleme, görüntü işleme gibi birçok alt dala ayrılmıştır.

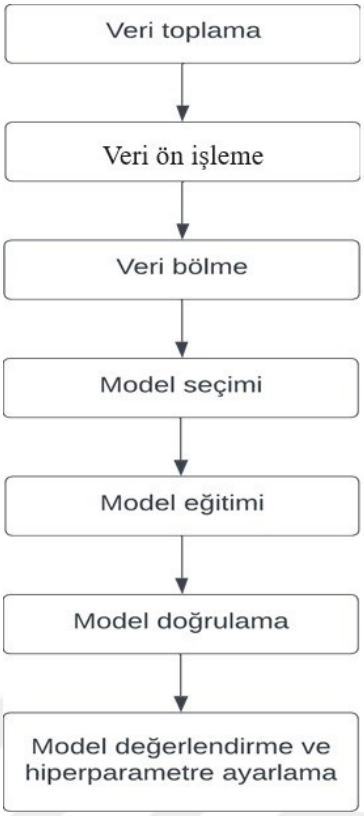
1.2.1. Makine öğrenmesi

İnsanoğlu zamanla bilgisayarları programlayarak ön tanımlı bilgilerle çalışan ve girdi-çıktı çiftleri arasındaki ilişkiler doğrultusundaki belirli kurallara göre işletilen bilgisayar programları geliştirmeyi başarmıştır. Bu gelişme pek çok soruna çözüm olup onlarca kullanım alanında bilgisayarları insan hayatının önemli bir parçası haline getirmiştir. Ancak buna rağmen bazı görevlerde girdi ve çıktı çiftleri arasındaki ilişkiler çok net olmayabilir ya da büyük ölçekli veri kümeleri göz önünde bulundurulduğunda veriler arasında bazı önemli bağlar saklı olabilir. Büyük ölçekli veriler söz konusu olduğunda bu veriler arasındaki ilişkilerin kodlanması çok uzun zaman alabilir. Ayrıca zaman içerisinde yeni verilerle programın güncellenmesi gerekebilir [9]. Bu gibi gereklilikler yapay zekanın bir alt dalı olan makine öğrenmesi dalının doğuşuna zemin hazırlamıştır. Makine öğrenmesi, verilen bilgilerle otomatik bir şekilde deneyim kazanabilen bilgisayar

programı geliştirmekle ilgilenir [10].

Makine öğrenmesi bilgisayar programlarının insan dahli olmadan deneyim kazanarak belirli görevleri yerine getirebilmesine imkan tanır. Bu görevler kimi zaman birtakım sınıflandırma problemleri olabileceği gibi ses tanıma, görüntü tanıma, doğal dil işleme, siber saldırı tespiti, kişiselleştirilmiş reklam yelpazesi oluşturma da olabilir.

Makine öğrenmesi çalışma mantığı temel olarak Şekil 1.1'de de görülebileceği gibi veri toplama, verileri ön işleme, veri bölme, model seçimi, model eğitimi, model doğrulaması, model değerlendirmesi ve hiper parametre ayarlarının yapılması adımlarından oluşur. Veri toplama adımı görev kapsamında ihtiyaç duyulan modelin oluşturulabilmesi için uygun verileri elde etme ve bu verilerden öznitelikler çıkarma işlemlerini kapsar. İhtiyaç duyulması durumunda öznitelik mühendisliği yapılarak öğrenme sürecinde kullanılacak özniteliklerin en uygun olanları seçilir ve gerekiyorsa öznitelik türetilir. Sonrasında toplanan veriler arasında bozuk ya da eksik veriler varsa veri kümesini bu verilerden kurtarmak için ön işleme yapılır. Ön işleme kapsamında bozuk ve eksik verilerden kurtulmanın yanı sıra verileri modelde kullanılabilir şekilde anlamlı hale getirme işlemleri de uygulanır. Bu kapsamda sayısallaştırma, kategorik kodlama ya da veri normalizasyonu gibi teknikler uygulanabilir. Veri ön işleme sonrasında veriler eğitim ve test işlemleri için bölünür. Bu adımın ardından hedeflenen görevi yerine getirmeye uygun bir model seçilir. Model seçiminde görevin hedefleri, verilerin tipi ve veri kümesinin büyüklüğü gibi parametreler de göz önünde bulundurulur. Model seçiminden sonra eğitim verileri ile ilgili model eğitilir ve deneyim kazandırılır. Bir sonraki aşamada test verileri ile model test edilir ve performansı değerlendirilir. Gerekli görüldüğü durumlarda modelin tipine göre hiper parametrelerde güncellemeler yapılarak modelin performansı artırılabilir.



Şekil 1.1. Makine öğrenmesi akış diyagramı.

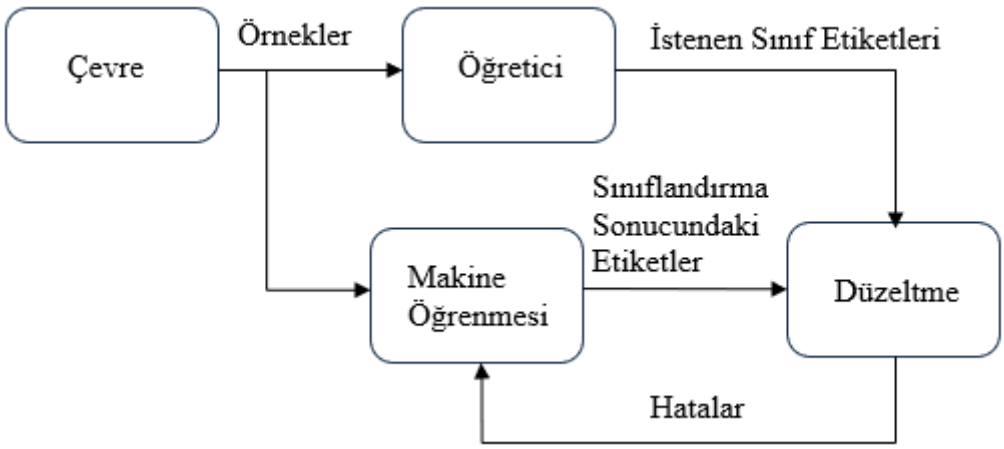
Görevlerin çeşitliliği farklı öğrenme tiplerinin oluşumuna yol açmıştır. Bu farklılıklar neticesinde denetimli öğrenme, denetimsiz öğrenme, yarı denetimli öğrenme, pekiştirmeli öğrenme ve topluluk öğrenmesi gibi farklı öğrenme türleri ortaya çıkmıştır.

1.2.1.1. Denetimli öğrenme

Veri kümesindeki her bir örneğin hangi sınıfa ait olduğunu ifade edecek şekilde etiketli olduğu makine öğrenmesi tekniğidir. Denetimli öğrenmede temel amaç veri kümesindeki bir kayda ait özniteliklerle o kaydın etiketi arasındaki ilişkiyi modelleyebilecek bir fonksiyon oluşturabilmektir [11].

Şekil 1.2'de görülebileceği gibi denetimli öğrenmede iki tip çıktıdan bahsetmek mümkündür. Bunlardan biri her bir eğitim örneğine karşılık gelen etiketin ifade ettiği çıktıdır. Diğeri ise öğrenme algoritması sonucunda hesaplanan çıktıdır. Denetimli öğrenmede hesaplanan çıktı ile hedef çıktı arasındaki farkı en aza indirgemek için öğrenme modeli oluşturulur. Modelin oluşumu esnasında parametreler optimize edilerek en iyi performansa ulaşması hedeflenir. Denetimli öğrenme çalışma prensibine göre kullanılan algoritmada ilk olarak ilgili parametreler rassal olarak belirlenir. Eğitim

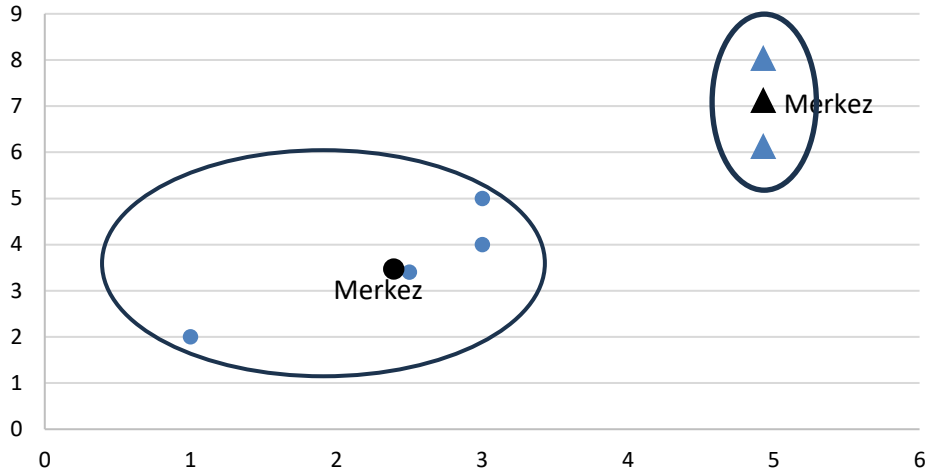
aşamasında her bir iterasyonda eğitim örneği için çıktı hesaplanır ve hata oranının iyileştirilmesi için parametreler güncellenir. Bu süreç hatayı minimize edene ya da parametrelerin yakınsamasına ulaşana dek devam eder. Rassal Orman (Random Forest) algoritması, Karar Ağaçları (Decision Trees), Adaboost ve XGBoost algoritması denetimli öğrenme algoritmalarına örnek olarak verilebilir. Denetimli öğrenme sınıflandırma ya da regresyon problemlerinde kullanılır [12].



Şekil 1.2. Denetimli öğrenme [12].

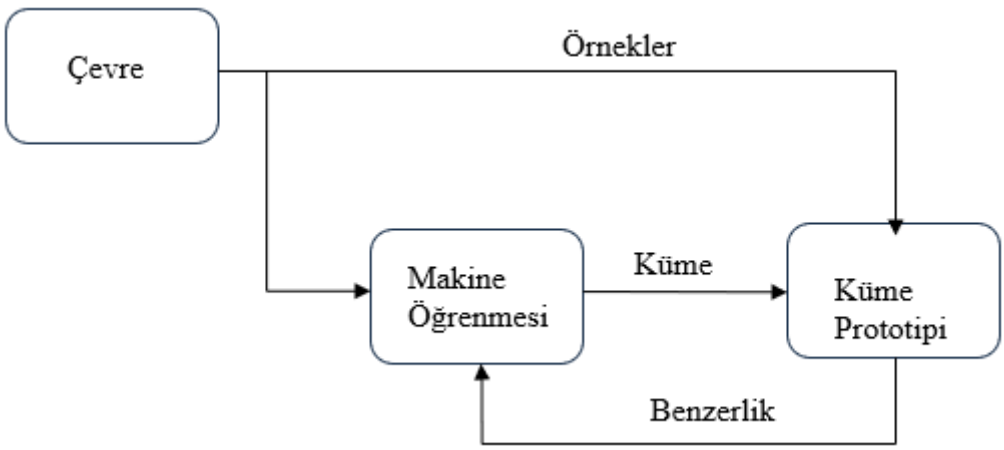
1.2.1.2. Denetimsiz öğrenme

Denetimsiz öğrenme modelinde öznitelikler ve veri deseni etiketlenmez. Bunun yerine bilgisayar programının veriler arasındaki gizli desenleri tespit edip etiketlemesi beklenir. Bu amaçla en yaygın kullanılan algoritmalardan biri Şekil 1.3'teki gibi kümeleme yapan k-ortalama algoritmasıdır [13].



Şekil 1.3. k-ortalama algoritması ile veri kümeleme [13].

Veri örneklerini kümeleme denetimsiz öğrenmenin görevidir. Kümeleme işlemi yapılırken ilk olarak rassal şekilde bir kümeleme prototipi belirlenir. Sonrasında modelin eğitimi sırasında eğitim verileri arasındaki benzerlik göz önünde bulundurularak kümeleme parametreleri optimize edilir. Şekil 1.4.'te de görülebileceği üzere denetimsiz öğrenme için öncelikle kümelerin sayısı belirlenip kümeleme prototipleri oluşturulur. Eğitim veri kümesindeki her bir örneğin, söz konusu kümelerden hangisine daha yakın olduğu hesaplanır. Kümeleme prototiplerindeki parametre güncellemesi, eğitim verileri ile eğitilerek kümeleme parametreleri yakınsayana dek eğitilir.

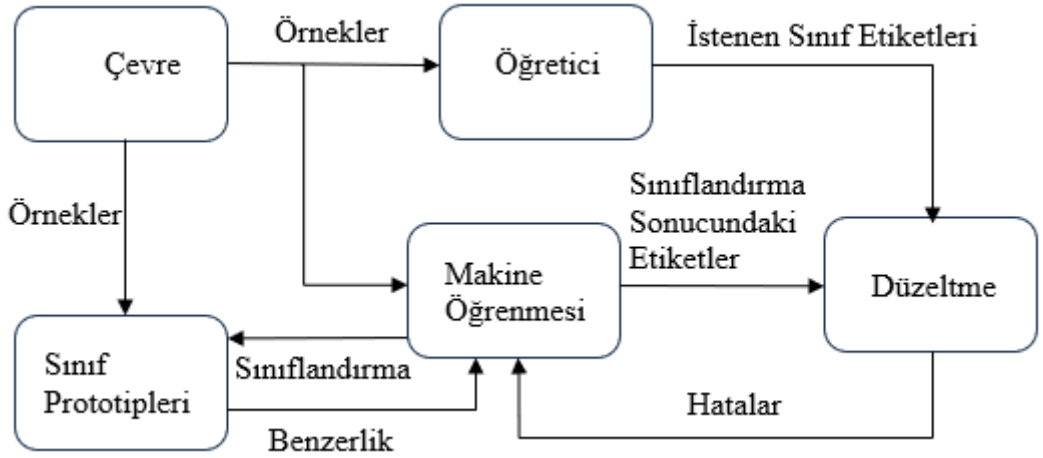


Şekil 1.4. Denetimsiz öğrenme [12].

1.2.1.3. Yarı denetimli öğrenme

Yarı denetimli öğrenme etiketlenmiş ve etiketlenmemiş verilerden faydalanarak oluşturulan bir makine öğrenmesi yaklaşımıdır. Etiketli verileri elde etmek oldukça maliyetli bir süreçtir. Ancak etiketsiz verileri elde etmek çok daha kolaydır. Bu sebeple güçlü bir model geliştirmek için ihtiyaç duyulan tüm verileri etiketli verilerden almak yerine etiketsiz verileri de kullanarak daha güçlü bir model elde edilmesi mümkündür. Yarı denetimli öğrenme yaklaşımında model denetimli öğrenmenin aksine sadece etiketli verilerle eğitilmez. Öğrenme aşamasında etiketli verilerin yanı sıra etiketsiz verilerdeki benzerlik oranından da faydalanılır [12].

Yarı denetimli öğrenmenin temel kullanım alanı denetimli öğrenmede olduğu gibi sınıflandırma ve regresyon problemleridir. Şekil 1.5'teki akış diyagramında da görülebileceği gibi yarı denetimli öğrenme modelinde öncelikle veri kümesi verilerin etiketli olup olmaması durumuna göre ikiye ayrılır. Sonrasında etiketsiz veriler, benzerlikler doğrultusunda yapay olarak etiketli hale getirilir. Eğitim sürecinde makine öğrenmesi algoritmasının parametreleri hedef çıktılar ve hesaplanan çıktılar göz önünde bulundurularak optimize edilir. Bu aşamada yarı denetimli öğrenmede denetimli öğrenmeden farklı olarak parametre optimizasyonu yapılırken etiketsiz verilerin küme prototipleri arasındaki mesafenin minimize edilmesi de göz önünde bulundurulur.

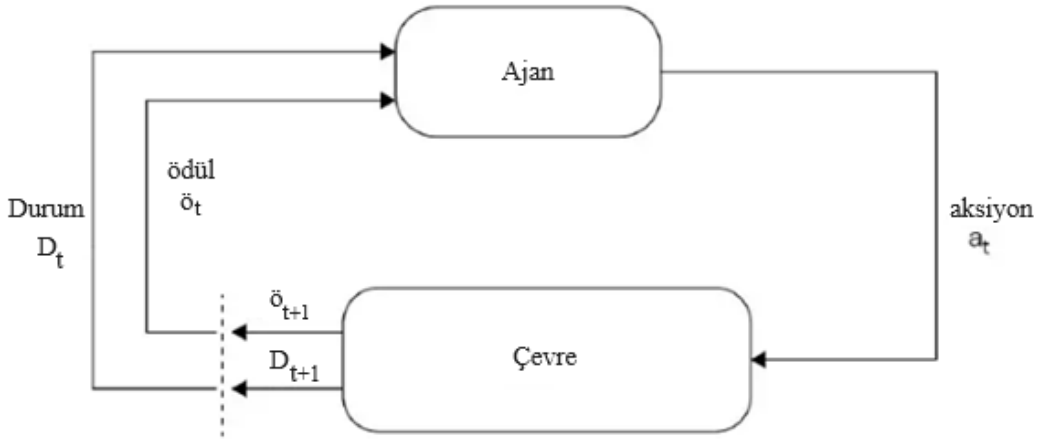


Şekil 1.5. Yarı denetimli öğrenme akış diyagramı [12].

1.2.1.4. Pekiştirmeli öğrenme

İlk kez 1950 yılında Richard Bellman tarafından dinamik programlama ve yarı lineer denklemler kapsamında çalışılan [14] pekiştirmeli öğrenme, sıralı karar vermeyi

gerektiren problemlerin çözümünde kullanılır. Günlük hayatta video oyunu oynama, robotik kontrol ve araç sürme bu kapsamdaki problemler arasında değerlendirilebilir. Ardışık karar vermeyi gerektiren problemler çözülürken hedefe ulaşmak için bir karar doğrultusunda aksiyon alınır. Aksiyon neticesinde hedefe ne kadar yaklaşıldığı bilgisi ile geri besleme yapılarak ajan tarafından sonraki aksiyon belirlenir. Her aksiyonun neticesinde çevrede değişim olabilir. Bu sebeple geri bildirimle birlikte çevredeki değişiklikler de gözlenir. Pekiştirmeli öğrenmenin en temel yapı taşları ajan ve çevredir. Çevre sistemin durumunu ifade eden bilgiyi üretir. Ajan ise sistemin durumunu değerlendirerek çevre ile etkileşime girer ve yeni aksiyonu belirler. Çevre yeni aksiyonu ve yeni durumu kabul eder. Sonrasında bir sonraki durumu ve ödülü ajana gönderir. Bu şekilde Şekil 1.6'daki gibi ilk döngü tamamlanır. Bu döngü problem çözülene dek devam eder [15].



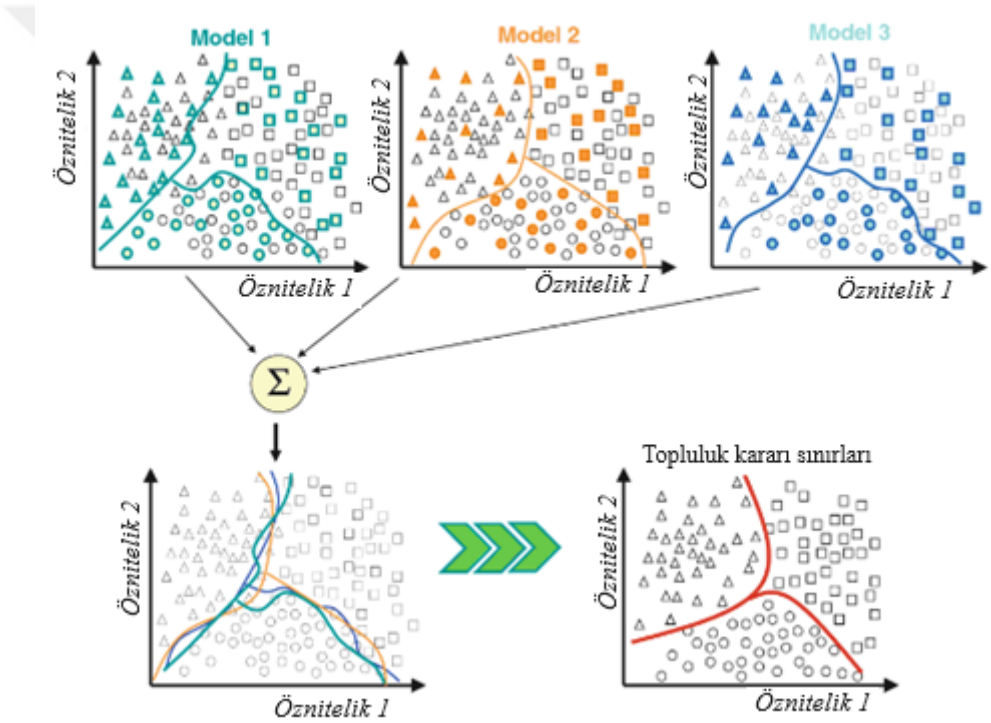
Şekil 1.6. Pekiştirmeli öğrenmede kontrol döngüsü [16].

1.2.1.5. Topluluk öğrenmesi

Topluluk öğrenmesi algoritması insanın günlük hayatta pek çok alanda da kullandığı bir algoritmadır. Pek çok insan günlük hayatta bir konuda karar verirken farklı insanlara danışıp çoğunluğun kararını doğru kabul edip sorununu çözmeye çalışır ve kendi kararını diğer insanların kararlarını dikkate alarak verir. Topluluk öğrenmesinin makine öğrenmesi dünyasındaki kullanımı da bundan çok farklı değildir.

Topluluk öğrenmesi birden fazla model veya tahminciyi birlikte kullanarak daha iyi performans elde etmeyi amaçlayan bir makine öğrenmesi algoritmasıdır [17].

Son yıllarda çok geniş bir alandaki başarılı performansıyla dikkatleri üzerine çekmeyi başarmış olan topluluk öğrenmesi algoritması ilk olarak varyansı azaltmak için ortaya çıksa da zaman içerisinde öznitelik seçimi, hata düzeltimi ve dengesiz veri kümesi problemleri gibi pek çok alanda doğruluk oranını iyileştirmek için kullanılmıştır. Sınıflandırma hataları genel olarak sapma (bias) ve değişkenlik (variance) kavramlarından oluşur ve bu iki kavram çoğu durumda birbiri ile ödünleşim halindedir. Yani yüksek sapma beraberinde düşük değişkenlik getirir. Topluluk öğrenmesi yaklaşımı ile sapmanın kontrol altında tutularak ortalama alma yöntemiyle de değişkenliğin azaltılması amaçlanmaktadır [18]. Şekil 1.7'de topluluk öğrenmesi yöntemleri ile değişkenliğin azaltılması gösterilmiştir.



Şekil 1.7. Topluluk öğrenmesi yöntemleriyle değişkenliğin azaltılması [18].

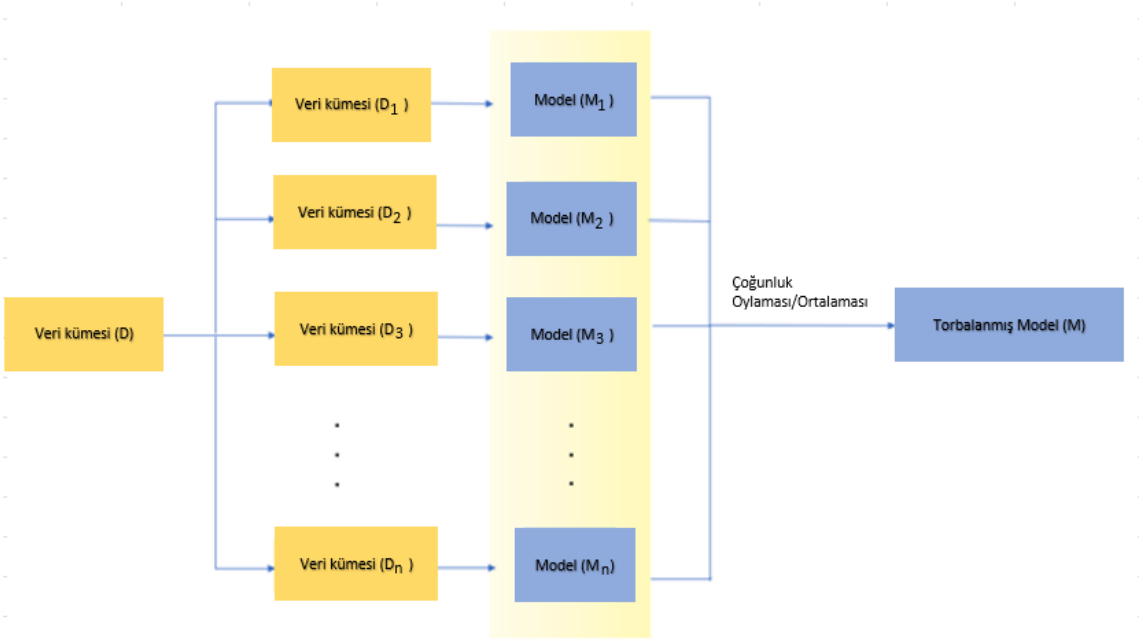
2018 yılında Olson [19] ve arkadaşları 14 popüler makine öğrenmesi algoritmasını 165 benchmark veri kümesi üzerinde test ederek kıyaslamıştır. Çalışma neticesinde algoritmaların bu veri kümelerinin kaçında birbirlerine üstünlük kurduğu ortaya çıkarılmıştır. Çalışma neticesinde Tablo 1.1'de görüldüğü gibi topluluk öğrenmesi algoritmaları açık şekilde diğer algoritmalara göre daha iyi performans göstermiştir.

Tablo 1.1. 165 veri kümesinde 14 algoritmanın üstünlük karşılaştırması [17].

Kazanma \ Kaybetme	XGBoost	Gradient Boosting	Extra Trees	Random Forest	Kernel SVM	Decision Tree	K-Nearest Neighbor	AdaBoost	Logistic Regression	Linear SVM	Passive Aggressive	Bernoulli Naive Bayes	Gaussian Naive Bayes	Multinominal Naive Bayes
XGBoost-	0	34	51	48	74	124	130	132	128	129	139	151	158	157
Gradient Boosting-	12	0	40	37	67	116	115	129	122	124	137	149	159	156
Extra Trees-	22	28	0	27	59	107	128	116	116	121	133	146	159	157
Random Forest-	16	20	28	0	59	105	120	118	118	122	133	143	158	154
Kernel SVM-	21	24	27	34	0	83	111	101	102	111	128	138	154	151
Decision Tree-	1	2	4	0	26	0	72	81	85	87	99	119	137	137
K-Nearest Neighbor-	4	9	3	4	8	49	0	72	70	72	90	119	140	137
AdaBoost-	0	3	11	9	15	35	55	0	58	60	73	98	131	126
Logistic Regression-	6	7	10	10	6	42	55	58	0	25	82	93	132	139
Linear SVM-	5	6	8	9	3	38	47	52	6	0	68	88	132	137
Passive Aggressive	1	3	5	9	1	35	40	51	15	19	0	85	129	124
Bernoulli Naive Bayes-	0	1	1	1	5	17	29	27	24	28	38	0	99	107
Gaussian Naive Bayes-	1	0	0	1	5	15	7	16	14	14	18	39	0	77
Multinominal Naive Bayes-	2	2	2	2	1	10	10	22	1	3	8	25	66	0

1.2.2. Bagging algoritması

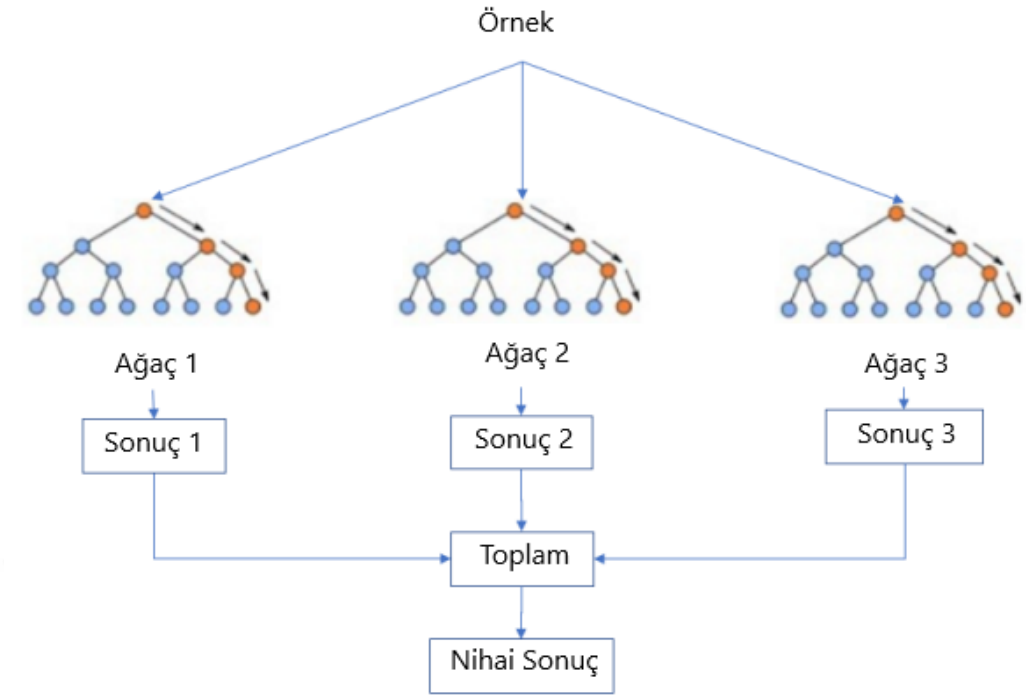
Bagging Algoritması 1996 yılında Leo Briemean tarafından tanıtılmış bir topluluk öğrenmesi tekniğidir. Gürültülü veri kümelerinde varyans değerini düşürme yoluyla zayıf modellerin performansını iyileştirmeyi amaçlar [20]. Bagging Algoritması çalışma prensibine göre veri kümesi Şekil 1.8'deki gibi rassal ve tekrarlı bir şekilde alt gruplara ayrılarak çok sayıda alt veri kümesi elde edilir. Alt veri kümelerinin her biri ayrı bir temel tahminleyici ile eğitilir ve alt modeller oluşturulur. Eğitim sonrasında test verileri ile bu alt modeller test edilerek elde edilen sonuçlar çoğunluk oylaması ya da ortalaması hesaplanarak nihai model elde edilir.



Şekil 1.8. Bagging algoritması çalışma prensibi.

1.2.3. Random Forest algoritması

Random Forest algoritması sınıflandırma ve regresyon işlemleri için kullanılabilen bir topluluk öğrenmesi metodudur. Veri kümesindeki verilerin ve özniteliklerin rassal olarak çok sayıda küçük gruplara bölünerek Şekil 1.9'daki gibi farklı karar ağaçları oluşturmasına dayanır. Oluşturulan bu karar ağaçlarının her biri kendi içinde tahminlerde bulunur. Sınıflandırma problemlerinde nihai sınıf her bir karar ağacının yaptığı tahminlerin çoğunluk oylaması ile bulunur [21]. Regresyon problemlerindeyse karar ağaçlarının tahminlerinin ortalaması hesaplanarak nihai sonuç bulunur.



Şekil 1.9. Random Forest algoritması [21].

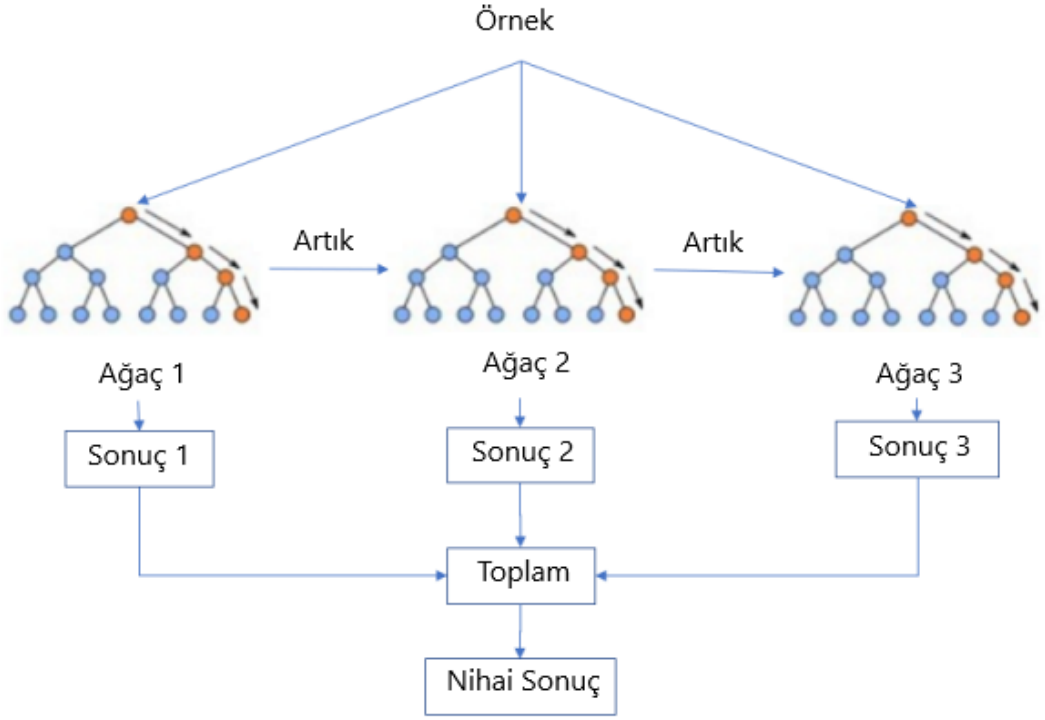
1.2.4. Extra Trees algoritması

Extra Trees algoritması da Random Forest algoritması gibi bir topluluk öğrenmesi algoritmasıdır. Random Forest algoritmasından farkı rassal olarak karar ağaçlarını oluşturma yöntemi ile ilgilidir. Random Forest algoritması eğitim veri kümesini alt gruplara ayırıp karar ağaçları oluşturur ve bunu rassal olarak yapar. Extra Trees algoritması ise eğitim veri kümesinin tamamını kullanarak rassal vektörler seçer ve karar ağaçlarını oluşturur [21].

1.2.5. Extreme Gradient Boosting algoritması

Extreme Gradient Boosting (XGBoost) algoritması sınıflandırma ve regresyon problemlerinde kullanılabilen bir topluluk öğrenmesi metodudur. XGBoost algoritması yüksek düzeyde ölçeklenebilirdir ve diğer popüler algoritmalara göre yaklaşık 10 kat daha hızlıdır [22]. XGBoost algoritmasının diğer önemli katkıları seyreklik farkındalığına sahip olması, etkili bir ön bellek farkındalık blok yapısına sahip olması ve teorik olarak doğrulanabilir ağırlıklandırılmış niceliksel taslak sunmasıdır [22]. Random Forest algoritmasından farklı olarak modeli oluşturan karar ağaçları tamamen rassal şekilde

oluşmaz. Oluşturulan her yeni karar ağacı Şekil 1.10'da görülebileceği gibi kendisinden önceki karar ağacının hatalarını iyileştirmeye odaklanır.

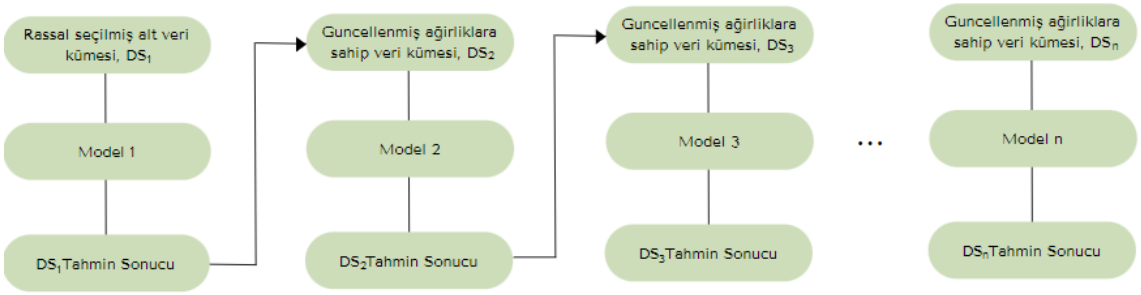


Şekil 1.10. XGBoost algoritması [21].

1.2.6. Adaptive Boosting algoritması

1996 yılında Robert Schapire ve Yoav Freund tarafından önerilen Adaptive Boosting (Adaboost) algoritması bir topluluk öğrenmesi sınıflandırıcısıdır [23]. Daha yüksek doğruluk oranına sahip bir sınıflandırıcı elde edebilmek için çok sayıda daha zayıf sınıflandırıcının yinelemeli bir yaklaşımla birleştirilmesi prensibine dayanır. Adaboost sınıflandırıcısının diğer boosting sınıflandırıcılardan en belirgin farklılığı hem her bir alt sınıflandırıcı için oluşturulacak alt veri kümesinin ağırlığı hem de her bir sınıflandırıcının ağırlığının her bir yinelemede belirlenmesidir. Ağırlıkların farklı belirlenmesinin sebebi bir önceki sınıflandırıcıda yapılan hatalar doğrultusunda daha yüksek doğruluk sonucu veren yeni bir sınıflandırıcı oluşturabilmektir. Adaboost algoritmasının çalışma prensibi Şekil 1.11'de görülebileceği gibi ilk olarak veri kümesinden rassal bir alt veri kümesi seçimi ile başlar. Seçilen bu alt veri kümesi uygun bir modelle eğitilir ve ilk alt veri kümesi için bir sınıflandırıcı oluşturulur. Sınıflandırma sonucuna göre bir sonraki yinelemede seçilecek alt veri kümesindeki sınıfların ağırlığı belirlenir. İlk sınıflandırmada

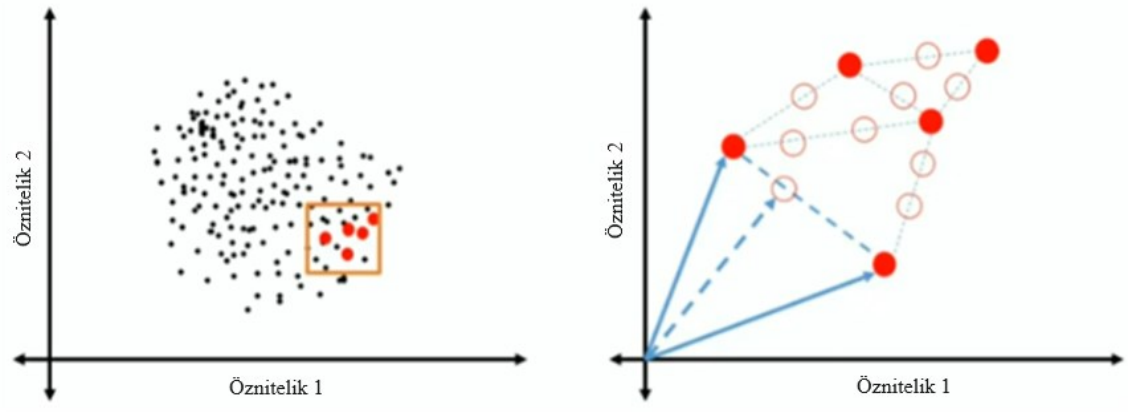
hatalı sınıflandırılan sınıfların ağırlığı yeni veri kümesinde daha yüksek olacak şekilde belirlenir. Bu şekilde bir sonraki yinelemede hatalı sınıflandırılan sınıfın veri kümesindeki oranı artırılarak daha iyi öğrenilmesi hedeflenir. Ayrıca her bir sınıflandırıcının ağırlığı da doğruluk oranı ile doğru orantılı olacak şekilde belirlenir. Daha yüksek doğruluk oranına sahip sınıflandırıcıya daha yüksek oranda ağırlık atanır. Bu yaklaşım, model oluşturulurken belirlenen yineleme sayısınca devam eder. Çok sayıda zayıf sınıflandırıcının yinelemeli bir yaklaşımla bir araya getirilmesiyle daha yüksek doğruluk oranına sahip yeni bir sınıflandırıcı elde edilmiş olur.



Şekil 1.11. Adaboost algoritması çalışma prensibi.

1.2.7. SMOTE

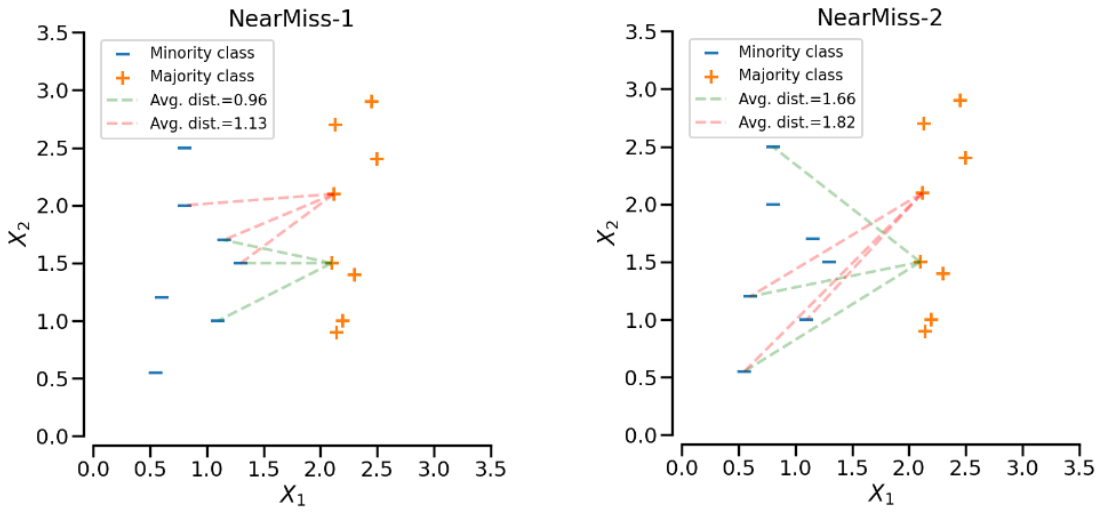
SMOTE sınıflar arası dengesizlik bulunan veri kümelerindeki dengesizliği gidermek amacıyla azınlık durumunda olan sınıflar için yapay olarak yeni veriler üreten bir yukarı örnekleme (oversampling) yöntemidir [24]. SMOTE, yukarı örnekleme işlemini yaparken azınlık durumundaki sınıfa ait her bir örnek için en yakın k komşuluğundaki örneği tespit eder. Daha sonra Şekil 1.12'deki gibi referans aldığı örnekle k komşuluğundaki örneğin arasında sentetik örnekler oluşturur [25]. Sentetik örnekler oluşturulurken referans örnekle en yakın komşuluğundaki örnek arasındaki mesafe hesaplanır ve 0 ile 1 arasındaki rassal bir değerle mesafe çarpılır [26].

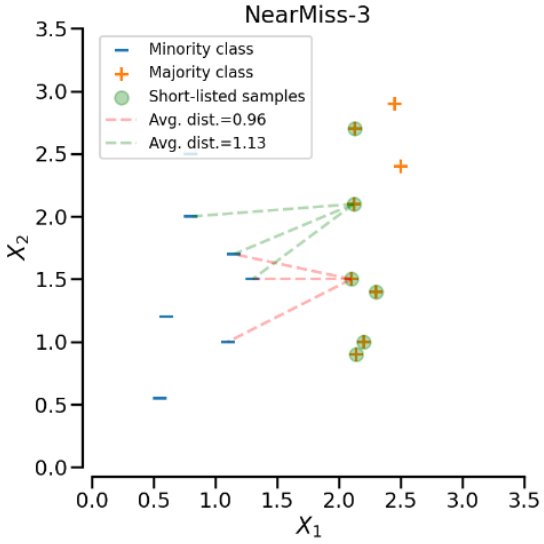


Şekil 1.12. SMOTE [27].

1.2.8. NearMiss

NearMiss, sınıflandırma problemlerinde veri dengesizliklerini aşağı örnekleme (undersampling) işlemi yaparak gidermek gerektiğinde kullanılan bir veri örnekleme metodudur. NearMiss yönteminin NearMiss-1, NearMiss-2 ve NearMiss-3 adında 3 farklı çeşidi vardır. Bu 3 çeşidin örnek azaltma prensibi Şekil 1.13'te görülmektedir. NearMiss-1 yönteminde azınlık durumundaki sınıfa ait en yakın N örneğe olan ortalama mesafelerin en küçük olduğu çoğunluk durumundaki sınıfın örnekleri seçilir. NearMiss-2 yönteminde azınlık durumundaki sınıfın en uzak N örneğine olan ortalama mesafelerin en küçük olduğu çoğunluk durumundaki sınıfın örnekleri seçilir. NearMiss-3 yönteminde ise her bir azınlık sınıfının örneği için en yakın M komşuluğundaki örnekler belirlenir. Daha sonra bu örneklere çoğunluk durumdaki sınıfa ait örnekler arasından en yakın örneklere olan ortalama mesafeler hesaplanarak en uzak mesafeye sahip olan çoğunluk örneği seçilir [28].



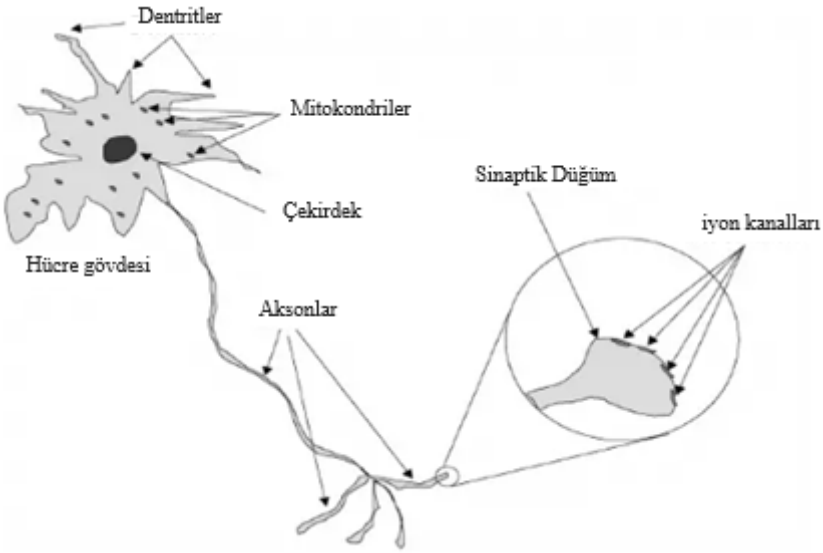


Şekil 1.13. NearMiss tekniği çeşitleri [28].

1.2.9. Yapay sinir ağları

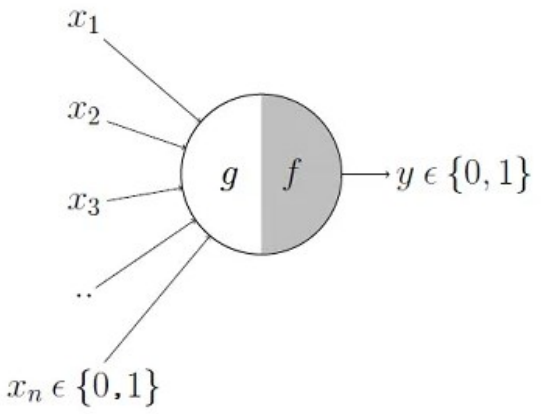
Yapay sinir ağı, insan bilincinin matematiksel olarak modellenebilmesinin genelleştirilmesi olarak geliştirilmiş bir bilgi işlem sistemidir. İnsan beynindeki nöronlardan esinlenerek oluşturulmuştur. Yapay sinir ağının temel fonksiyonu girdi olarak sunulan verilerden çıktı modelini oluşturmaktır. Bir başka deyişle yapay sinir ağı, öğrenme sürecinde elde ettiği verileri depolayan ve uyarlanabilir ağırlıklarla birbirine bağlanan düğümlerden oluşan ağların incelenmesidir [29].

Yapay sinir ağı modeli Şekil 1.14'te de resmedilen insan beynindeki nöronun yapısını örnek alarak modellenir. İnsan beyni çevresel etmenlerin etkisiyle uyarıldığında nöron sodyum ve potasyumu içeri ya da dışarı pompalayarak nöronun elektriksel potansiyelini değiştirir. Eşik değerine bağlı olarak bu potansiyel fark belirli seviyeye ulaşınca ateşleme gerçekleşir ve bir sinir hücresinin aksonundan diğer sinir hücresinin dendritine doğru iletim gerçekleşir. Nöronun ateşlenme hızı voltaj artışına neden olan uyaranın miktarına bağlıdır. Uyaran miktarı düşükse ateşleme hızı yavaş, yüksekse ateşleme daha hızlı olur. İnsan beynindeki bir nöron diğer nöronlardan girdi alır ve diğer sinir hücrelerine çıktı olarak iletir [30].



Şekil 1.14. İnsan beynindeki nöronun yapısı [30].

1943 yılında Warren McCulloch ve Walter Pitts tarafından geliştirilen McCulloch-Pitts (MP) nöron modeli yapay sinir ağları araştırmalarının öncüsü niteliğindedir [7]. Şekil 1.15'te görüldüğü gibi g aldığı inputları toplar ve f nihai kararı verir.



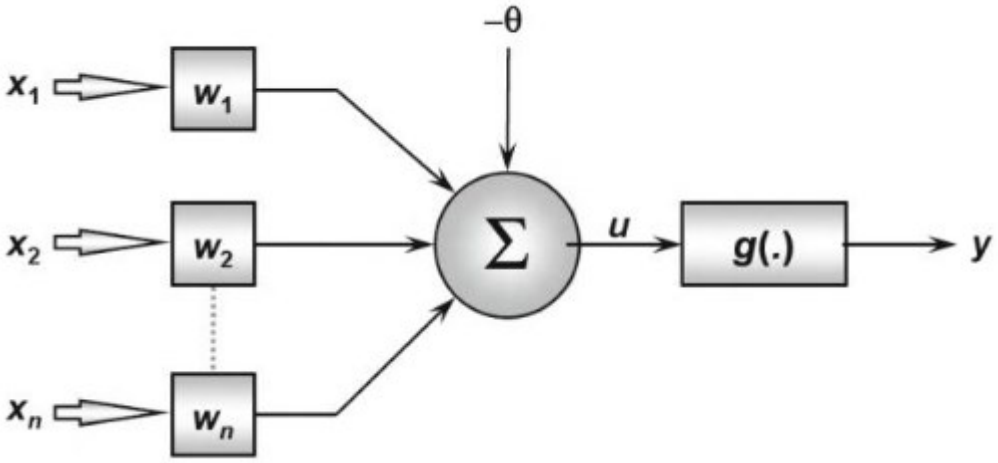
Şekil 1.15. MP Nöron [31].

Nörona gönderilen girdilerin bazılarının nöronun iletimi sağlaması için gereken eşik değeri belirleme üzerinde etkisi daha fazla olabilir. Yani tüm girdilerin etkisinin aynı olmaması mümkündür. Bir başka deyişle her bir girdinin ağırlığı farklı olabilir. Bu doğrultuda nöronu modelleyen denklem, denklem 1.1'deki gibi girdi ağırlığına da bağlıdır.

$$z = f\left(\sum_{i=0}^{n} w_i x_i\right) \tag{1.1}$$

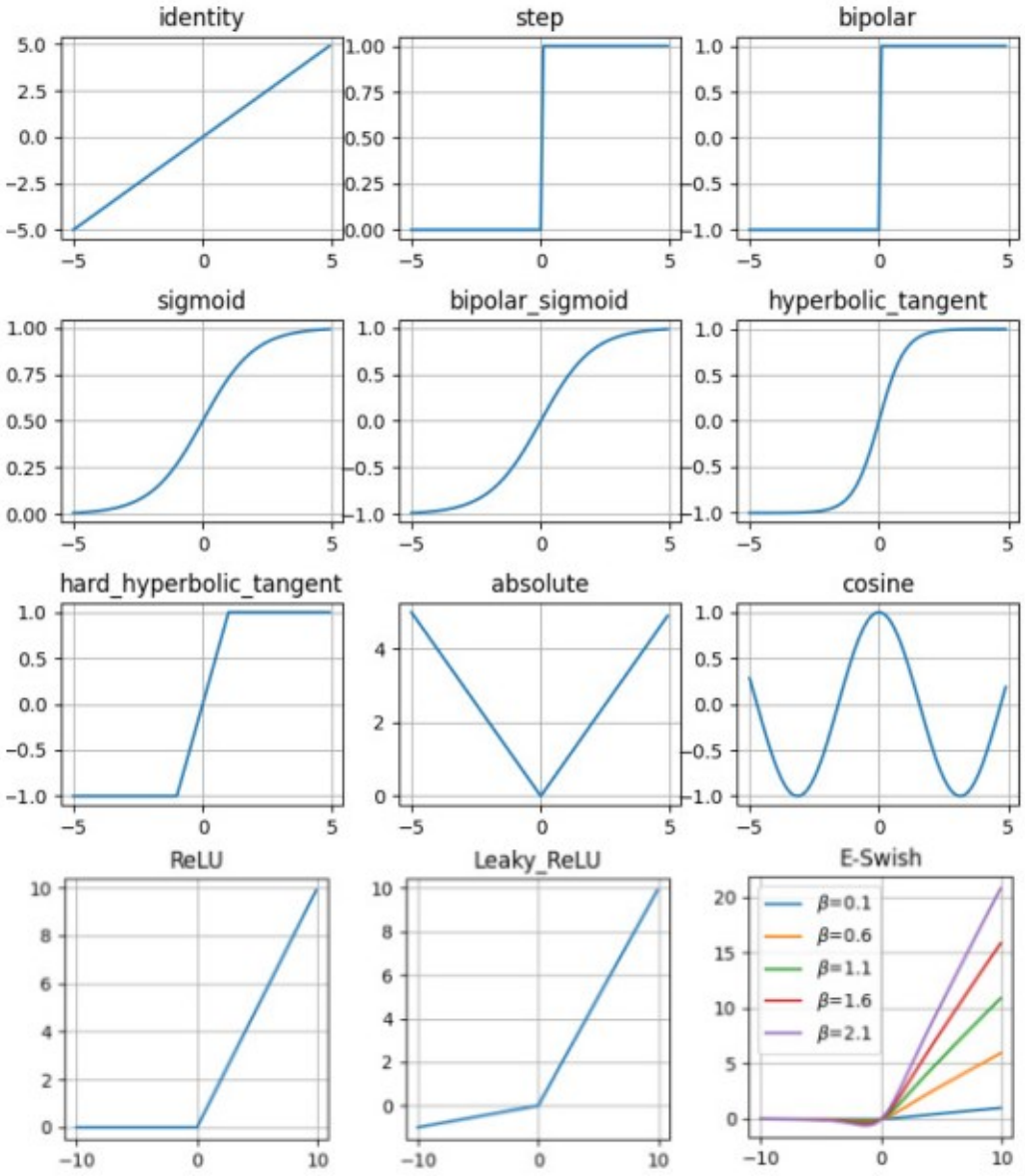
İnsan vücudundaki nöronlar arasındaki uyarı iletimi, nöronun aksiyon potansiyelini aksondan dendrite doğru diğer nörona iletmesiyle mümkün olmaktadır. Matematiksel olarak bunun modellenmesi ise denklem 1.1' de f olarak ifade edilen aktivasyon fonksiyonu ile mümkündür. Bir nörondan diğer nörona iletim için gerekli aksiyon potansiyeli miktarını ifade etmek için kullanılır [31]. Şekil 1.16'da girdi ağırlıklarının ve aktivasyon fonksiyonunun entegre edildiği nöron modeli görülmektedir. Bu modele göre yapay bir nöron girdi sinyali ($x_1, x_2, x_3, \dots, x_n$), sinaptik ağırlıklar ($w_1, w_2, w_3, \dots, w_n$), doğrusal toplayıcı ($\sum$), aktivasyon eşik değeri ya da bias (θ), aktivasyon potansiyeli (u), aktivasyon fonksiyonu (g) ve çıktı sinyali ile birlikte 7 parçadan oluşmaktadır [32]. Bu modele göre:

- Girdi sinyali, çevreden gelen sinyal ve örnekleri,
- Sinaptik ağırlıklar, her bir girdi sinyalinin ağırlığını,
- Doğrusal toplayıcı, aktivasyon voltajını oluşturmak için ağırlıklarıyla birlikte tüm girdi sinyallerini toplama işlemini,
- Aktivasyon eşiği (bias) değişkeni, çıktı sinyalinin tetiklenmesi için doğrusal toplayıcı tarafından üretilmesi gereken tetikleyici değerin eşik değerini,
- Aktivasyon potansiyeli, doğrusal toplayıcı ve aktivasyon eşiği arasındaki farkı,
- Aktivasyon fonksiyonu, kendi fonksiyonel karakteristiğine bağlı olarak nöron çıktısının belirli sınırlar arasında kalmasını sağlayan fonksiyonu,
- Çıktı sinyali ise bir dizi girdi sinyali verilen nörondan çıkan nihai sinyali temsil eder.



Şekil 1.16. Yapay sinir ağı [32].

Yapay sinir ağlarında eğer her bir nöronun aktivasyonu sadece ağırlıklandırılmış girdi sinyallerinin toplamı olarak ifade edilseydi söz konusu nöron doğrusal bir fonksiyon olmuş olurdu [33]. Bu tip modeller doğrusal olmayan özelliklerin ayrıştırılabilmesini gerektiren problemlerde iyi performans gösteremezler. Gerçek hayattaki pek çok problem doğrusal olarak ayrıştırılamadığı için doğrusal olmayan özelliklerin ayrıştırılabilmesi oldukça önemlidir. Bu sebeple genellikle doğrusal olmayan özelliklerin de dikkate alınabilmesi için modele aktivasyon fonksiyonu eklenir. Böylelikle karmaşık ilişkilerin öğrenilebilmesi mümkün kılınmış olur. Kullanılması gereken aktivasyon fonksiyonu, üzerinde çalışılan göreve ve veri kümesine göre değişiklik gösterebilir. Yaygın kullanılan aktivasyon fonksiyonları Şekil 1.17'de belirtilmiştir.



Şekil 1.17. Yaygın kullanılan aktivasyon fonksiyonları [34].

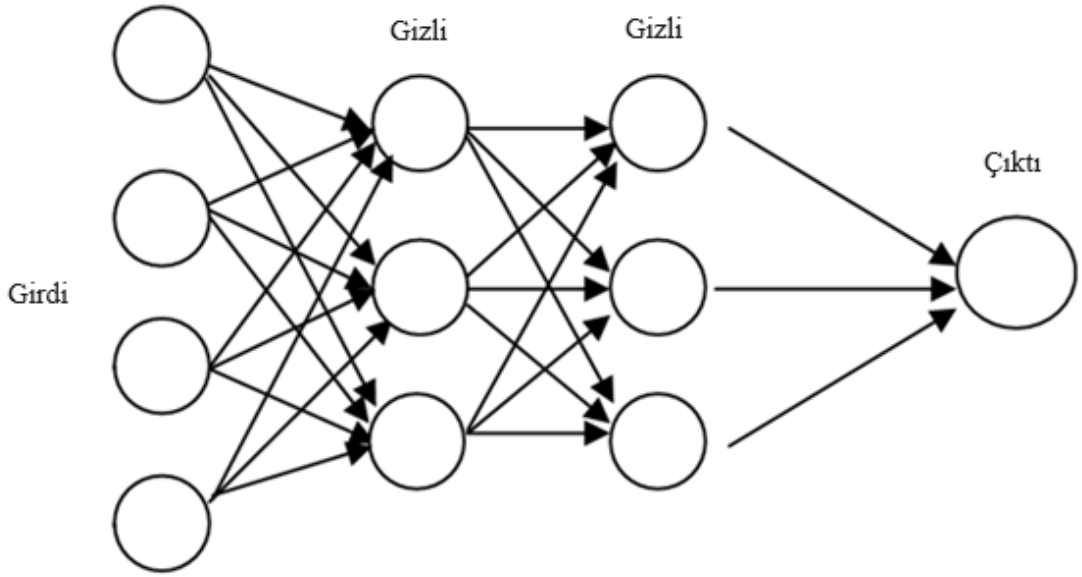
1.2.10. Derin öğrenme

Derin öğrenme, verileri analiz edip işlemek ve öğrenmek için yapay sinir ağlarının geliştirilmesine ve kullanımına odaklanan yapay zeka ve makine öğrenmesinin bir alt dalıdır. Çok sayıda yapay sinir ağı katmanından oluşur ve karmaşık paternleri öğrenerek zor görevleri yerine getirebilir. Derin öğrenme süreci verilerden öğrenme kabiliyeti olan çok sayıda yapay sinir ağından oluşur [35].

Basit makine öğrenme algoritmaları pek çok problemi çözüp görevi tamamlamada başarılı olsa da yapay zeka alanındaki nesne tanıma ya da konuşma tanıma gibi bazı temel

problemlerin çözümünde başarılı olamamıştır. Derin öğrenmenin geliştirilmesini motive eden de aslında bu tarz problemlerde makine öğrenmesi algoritmalarının yeterince başarılı olamayışıdır [36].

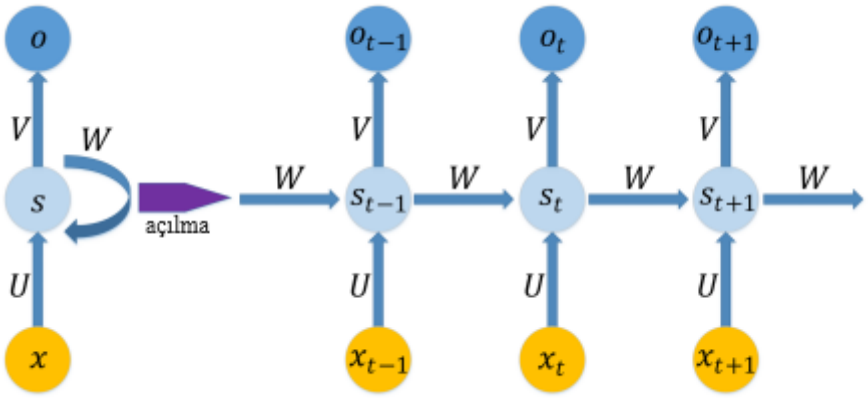
Derin öğrenmenin temeli, Şekil 1.18'de gösterilen ileri beslemeli sinir ağının mimarisine dayanır. İleri beslemeli sinir ağı, perceptron adı verilen çok sayıda katmandan oluşur. Perceptron ilk kez 1958 yılında Fran Rosenblatt tarafından tanıtılan, girdi verilerindeki doğrusal ayrılabilir paternleri öğrenebilen temel bir ikili sınıflandırıcı modelidir.



Şekil 1.18. İki gizli katmandan oluşan bir ileri beslemeli sinir ağı yapısı [37].

1.1.3.1. Yinelemeli sinir ağları (Recurrent Neural Network - RNN)

Yinelemeli sinir ağları dizisel verileri modelleyebilmek için geliştirilmiş bir yapay sinir ağıdır. Geleneksel ileri beslemeli sinir ağından farkı RNN'in sıradaki girişi önceki durumla harmanlayan döngüsel bir yapısı olmasıdır. En temel özelliği karar verirken önceki durum bilgisini kullanabilmesidir. Her bir adımda girdi vektörü ile birlikte önceki durum vektörünü birlikte ele alır ve çıktı olarak çıktı vektörü ile birlikte durum vektörünü verir. Böylelikle mevcut çıktıyı önceki durum vektörünü de hesaba katarak oluşturmuş olur ve dizisel verileri modelleyebilir [38]. Şekil 1.19'de RNN mimarisi görülmektedir.

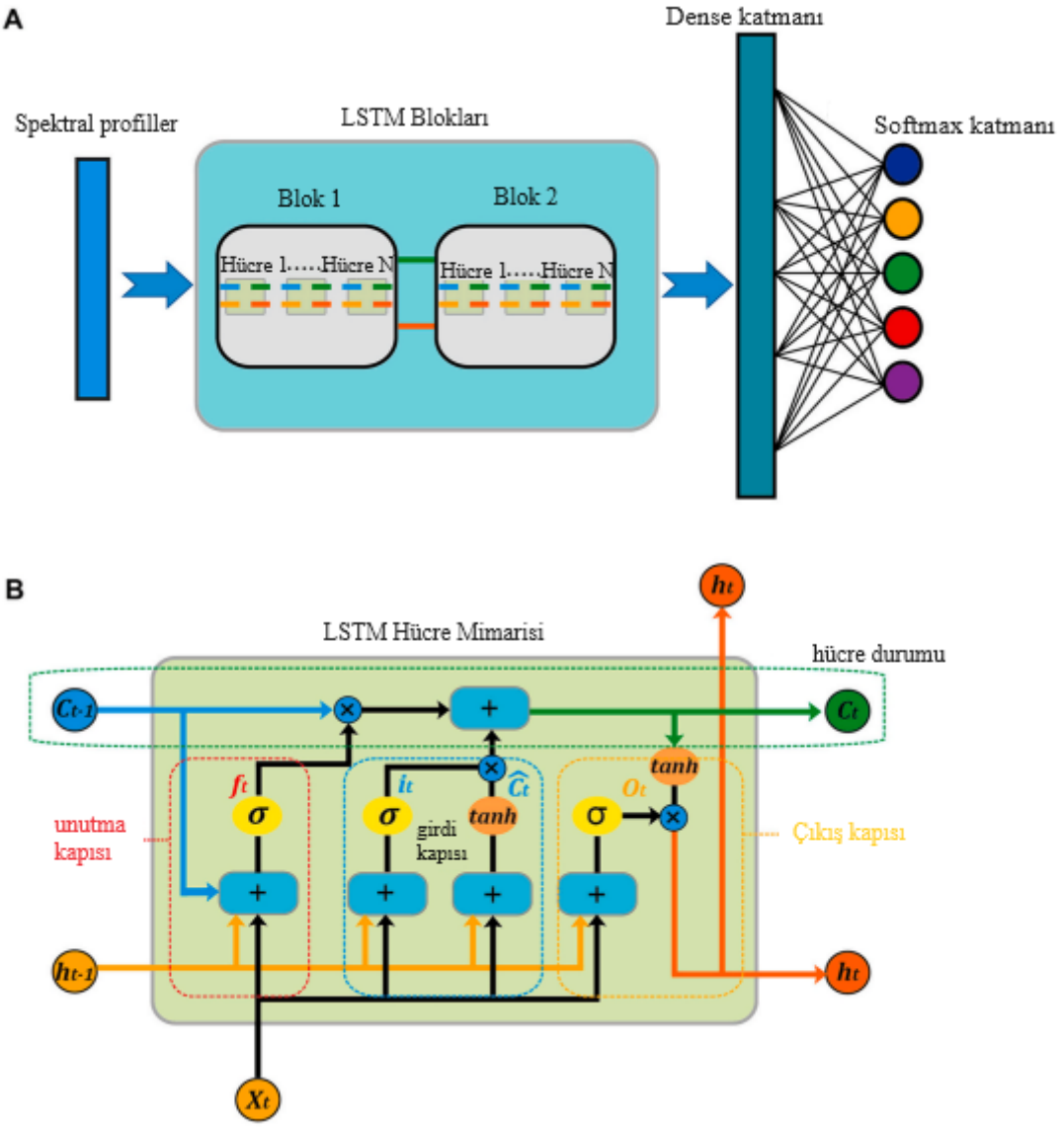


Şekil 1.19. Yinelemeli sinir ağı mimarisi [39].

1.1.3.2. Uzun kısa dönemli bellek (Long Short-Term Memory - LSTM)

LSTM, çok sayıdaki RNN mimarilerinden biridir [40]. RNN her ne kadar zaman serisi problemlerini iyi modelleyebilse de kaybolan gradyan problemi sebebiyle uzun dönemli bağımlılıkları öğrenmekte zorlanır [41]. LSTM bu sorunun üstesinden gelmek için geliştirilmiş bir çözümdür [42].

Kaybolan gradyan sorunu, eğitim sırasında bir ağ üzerinden akan gradyanların zamanda geriye doğru hareket ettikçe hızla azalması ve yavaş eğitim veya yakınsama sorunlarına yol açmasıyla ortaya çıkar. Bu sorunun üstesinden gelmek için LSTM, üç kapıdan (girdi kapısı, unutma kapısı ve çıktı kapısı) oluşan bir bellek hücresi içerir. LSTM, Şekil 1.20'de gösterildiği gibi LSTM hücrelerinin çok sayıda zincir benzeri bir konfigürasyonda birbirine bağlanmasıyla oluşturulur. LSTM hücrelerindeki girdi kapısı, girdi verilerinden hangisinin faydalı olacağına karar verip bir tanh fonksiyonu ile 0-1 aralığında bir çıktı vektörü üretir. Unutma kapısı, bellekteki önceki verilerden hangilerinin unutulup hangilerinin hatırlanacağını belirler. Çıktı kapısı, girdi değerini, önceki gizli durumu ve mevcut bellek durumunu girdi olarak alır ve her girdi öğesinin alakasını belirten ve 0 ile 1 arasında değişen değerlerden oluşan bir vektör üretir [43].



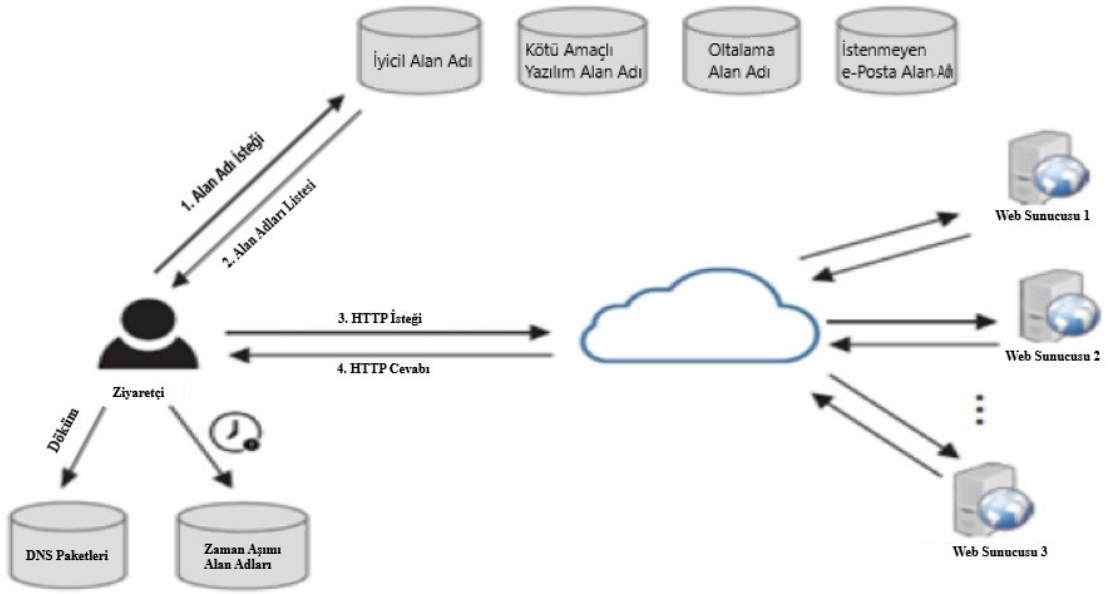
Şekil 1.20. LSTM mimarisi A. mimariye genel bakış, B. LSTM hücre yapısı [44].

1.3. Literatür Çalışmaları

Bu tez kapsamında zararlı alan adı sınıflandırması yapılması amaçlanmıştır. Literatürde bugüne kadar zararlı alan adı sınıflandırma konularında birtakım çalışmalar yapılmıştır. Bu çalışmalar arasında makine öğrenmesi ve derin öğrenme yöntemlerini kullanarak zararlı alan adı sınıflandırması yapan çalışmalar da bulunmaktadır. Ancak topluluk öğrenmesi yöntemlerini kullanarak zararlı alan adı sınıflandırması yapan çalışma pek fazla bulunmamaktadır. Tez kapsamında yapılan çalışmada Adaboost, XGBoost, Random Forest ve Bagging algoritmaları ve LSTM algoritmaları kullanılmıştır. Literatürdeki araştırmalar büyük ölçüde sadece başarı oranlarına odaklanmıştır. Ancak

böyle bir sistemde başarı oranı kadar sınıflandırma için geçen süre de önem arz etmektedir. Bu tez kapsamında Canadian Institute for Cybersecurity tarafından sağlanan CIC Bell DNS 2021 veri kümesi kullanılarak alan adlarının 4 farklı sınıf (phishing, malware, spam, benign) için operasyonel olarak işletilebilir sürede yüksek doğruluk oranı ile sınıflandırılması amaçlanmıştır. Tez kapsamında verilere ön işleme yapılmış ve sonrasında topluluk öğrenmesi yöntemleri kullanılarak ilgili algoritmalar kullanıldığında ne kadar sürede ne kadar başarılı sınıflandırma yapıldığı incelenmiş ve operasyonel anlamda kullanılabilir model tespit edilmiştir.

Mahdavifar ve arkadaşları Şekil 1.21'deki gibi DNS paketlerini alarak kendi oluşturdukları veri kümesini kullanarak farklı oranlardaki eğitim veri kümesi dengesizlik oranına bağlı olarak %94,4 ile %99,4 arasında başarı oranı elde etmişlerdir. Çalışmalarını yaparken öz nitelik çıkarma konusuna eğilmişlerdir. Öznitelikleri sözcük tabanlı, DNS istatistiksel tabanlı ve üçüncü taraf tabanlı (üçüncü tarafların oluşturduğu API'lerden elde edilenler) öz nitelikler olarak nitelemişler ve veri kümesini hazırlarken bu doğrultuda öznitelikleri çıkarmışlardır. Yüksek başarı oranı elde etseler de sınıflandırma için geçen süre üzerine bir çalışma yapmamışlardır ve topluluk öğrenmesi yöntemleri kullanmamışlardır [5].



Şekil 1.21. DNS paketi alma [5].

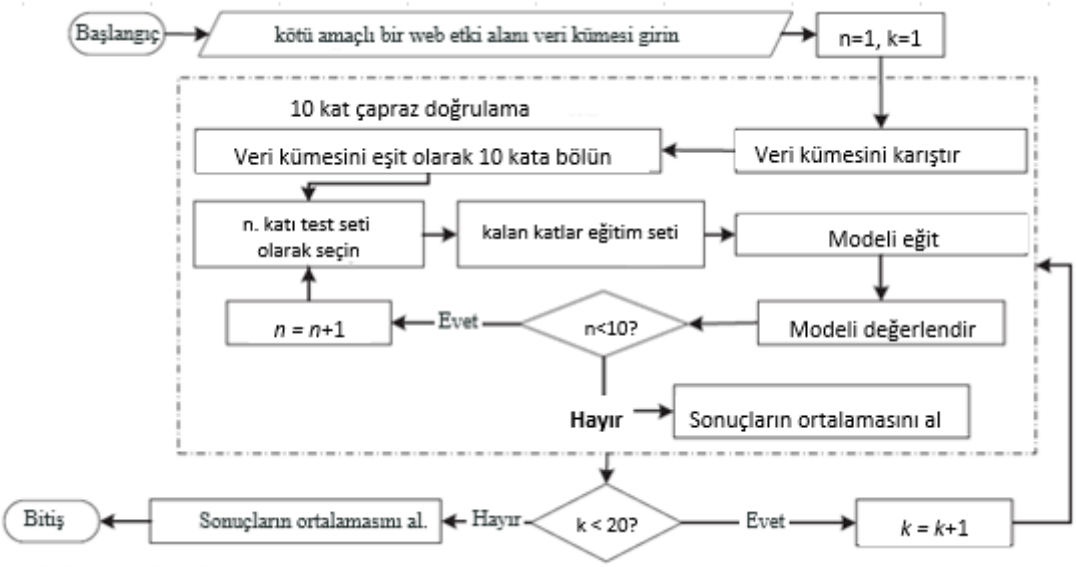
Park ve arkadaşları zararlı alan adı sınıflandırması alanında çalışma yapmak için ihtiyaç

duyulan veri kümelerinin elde edilmesi sürecini kolaylaştırmak için sadece normal alan adlarından oluşan örnekler toplamışlardır ve bu örneklere ait öznitelikleri çıkararak sonrasında bir alan adının kötücül (malicious) olup olmadığını tespit etme yoluna gitmişlerdir. Önerdikleri yöntem kötücül alan adlarının tipinin sınıflandırılmasına dair bir imkan tanımasa da bir alan adının kötücül olup olmadığının tespitine veri kümesi toplama zahmetini azaltmaları açısından katkıda bulunmuşlardır. Nihai olarak CNN ve LSTM derin öğrenme metotlarını kullanarak ikili sınıflandırma yapmışlardır ve %99,4 F1 skoruna ulaşmışlardır. Çalışmaları kapsamında kötücül ve normal örneklerden oluşan veri kümeleri ve sadece normal örneklerden oluşan veri kümeleri ile eğittikleri modellerde aynı algoritmaları kullanarak yaptıkları sınıflandırmada sadece normal örneklerle eğitilen modelde daha başarılı sonuç almışlardır [44].

Yang ve arkadaşları önerdikleri modelde kötücül alan adlarını rastgele karakter tabanlı, sözcük tabanlı, tahmin edilen karakter tabanlı ve çok öğeli olmak üzere dört tipe ayırmıştır. Yaptıkları çalışma neticesinde bu ayrımların başarı oranını pozitif yönde etkilediğini belirtmişlerdir ve derin öğrenme tekniklerinden LSTM ve CNN algoritmalarını kullanarak önerdikleri modelle %94,6 kesinlik oranına ulaşmışlardır [45].

Cucchiarelli ve arkadaşları önerdikleri yaklaşımda alan adlarını, sınıflandırılmamış bir alan adındaki 2-gram ve 3-gram arasındaki benzerliği kötücül ve normal alan adındaki benzerlikle kıyaslayarak ifade eden bir dizi özniteliği kullanarak çalışma yapmışlardır. Farklı derin öğrenme metotlarıyla farklı veri kümelerindeki performansları kıyaslamışlardır ve kendi önerdikleri metotla farklı veri kümelerinde %82 ile % 97,2 arasında değişen oranlarda başarı oranı elde etmişlerdir [46].

Chiong ve arkadaşları Fuzzy-Weighted Least Squares Support Vector Machine (EFW-LS-SVM) adını verdikleri yöntemle farklı oranda normal örnek içeren eğitim setiyle eğitilen modellerdeki başarı oranları üzerine çalışmışlardır. Yaptıkları SMOTE yukarı örnekleme ile başarı oranlarının önemli oranda arttığını saptamışlardır. Çalışmalarında, XGBoost, ANN, SVM, Random Forest algoritmalarını temel almışlardır. Son olarak EFW-LS-SVM modeliyle %94,5 F1 değerini yakalamışlardır [47]. Ancak çalışmaları kapsamında sınıflandırma süresi üzerine herhangi bir çalışma yapmamışlardır. Ayrıca tez kapsamında kullanılacak veri kümelerinden farklı bir veri kümesi kullanmışlardır.



Şekil 1.22. EFW-LS-SVM modeli akış diyagramı [47].

Sun ve arkadaşları etiketsiz verilerden öğrenme ve öznitelik çıkarma üzerine yaptıkları çalışmada %99,19 F1 skoruna ulaşmayı başarmışlardır. Yaptıkları çalışma ile öznitelik çıkarma sürecini 11.53 kat hızlandırdıklarını belirtmektedirler. Öznitelikleri çıkardıktan sonra SVM ve XGBoost algoritmaları ile sınıflandırmışlardır ve önerdikleri yöntemin literatürdeki diğer çalışmalarla benzer başarı oranlarına ulaştığını tespit etmişlerdir. Öznitelikleri hızlıca çıkarmaları yönüyle literatüre katkıda bulunmuşlardır [48].

Palaniappana ve arkadaşları sözcüksel analiz, kara liste kullanımı ve DNS verilerinden elde edilen özniteliklerin yanı sıra web-tabanlı özniteliklerin de zararlı alan adı tespitine katkı sağladığını savunan bir çalışma yapmışlardır. Sonrasında logistic regression sınıflandırıcısı kullanarak yaptıkları ikili sınıflandırmada %60 doğruluk oranı elde etmişlerdir [49].

Yang ve arkadaşları fast3DS adını verdikleri modelle gerçek zamanlı zararlı alan adı tespiti üzerine çalışma yapmışlardır. Önerdikleri modelde geleneksel evrişim (convolution) yöntemlerinden farklı olarak paralel derinlikli evrişim yaklaşımını kullanmışlardır ve derin öğrenme ile ikili sınıflandırma yaparak %97,25 kesinlik oranına ulaşmışlardır [50].

Sun ve arkadaşları DeepDom adını verdikleri modelde heterojen çizge evrişimsel ağ yöntemi üzerine bir çalışma yaparak önerdikleri modelin yeni yöntemlerle oluşturulan zararlı alan adlarını tespit etmekte daha başarılı olduğunu iddia etmişlerdir ve %97,82

kesinlik oranına ulaşmışlardır [51].

Raja ve arkadaşları sadece sözcüksel öznitelikleri kullanarak Random Forest sınıflandırıcısı kullanarak daha çevik bir model oluşturabilmek için çalışma yapmışlardır ve %99 kesinlik oranı elde etmeyi başarmışlardır [52].

Selvi ve arkadaşları bilgisayar ortamında oluşturulan zararlı alan adlarını maskeli n-gram tekniklerini kullanarak Random Forest sınıflandırıcısı ile sınıflandırmışlardır ve önerdikleri yöntemin muadillerine göre daha performanslı çalıştığını iddia etmişlerdir. Çalışma sonuçlarına göre yaklaşık %98 kesinlik oranına ulaşmışlardır [53].

Li ve arkadaşları öznitelik mühendisliği üzerine bir çalışma yaparak doğrusal olan ve doğrusal olmayan uzay dönüşüm yöntemlerinin bir kombinasyonunu kullanarak çalışma neticesinde KNN, SVM ve MLP algoritmalarında elde ettikleri başarı oranlarında yaklaşık %20 iyileşme gözlemlemişlerdir [54].

Liang ve arkadaşları alan adlarının uzunluğunun zararlı alan adı tespit yöntemlerinin performansı üzerinde önemli düzeyde etkisi olduğunu ve kısa alan adlarına sahip zararlı alan adlarının tespit sistemlerini aldatabilmesinin önüne geçebilmek için alan adlarının uzunluğu ile ilişkili 3 farklı yaklaşımla öznitelik çıkarılması gerektiğini önermişlerdir. Bu şekilde HAGDetector adını verdikleri modelle çok kısa alan adları için %91,63 çok uzun alan adları için %98,75 doğruluk oranları elde etmişlerdir [55].

Çil ve arkadaşları derin sinir ağları (Deep Neural Network) kullanarak DDOS saldırılarını tespit edip sınıflandıran bir model geliştirmişlerdir. Her biri 50 birimden oluşan 3 gizli katman kullanarak oluşturdukları modelde %99,99 oranla ikili sınıflandırma doğruluk oranı elde etmişlerdir ve yaklaşık %95 doğruluk oranıyla DDOS saldırı türünü sınıflandırabilmişlerdir [56].

2. MATERYAL VE YÖNTEM

Tez kapsamında kötücül alan adlarının tespit edilmesi ve sınıflandırılması amacıyla topluluk öğrenmesi yöntemi kullanan bir model önerilmiş ve literatürdeki diğer çalışmalarla kıyaslanmıştır. Ayrıca derin öğrenme yöntemi ile de sınıflandırma sonuçları elde edilerek topluluk öğrenmesi yöntemleri ile kıyaslanmıştır. Tez kapsamında modelin önerilmesi noktasında doğruluk oranı kadar sınıflandırma süresi için harcanan süre de göz önünde bulundurulmuştur.

Kötücül, oltalama, istenmeyen e-posta ve iyicil alan adları sınıflarından verilerin oluşturduğu Kanada Siber Güvenlik Enstitüsü tarafından hazırlanan CIC-Bell-DNS 2021 veri kümesi üzerinde çeşitli topluluk öğrenmesi algoritmaları ve LSTM derin öğrenme yöntemi ile sınıflandırma işlemi yapılmıştır. Sınıflandırma kapsamında veri kümesi hem zararlı ve zararlı olmayan olmak üzere ikili sınıflandırma işlemine tabi tutulmuştur. Hem de söz konusu dört sınıf kapsamında da çoklu sınıflandırma yapılmıştır.

Bu bölümde, tez kapsamında kullanılan veri kümesinden, ön işleme çalışmalarından ve topluluk öğrenmesi ile derin öğrenme modellerinin detaylarından bahsedilecektir.

2.1. Veri Kümesi

Çalışma kapsamında Mahdavifar ve arkadaşlarının hazırladığı CIC-BELL-DNS 2021 veri kümesi kullanılmıştır. Söz konusu veri kümesinde Tablo 2.1'de belirtilen DNS istatistikleri tabanlı, sözcük tabanlı ve üçüncü taraf tabanlı 32 öznitelik bulunmaktadır.

Tablo 2.1. Mahdavifar ve arkadaşlarının veri kümesi için topladığı özellikler.

Öznitelik	Öznitelik Adı	Açıklama
	Sözcük Tabanlı Öznitelikler	
F1	Subdomain	Alt alan adı var mı yok mu
F2	TLD	En üst düzey alan adı
F3	SLD	İkinci düzey alan adı
F4	Len	Alan ve alt alanın uzunluğu
F5	Numeric percentage	Alan ve alt alan içindeki rakamların sayısı
F6	Character distribution	Alan adı içindeki her harfin sayısı

F7	Entropy	Entropi - Harf dağılımının entropisi
F8	1-gram	1-gram - Alanın harf düzeyinde 1-gram'ı
F9	2-gram	2-gram - Alanın harf düzeyinde 2-gram'ı
F10	3-gram	3-gram - Alanın harf düzeyinde 3-gram'ı
F11	Longest word	İkinci düzey alan adındaki en uzun anlamlı kelime
F12	Distance from bad words	Kötü kelimelerden ortalama uzaklık
F13	Typos	Yazım hatası (typosquatting)
F14	Obfuscation	URL gizleme için maksimum değer
	DNS İstatistiksel Öznitelikler	
F15	Unique country	Benzersiz Ülke Adı - Pencere τ içindeki farklı ülke adlarının sayısı
F16	Unique ASN	Benzersiz ASN - Pencere τ içindeki farklı ASN (Autonomous System Number) değerlerinin sayısı
F17	Unique TTL	Benzersiz TTL - Pencere τ içindeki farklı TTL (Time to Live) değerlerinin sayısı
F18	Unique IP	Benzersiz IP - Pencere τ içindeki farklı IP değerlerinin sayısı
F19	Unique domain	Benzersiz Alan Adı - Pencere τ içindeki farklı alan adı değerlerinin sayısı
F20	TTL mean	TTL Ortalaması - Pencere τ içindeki TTL'lerin ortalaması
F21	TTL variance	TTL Varyansı - Pencere τ içindeki TTL'lerin varyansı
	Üçüncü Taraf Tabanlı Öznitelikler	
F22	Domain name	Alan Adı
F23	Registrar	Alan adının kayıt yaptırdığı kurum
F24	Registrant name	Alan adının kaydını yaptıranın adı
F25	Creation date time	Alan adının oluşturulma tarihi ve saati
F26	Emails	Alan adı ile ilişkilendirilen e-postalar
F27	Domain age	Alan adının yaşı
F28	Organization	Hangi organizasyona bağlı olduğu
F29	State	Ana şubenin bulunduğu eyalet
F30	Country	Ana şubenin bulunduğu ülke
F31	Name server count	Alan adıyla ilişkilendirilen toplam ad sunucu sayısı
F32	Alexa rank	Alexa sıralaması

Tablo 2.1'de belirtilen bilgiler arasında sayısal değere sahip olmayanlar

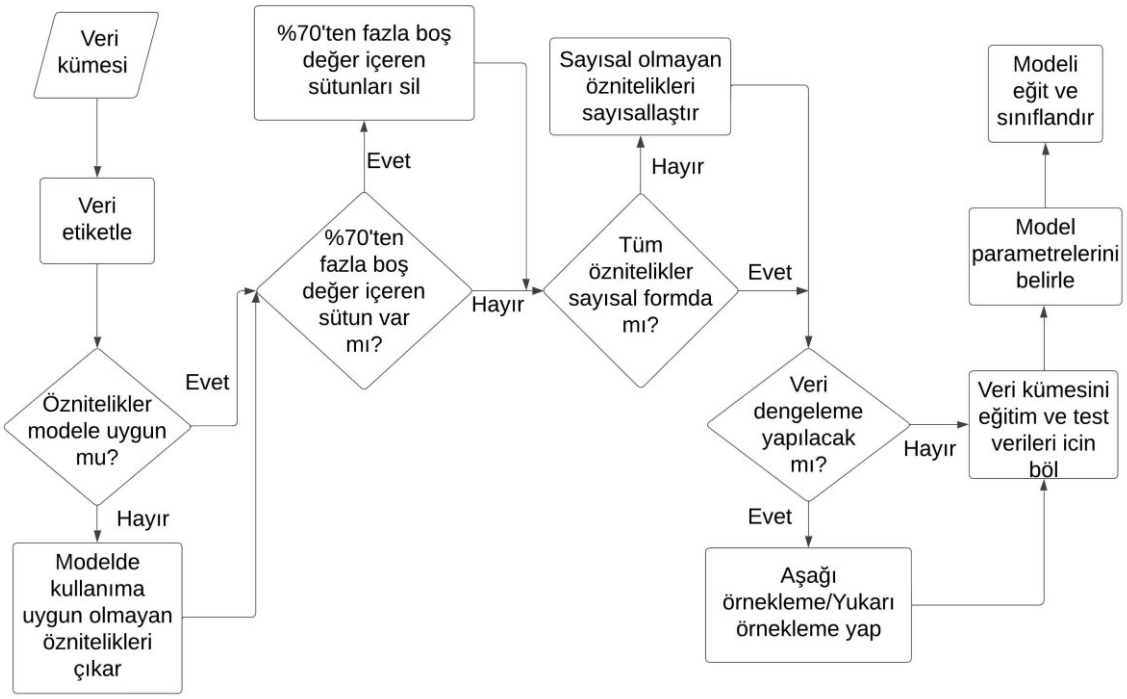
sayısallaştırılmıştır. Kullanılacak modellere uygun olmayan öznitelikler kullanılmamıştır. Nihai durumda tez kapsamında Tablo 2.2'de belirtilen 21 öznitelik kullanılmıştır.

Tablo 2.2. Tez kapsamında kullanılan öznitelikler.

Öznitelik	Açıklama
dec_8	8 bit Decimal IP obfuscation
Page_Rank	Sayfa sıralaması
Alexa_Rank	Alan adının Alexa sıralaması
shortened	URL in kısaltılıp kısaltılmadığı
dec_32	32 bit Decimal IP obfuscation
hex_32	32 bit Hexadecimal IP Obfuscation
subdomain	Alt alan adı var mı yok mu
hex_8	8 bit Octal IP obfuscation
Country	Ana şubenin bulunduğu ülke
oc_8	8 bit Hexadecimal IP obfuscation
Domain_Age	Alan adının yaşı
oc_32	32 bit Octal IP obfuscation
Name_Server_Count	Alan adıyla ilişkilendirilen toplam ad sunucusu
puny_coded	Punycode encode edilip edilmediği
entropy	Harf dağılımının entropisi
len	Alan ve alt alanın uzunluğu
longest_word_length	Alan adındaki en uzun kelimenin uzunluğu
State	Ana şubenin bulunduğu eyalet
tld	En üst seviye alan adı
obfuscate_at_sign	alan adında @ işaretinin varlığı
numeric_percentage	Alan ve alt alan içindeki rakamların sayısı

2.2. Sistem Mimarisi

Çalışma kapsamında iyi performans elde edebilmek için önerilen modele ait akış diyagramı Şekil 2.1'de gösterilmektedir. Model eğitilmeden önce veri kümesi üzerinde ön işleme yapılmıştır. Ön işleme kapsamında veri etiketleme yapılmış ve veri kümesindeki modelin eğitimine uygun olmayan öznitelikler veri kümesinden çıkarılmıştır. Ayrıca %70'in üzerinde null değer içeren kolonlar veri kümesinden çıkarılmıştır. Ardından veri kümesindeki sayısal olmayan veriler sayısallaştırılmıştır. Sayısallaştırma kapsamında "NaN" değerleri de başka herhangi bir kolonda bulunmayan '-990011' değeri ile değiştirilmiştir.



Şekil 2.1. Çalışma kapsamında uygulanan akış diyagramı.

Çalışma kapsamında veri kümesinin dengesiz olup olmaması durumlarının performansa etkilerini inceleyebilmek için hem veri kümesinin kısmi dengeli versiyonu üzerinde hem aşağı örnekleme yapılmış versiyonunda hem de yukarı örnekleme yapılmış versiyonunda çalışılmıştır. Bu bağlamda veri kümesi temizleme adımından sonra NearMiss yöntemiyle aşağı örnekleme ve gerçek veriye yakın sentetik veriler üreterek çeşitliliği koruduğu için SMOTE yöntemiyle de yukarı örnekleme uygulanarak veri dengeleme işlemi yapılmıştır.

Model eğitimine geçmeden önce son olarak veri kümesi %70 eğitim ve %30 test için kullanılmak üzere bölünmüştür. Genel sistem mimarisi Şekil 2.2'de gösterildiği gibidir.



Şekil 2.2. Önerilen sistem mimarisi.

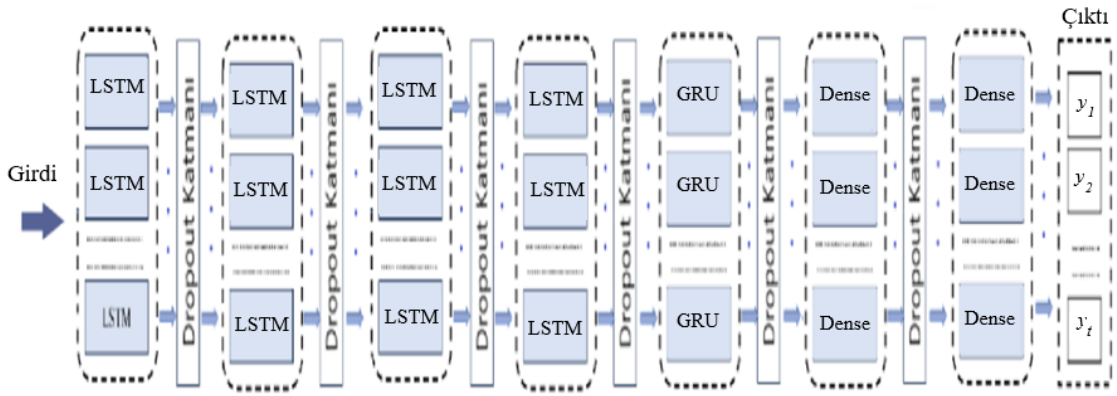
2.3. Model Yapısı

Tez kapsamında hem topluluk öğrenmesi yöntemleri hem de derin topluluk öğrenmesi yöntemleri kullanılarak performanslar değerlendirilmiştir. Bu bağlamda topluluk öğrenmesi algoritmaları arasından Random Forest, XGBoost, Adaboost, Extra Trees ve Bagging algoritmaları kullanılmıştır. Derin topluluk öğrenmesi yaklaşımı için ise LSTM algoritması kullanılmıştır. Tablo 2.3'te çalışmada kullanılan topluluk öğrenmesi algoritmalarında seçilen parametreler görülmektedir. Söz konusu parametrelerin belirlenmesinde RandomizedSearchCV kütüphanesinden faydalanılmıştır.

Tablo 2.3. Algoritmaların parametre değerleri.

Algoritma	**Parametre**	**Değer**
Random Forest	max_features	auto
	min_samples_leaf	1
	max_depth	None
	min_samples_split	2
	n_estimators	200
	cv	5
XGBoost	colsample_bytree	0,8
	gamma	0,05
	learning_rate	0,2
	max_depth	6
	min_child_weight	1
	n_estimators	400
	subsample	0,9
	cv	5
Bagging	max_features	0,7
	max_samples	0,7
	n_estimators	300
	cv	5
Extra trees	n_estimators	500
	min_samples_split	5
	min_samples_leaf	1
	max_depth	40
	cv	5
Adaboost	learning_rate	0,01
	n_estimators	200
	cv	5

LSTM tabanlı oluşturulan derin topluluk öğrenmesi modeli Şekil 2.2'de gösterildiği gibi LSTM, GRU ve Dense katmanlarından oluşmaktadır. Aşırı öğrenmeyi düşürebilmek için her katmanda dropout fonksiyonu uygulanmıştır. Optimizer olarak Adam optimizeri kullanılmıştır. Batch_size parametresi 128 olarak belirlenen modelde validation_split oranı 0,1 seçilmiştir.



Şekil 2.3. Çalışmada kullanılan derin öğrenme mimarisi.

Çalışmada kullanılan LSTM modelinde her bir LSTM katmanındaki gizli nöron sayısı 128 olarak belirlenmiştir ve her LSTM katmanında aşırı öğrenmeyi engellemek için 0,1 oranında dropout uygulanmıştır. Aktivasyon fonksiyonu olarak LSTM katmanlarında ve ilk Dense katmanında tanh fonksiyonu kullanılırken son Dense katmanında softmax kullanılmıştır. Model 25, 50, 100, 150 ve 200 epoch değerleri ile eğitilmiş ve performansı test edilmiştir.

3. BULGULAR VE TARTIŞMA

Bu bölümde zararlı alan adlarının tespiti ve sınıflandırılması için önerilen modelin çalıştırıldığı deney ortamından, önerilen topluluk öğrenmesi yöntemleri kullanılarak veri kümesi üzerinde test edilmesinden ve elde edilen test sonuçlarının doğruluk ve işlem süresi açısından değerlendirilen başarısından söz edilmektedir.

3.1. Deney Ortamı

Tez kapsamında kullanılan topluluk öğrenmesi yöntemleri ve derin topluluk öğrenmesi modelinin eğitim ve test işlemleri için kullanılan ortam Windows 11 işletim sistemine sahiptir. AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz işlemci, 16 GB Ram, NVIDIA GeForce GTX 1650 Ti ekran kartı ve 512 GB SSD sabit disk donanımlarını bünyesinde barındırmaktadır. Python 3.8.8 programlama dili kullanılarak Spyder IDE 4.2.5 [57] geliştirme ortamında model eğitilip test edilmiştir. Modelin geliştirilmesi aşamasında Pandas [58], Keras [59] ve Tensorflow [60] kütüphaneleri kullanılmıştır.

3.2. Performans Ölçütleri

Kullanılan topluluk öğrenmesi yöntemlerinin performansını değerlendirebilmek için test işlemi sonucunda elde edilen doğruluk, kesinlik, duyarlılık, F1 değeri ve işlem süresi ölçütleri dikkate alınmıştır.

3.2.1. Doğruluk

Makine öğrenmesi ya da derin öğrenme yöntemleri ile sınıflandırma işlemi yapılırken model tarafından yapılan isabetli tahminlerin tüm tahminlere oranı hesaplanarak bulunur. (3.1) numaralı denklemdeki formülle hesaplanmaktadır.

$$\text{Doğruluk} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

3.2.2. Kesinlik

Sınıflandırma işleminde (3.2) denkleminde belirtildiği şekilde isabetli pozitif tahminlerin tüm pozitif tahminlere oranı hesaplanarak tespit edilir. Yanlış pozitiflerin sayısının etkisinin minimize edilmesi gereken sistemlerde değerlendirmeye dahil edilmesi önem arz eder.

$$\text{Kesinlik} = \frac{TP}{TP + FP} \quad (3.2)$$

3.2.3. Duyarlılık

Sınıflandırma problemlerinde (3.3) denkleminde belirtildiği şekilde isabetli pozitif tahminlerin, isabetli pozitif ve isabetsiz negatif tahminlerin toplamına oranı hesaplanarak bulunur.

$$\text{Duyarlılık} = \frac{TP}{TP + FN} \quad (3.3)$$

3.2.4. F1 değeri

F1 değeri, (3.4) numaralı denklemdeki formül aracılığıyla kesinlik ve duyarlılık arasındaki ilişkiyi kullanarak modelin ne kadar dengeli olduğunu ifade eder.

$$\text{F1 Değeri} = 2 \times \frac{Kesinlik \times Duyarlılık}{Kesinlik + Duyarlılık} \quad (3.4)$$

3.2.5. Test süresi

Sınıflandırma sürecinde modelin ya da sınıflandırıcının 1 adet örneği sınıflandırmak için harcadığı zamanı ifade eder. Tez kapsamında odaklanılan noktalardan biri de farklı algoritmaların ve modellerin sınıflandırma için harcanan zamanı göz önünde bulundurarak etkin bir şekilde kullanılabileceğini ortaya koymaktır.

3.3. Deney Sonuçları

Bölüm 2'de bahsedilen topluluk öğrenmesi yöntemleri ve derin topluluk öğrenmesi modeli CIC-Bell-DNS 2021 veri kümesi üzerinde kullanılarak çalışma yapılmıştır. Çalışmada hem topluluk öğrenmesi yöntemleri hem de derin topluluk öğrenmesi modeli ayrı ayrı kullanılmıştır. Her iki yaklaşımla da hem ikili sınıflandırma hem de çoklu sınıflandırma yapılmıştır. Bu bölümde iki alt başlıkta her iki yaklaşıma ait sonuçlara yer verilmiştir.

3.3.1. Topluluk öğrenmesi yöntemleri ile elde edilen sonuçlar

Tez kapsamında kullanılan veri kümesi dengesiz olduğu için aşağı örnekleme ve yukarı örnekleme işlemlerinden faydalanılmıştır. Nihai durumda veri kümesinin kısmen dengeli versiyonu, aşağı örnekleme yapılması sonucu oluşan versiyonu ve yukarı örnekleme

işlemi sonucunda oluşan versiyonu ayrı ayrı test edilerek sonuçlar değerlendirilmiştir.

Tablo 3.1'de kısmen dengeli veri kümesi üzerinde topluluk öğrenmesi yöntemleri ile ikili sınıflandırma yapılması sonucu elde edilen sonuçlar görülmektedir.

Tablo 3.1. Topluluk öğrenmesi yöntemleri ile kısmen dengeli veri kümesi üzerinde ikili sınıflandırma sonuçları.

Algoritma	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
Extra Trees	0,9929	0,9929	0,9923	0,9929	0,0114
XGBoost	0,9923	0,9921	0,9918	0,9923	0,0018
Random Forest	0,9920	0,9918	0,9915	0,9920	0,0092
Bagging	0,9914	0,9915	0,9905	0,9914	0,0026
AdaBoost	0,9898	0,9902	0,9884	0,9898	0,0066

Tablo 3.2'de veri kümesi üzerinde aşağı örnekleme yapılıp topluluk öğrenmesi yöntemleri ile ikili sınıflandırma yapılması sonucu elde edilen sonuçlar görülmektedir.

Tablo 3.2. Aşağı örnekleme yapılmış veri kümesi üzerinde topluluk öğrenmesi yöntemleri ile ikili sınıflandırma sonuçları.

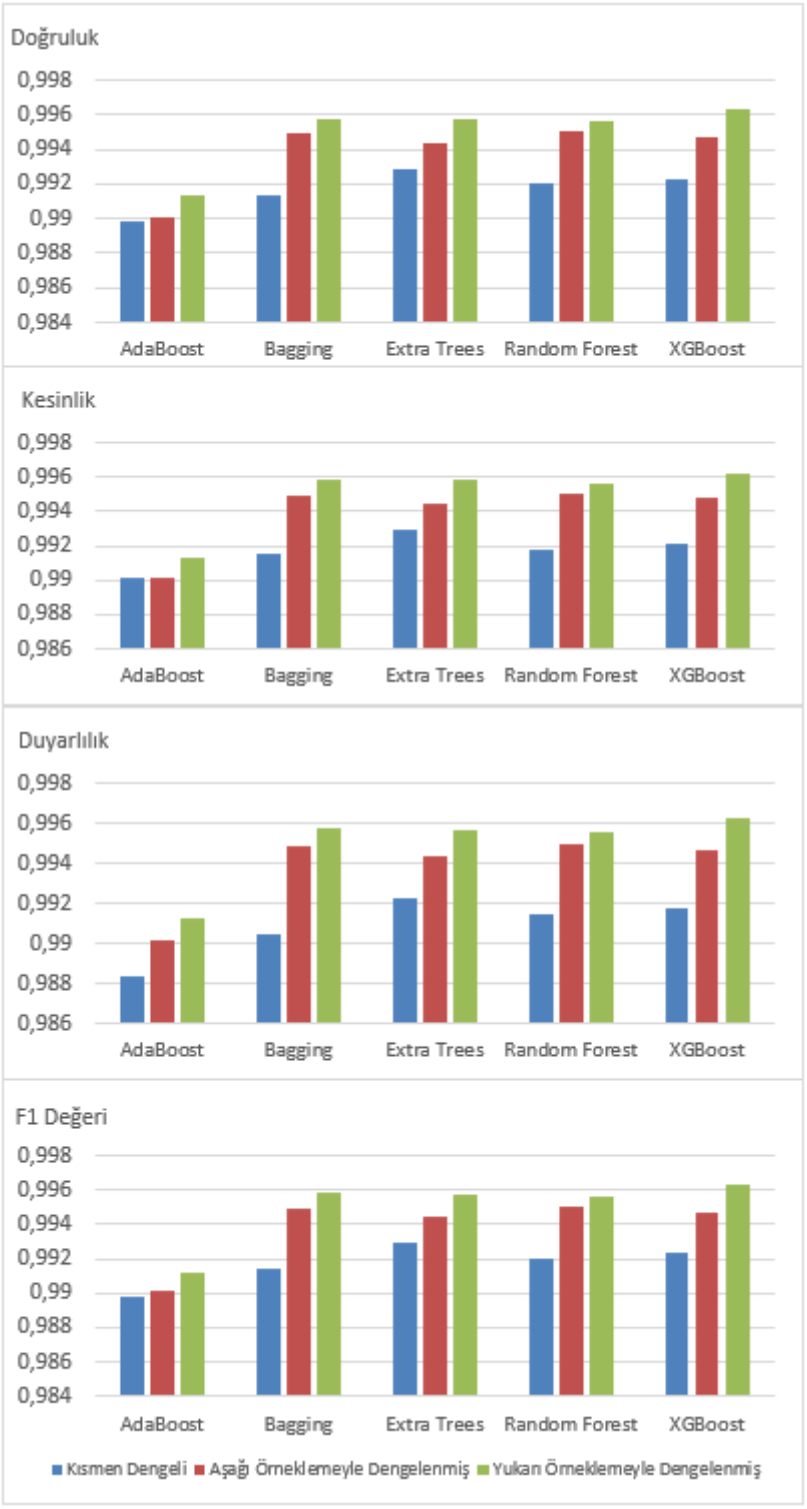
Algoritma	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
Random Forest	0,9950	0,9950	0,9950	0,9950	0,0081
Bagging	0,9949	0,9949	0,9949	0,9949	0,0022
XGBoost	0,9947	0,9948	0,9947	0,9947	0,0014
ExtraTrees	0,9944	0,9944	0,9944	0,9944	0,0109
AdaBoost	0,9901	0,9901	0,9902	0,9901	0,0078

Tablo 3.3'te veri kümesi üzerinde yukarı örnekleme yapılıp topluluk öğrenmesi yöntemleri ile ikili sınıflandırma yapılması sonucu elde edilen sonuçlar görülmektedir.

Tablo 3.3. Yukarı örnekleme yapılmış veri kümesi üzerinde topluluk öğrenmesi yöntemleri ile ikili sınıflandırma sonuçları.

Algoritma	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
XGBoost	0,9963	0,9962	0,9963	0,9963	0,0012
Bagging	0,9958	0,9958	0,9958	0,9958	0,0023
Extra Trees	0,9958	0,9958	0,9957	0,9957	0,0135
Random Forest	0,9956	0,9956	0,9956	0,9956	0,0091
AdaBoost	0,9913	0,9913	0,9913	0,9912	0,0064

Şekil 3.1'de de görülebileceği gibi topluluk öğrenmesi algoritmalarıyla yapılan ikili sınıflandırma deneyleri sonucunda veri kümesi üzerinde yukarı örnekleme yaptıktan sonra elde edilen sonuçlar doğruluk, kesinlik, duyarlılık ve F1 değeri bakımından az da olsa daha başarılı olmuştur. Sonuçlar incelendiğinde algoritmaların doğruluk performanslarının yakın olduğu ancak her bir örnek veri için harcanan test süresinin algoritmalar arasında farklılık gösterdiği anlaşılmaktadır. Tablo 3.3'teki test süreleri incelendiğinde XGBoost algoritmasının işlem süresi açısından birim işlem süresinin 0,0012 ms olduğu hesaplanmıştır. Diğer yandan Random Forest ve Extra Trees algoritmalarının birim işlem süresinin görece daha uzun olduğu tespit edilmiştir. Doğruluk ve kesinlik oranlarındaki yüksek başarı oranı da göz önünde bulundurulduğunda yeterli örnek sayısına sahip olunan durumda topluluk öğrenmesi algoritmaları arasından ikili sınıflandırma problemlerinde performans açısından en uygun algoritmanın XGBoost algoritması olduğu sonucuna varılmaktadır.



Şekil 3.1. Topluluk öğrenmesi ile ikili sınıflandırma performans özeti.

Tablo 3.4'te kısmen dengeli veri kümesi üzerinde topluluk öğrenmesi yöntemleri ile çoklu sınıflandırma yapılması sonucu elde edilen sonuçlar görülmektedir.

Tablo 3.4. Kısmen dengeli veri kümesi üzerinde topluluk öğrenmesiyle çoklu sınıflandırma sonuçları.

Algoritma	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
XGBoost	0,9976	0,9961	0,9960	0,9976	0,0018
Random Forest	0,9963	0,9955	0,9939	0,9963	0,0099
Bagging	0,9949	0,9930	0,9928	0,9948	0,0026
Extra Trees	0,9908	0,9922	0,9842	0,9907	0,0131
AdaBoost	0,8739	0,8494	0,8273	0,8649	0,0086

Tablo 3.5'te veri kümesi üzerinde aşağı örnekleme işlemi yapıldıktan sonra topluluk öğrenmesi yöntemleri ile çoklu sınıflandırma yapılması sonucu elde edilen sonuçlar görülmektedir.

Tablo 3.5. Aşağı örneklemeyle dengelenmiş veri kümesi üzerinde topluluk öğrenmesiyle çoklu sınıflandırma sonuçları.

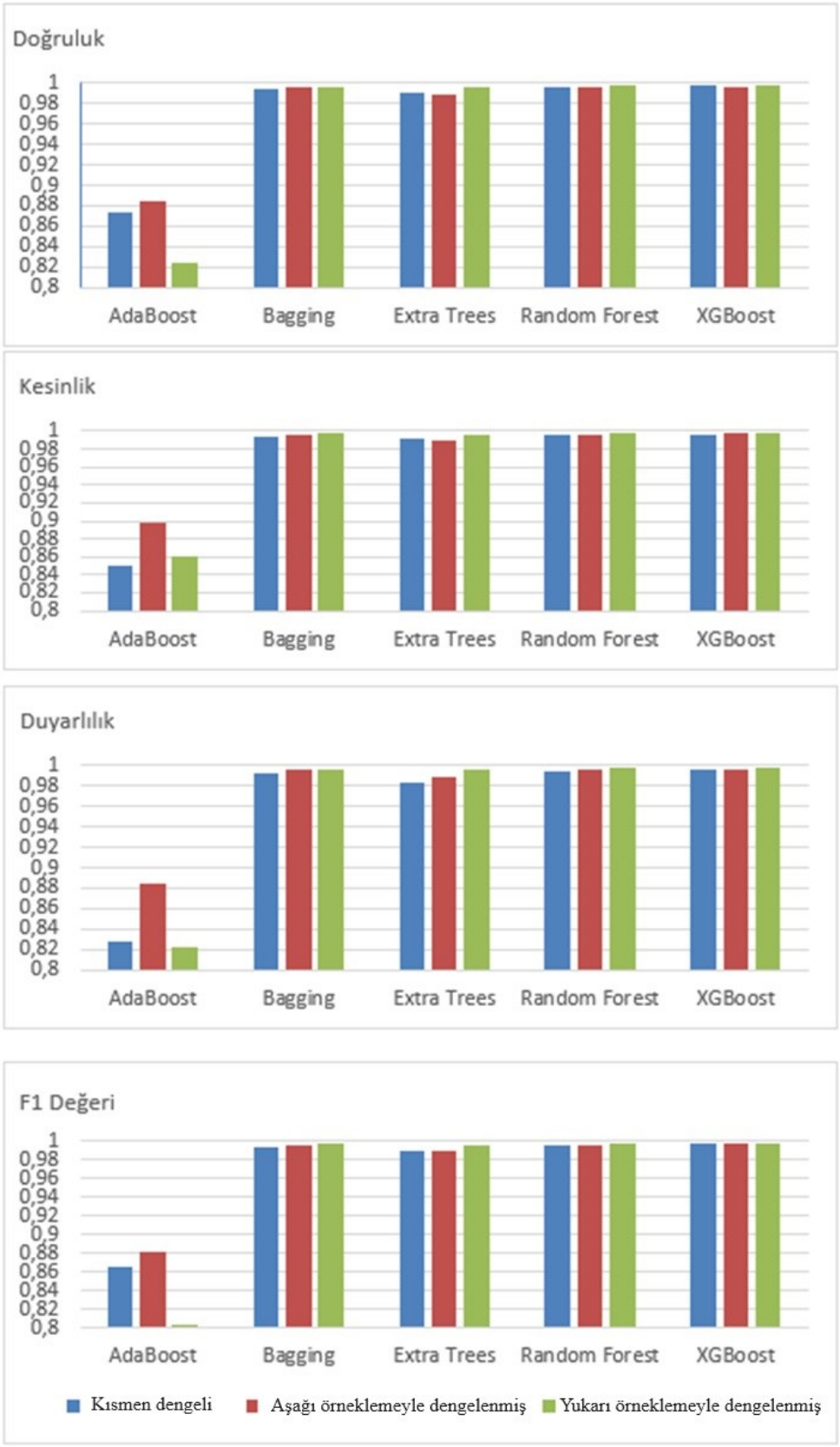
Algoritma	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
XGBoost	0,9969	0,9969	0,9969	0,9969	0,0025
Bagging	0,9960	0,9960	0,9960	0,9960	0,0023
Random Forest	0,9958	0,9958	0,9958	0,9958	0,0105
Extra Trees	0,9889	0,9888	0,9889	0,9889	0,0147
AdaBoost	0,8836	0,8981	0,8846	0,8806	0,0086

Tablo 3.6'da veri kümesi üzerinde yukarı örnekleme işlemi yapıldıktan sonra topluluk öğrenmesi yöntemleri ile çoklu sınıflandırma yapılması sonucu elde edilen sonuçlar görülmektedir.

Tablo 3.6. Yukarı örneklemeyle dengelenmiş veri kümesi üzerinde topluluk öğrenmesiyle çoklu sınıflandırma sonuçları.

Algoritma	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
XGBoost	0,9981	0,9981	0,9981	0,9981	0,0014
Random Forest	0,9975	0,9975	0,9975	0,9975	0,0113
Bagging	0,9970	0,9970	0,9970	0,9970	0,0024
ExtraTrees	0,9963	0,9962	0,9962	0,9963	0,0136
AdaBoost	0,8234	0,8608	0,8220	0,8023	0,0086

Söz konusu veri kümesi üzerinde yukarı örnekleme yapıldığında Şekil 3.2'de de görülebileceği gibi XGBoost, Random Forest, Bagging ve Extra Trees algoritmalarında doğruluk, kesinlik, duyarlılık ve F1 değerleri açısından performans, kısmen dengeli ve aşağı örneklenmiş veri kümesine oranla daha başarılı olmuştur. Ancak yukarı örnekleme yöntemleri her durumda her algoritma için doğruluk oranlarında iyileşme sağlamaz [61]. Bu bakımdan Adaboost algoritması ile yukarı örnekleme yapılmış veri kümesinde doğruluk oranının iyileşmemesi normal kabul edilmiştir. Öte yandan 1 adet örneği sınıflandırmak için geçen test süresi bakımından sınıflandırıcıların performansları daha önceki tablolarda belirtilen değerlerle beklenildiği gibi benzerlik göstermektedir. Tablolar incelendiğinde en yavaş algoritmaların Extra Trees ve Random Forest algoritmaları olduğu görülmektedir. Gerek doğruluk gerekse test süresi açısından ise en performanslı algoritmanın XGBoost algoritması olduğu tespit edilmiştir.



Şekil 3.2. Topluluk öğrenmesi ile çoklu sınıflandırma performans özeti.

3.3.2. Derin topluluk öğrenmesi yöntemleri ile elde edilen sonuçlar

Derin topluluk öğrenmesi modelinin kullanımı esnasında 25, 50, 100, 150 ve 200 epoch değerleri ile model eğitilerek deney yapılmıştır. Derin öğrenme algoritmaları yüksek sayıda veriye sahip olunduğunda daha yüksek performans gösterdiği için veri kümesi üzerinde yukarı örnekleme işlemi yapılmıştır. Ardından hem ikili sınıflandırma hem de çoklu sınıflandırma işlemi yapılmıştır.

Tablo 3.7'de veri kümesi üzerinde yukarı örnekleme yapıldıktan sonra derin topluluk öğrenmesi modeliyle ikili sınıflandırma işlemi sonuçları yer almaktadır.

Tablo 3.7. Derin topluluk öğrenmesi modeliyle yukarı örneklenmiş veri kümesi üzerinde yapılan ikili sınıflandırma sonucu.

Epoch	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
25	0,8777	0,8792	0,8779	0,8777	0,4435
50	0,9151	0,9169	0,9153	0,9150	0,4263
100	0,9435	0,9435	0,9435	0,9434	0,4152
150	0,9454	0,9454	0,9454	0,9454	0,4817
200	0,9529	0,9532	0,9530	0,9529	0,4694

Tablo 3.7 incelendiğinde düşük epoch değerlerinde modelin yeterince eğitilemediği, ancak 100 epoch kullanımından sonra dengeli bir model elde edildiği görülmektedir.

İkili sınıflandırma işleminin yanı sıra çoklu sınıflandırma işlemi için de aynı model kullanılmış ve Tablo 3.8'deki sonuçlar elde edilmiştir.

Tablo 3.8. Derin topluluk öğrenmesi modeliyle yukarı örneklenmiş veri kümesi üzerinde yapılan çoklu sınıflandırma sonucu.

Epoch	Doğruluk	Kesinlik	Duyarlılık	F1	Test Süresi (ms)
25	0,8337	0,8670	0,8238	0,8238	0,4606
50	0,9308	0,9353	0,9320	0,9308	0,4713
100	0,9432	0,9446	0,9437	0,9432	0,4672
150	0,9522	0,9532	0,9523	0,9522	0,5082
200	0,9522	0,9531	0,9526	0,9522	0,4462

Tablo 3.8'deki sonuçlara göre çoklu sınıflandırma işlemlerinde de ikili sınıflandırma işleminde olduğu gibi düşük epoch değerlerinde modelin iyi eğitilemediği anlaşılmaktadır. Ancak yine yaklaşık 100 epoch kullanımından sonra modelin yaklaşık %94,32 doğruluk sonucuna ulaştığı görülmektedir. İkili sınıflandırmadan farklı olarak

düşük epoch değerlerindeki doğruluk ve kesinlik oranları ikili sınıflandırmaya oranla daha düşük olmuştur. Ancak yeterli epoch seviyelerinde modelin performansı yaklaşık olarak ikili sınıflandırma ile aynı seviyelere gelmiştir. Bu sonuçlara göre önerilen derin topluluk öğrenmesi modeliyle en iyi sonuçlara 150 epoch değeriyle ulaşılmıştır. 1 adet örneğin sınıflandırılması için harcanan süre ise 0,46 ms olarak ölçülmüştür.

Sonuç olarak CIC-Bell-DNS 2021 veri kümesi üzerinde topluluk öğrenmesi sınıflandırıcıları ve derin topluluk öğrenmesi modeliyle yapılan deney sonuçlarına göre topluluk öğrenmesi sınıflandırıcıları kullanıldığında doğruluk, kesinlik, duyarlılık, F1 değeri ve işlem süresi ölçütlerinin tamamında daha yüksek performans elde edilmiştir. Özellikle algoritma olarak XGBoost kullanıldığında hem doğruluk performansının daha yüksek olduğu hem de sınıflandırma süresinin kullanılan derin topluluk öğrenmesi modeline göre yaklaşık 41 kat daha hızlı olduğu hesaplanmıştır. Bu sebeple CIC-Bell-DNS 2021 veri kümesi üzerinde ikili ve çoklu sınıflandırma işlemi yapılmak istenirse yukarı örnekleme yapıp XGBoost algoritmasının kullanılması hem doğruluk hem de hız açısından önerilmektedir.

3.4. Diğer Sonuçlarla Karşılaştırma

Deney kapsamında geliştirilen modellerin performansı literatürdeki benzer diğer çalışmalarla karşılaştırılmıştır. Zararlı alan adı tespiti için kullanılan veri kümesini hazırlayan Mahdavifar ve arkadaşları çalışmalarını veri kümesini 2 farklı şekilde kullanarak gerçekleştirmişlerdir. Bunlardan birinde 20.000 iyicil, 13.011 kötücül verilerden oluşan bir oranda oluşturdukları veri kümesi parçasıyla araştırmalarını yaparken diğerinde de 400.000 iyicil alan adı ve 13.011 kötücül alan adı üzerinden yapmışlardır.

Bu tez kapsamında yapılan deneylerden biri de Mahdavifar ve arkadaşları gibi 20.000 iyicil ve 13.011 kötücül alan adı arasında 4 sınıf için sınıflandırma yapılmasıdır. Yapılan deneyler neticesinde tez kapsamında önerilen topluluk öğrenmesi yöntemleri ile oluşturulan modellerin Tablo 3.9'da da görülebileceği gibi Mahdavifar ve arkadaşlarının oluşturduğu k-NN modelinden daha iyi performans gösterdiği tespit edilmiştir. Algoritmanın hızı da göz önünde bulundurulduğunda hem çok yüksek doğruluk ve kesinlik performansını sağladığı hem de sınıflandırma işlemini çok hızlı yaptığı için XGBoost algoritması önerilmektedir.

Tablo 3.9. Literatürle karşılaştırma.

Algoritma	**Doğruluk**	**F1 Değeri**	**Kesinlik**
k-NN (Mahdavifar et al.)	94,8%	94,8%	94,9%
XGBoost	99,8%	99,8%	99,6%
Random Forest	99,6%	99,6%	99,5%
Bagging	99,5%	99,5%	99,3%
Extra Trees	99,1%	99,1%	99,2%

4. SONUÇLAR

Bu tez çalışmasında, zararlı alan adlarının topluluk öğrenmesi yöntemleri ile tespiti ve sınıflandırılması için bir topluluk öğrenmesi yöntemi önerilmiştir. Literatürde ve yapay zeka yarışmalarında topluluk öğrenmesi algoritmalarının yüksek başarı elde etmesi sebebiyle bu yöntem üzerinde çalışılmıştır. Derin topluluk öğrenmesi yöntemi de kullanılarak çalışma zenginleştirilmiş ve kullanılan veri kümesinde farklı algoritmaların ne ölçüde performans gösterdiği incelenmiştir. Deneyler kapsamında hem ikili sınıflandırma hem de çoklu sınıflandırma kapsama dahil edilmiştir.

Çalışma kapsamında farklı veri dengeleme işlemleri yapılmış ve hem veri dengelemenin hem de veri sayılarının başarım oranlarına olan etkisi görülmüştür. Çalışma süresince algoritmaların doğruluk, kesinlik, F1 değeri gibi performans ölçütlerinin yanı sıra sınıflandırma için geçen süre de gözlenmiştir. Deneyler neticesinde hem hız hem de diğer performans kriterleri göz önünde bulundurulduğunda XGBoost algoritmasının diğer algoritmalardan daha başarılı bir performans gösterdiğini söylemek mümkündür. Çoklu sınıflandırmada yukarı örnekleme işlemi yapılmış veri kümesinde XGBoost algoritması ile %99,81 doğruluk oranı elde edilmiş ve her bir örnek için harcanan sınıflandırma süresi sadece 0,0014 ms düzeyinde kalmıştır. Benzer şekilde aşağı örnekleme yapılmış veri kümesinde ve kısmen dengeli veri kümesinde yapılan deneylerde de sırasıyla % 99,69 ve %99,76 başarı oranı elde edilmiştir. Doğruluk oranı açısından Random Forest algoritması da %99'un üzerinde yüksek başarı elde etse de hız açısından 0,0114 ms sürede sınıflandırma işlemini yaptığı için XGBoost algoritmasının gerisinde kalmıştır.

Çalışma kapsamında LSTM kullanılarak geliştirilen model ile yapılan deneylerde ise başarı oranı %95,29'u geçememiştir. Ayrıca sınıflandırma süresi de 0,44 ms olarak ölçülmüştür.

Çalışma kapsamında kullanılan veri kümesini oluşturan araştırmacıların kendi makalelerinde yer verdikleri deney sonuçları ile bu tez çalışması kapsamında yapılan deney sonuçları Tablo 3.9'da gösterilmiştir. Tablo 3.9'daki k-NN sonucu Mahdavifar ve arkadaşlarının çalışmalarında 20.000 iyicil ve 13.011 kötücül (4337 adet malware, 4337 adet phishing ve 4337 adet spam) alan adından oluşan %60/%40 düzeyinde dengeli veri kümesi üzerinde önerdiği k-NN algoritması kullanan modelin performansını göstermektedir. Tablodaki diğer sonuçlar bu tez kapsamında elde edilen sonuçlardır. Bu

doğrultuda XGBoost algoritması ile bu tez kapsamında önerilen modelin KNN algoritması ile önerilen modelden daha iyi sonuç verdiği görülmektedir. Bu sebeple CIC-BELL-DNS-2021 veri kümesi ile XGBoost algoritmasının kullanılması önerilmektedir.

Önerilen modelin çok yüksek başarı oranı ve yüksek hızı sayesinde Saldırı Tespit Sistemlerine (IDS) ve Saldırı Önleme Sistemlerine (IPS) entegre edilmesi ile birlikte gelecekte yaşanabilecek pek çok siber zafiyetin önüne geçebileceği değerlendirilmektedir.

KAYNAKLAR/REFERENCES

[1] Gürcistan Üniversite Sistemi, A Brief History of the Internet, https://www.usg.edu/galileo/skills/unit07/internet07_02.phtml#:~:text=January%201%2C%201983%20is%20considered,Protocol%20(TCP%2FIP). (19.06.2023).

[2] Falliere, N., O'Murchu, L., & Chien, E. (2010). W32.Stuxnet Dossier, Version 1.3, Symantec Corporation, https://www.wired.com/images_blogs/threatlevel/2010/11/w32_stuxnet_dossier.pdf (24.09.2023).

[3] Guembe, B., Azeta, A., Misra, S., Osamor, V. C., Fernandez-Sanz, L., Pospelova, V. (2022). The Emerging Threat of Ai-driven Cyber Attacks: A Review. Applied Artificial Intelligence, 36(1), 1–34.

[4] Preda, M. D., Christodorescu, M., Jha, S., Debray, S. (2007). A semantics-based approach to malware detection. Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 377–388.

[5] Mahdavifar, S., Maleki, N., Lashkari, A. H., Broda, M., Razavi, A. H. (2021). Classifying Malicious Domains using DNS Traffic Analysis. The 19th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC), Oct. 25-28, 2021, Calgary, Kanada.

[**6**] Fouchereau R. (2022). EfficientIP https://efficientip.com/resources/idc-dns-threat-report-2022/ (10.09.2023).

[7] Shi, Z. (2011). Advanced Artificial Intelligence, World Scientific Publishing Company, Singapur.

[8] Turing, A. M. (1950). Computing Machinery and Intelligence. Mind, 59(236), 433–460.

[9] Nilsson, N. J. (1998). Introduction To Machine Learning An Early Draft Of a Proposed Textbook, https://ai.stanford.edu/~nilsson/MLBOOK.pdf (25.09.2023).

[10] Mitchell, T. M., (1997). Machine Learning, McGraw-Hill Education, Almanya.

[11] C, G.L.T., Shang, M., (2021). Hands-on Supervised Learning with Python: Learn How to Solve Machine Learning Problems with Supervised Learning Algorithms Using Python (English Edition). BPB PUBN.

[12] Jo, T., (2021). Machine Learning Foundations: Supervised, Unsupervised, and Advanced Learning, Springer, Cham, İsviçre.

[13] Theobald, O. (2017). Machine learning for absolute beginners: a plain English introduction, 157, Scatterplot press, Londra, Birleşik Krallık.

[14] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., Levine, S., (2018). QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation.

[15] Graesser, L., & Keng, W. L. (2019). Foundations of deep reinforcement learning. Addison-Wesley Professional., Boston.

[16] Akhtar, F. (2017). Practical Reinforcement Learning: Develop Self-Evolving, İntelligent Agents With Openaı Gym, Phyton, And Java. Packt, Birmingham Mumbai.

[17] Kunapuli, G. 2023. Ensemble Methods For Machine Learning. Manning, Shelter Island, NY.

[18] Polikar, R. (2006). Ensemble Based Systems In Decision Making. IEEE Circuit Syst. Mag. 6, 21-45. Circuits And Systems Magazine, IEEE. 6. 21 - 45.

[19] Olson, R.S., La Cava, W., Mustahsan, Z., Varik, A., Moore, J.H., (2017). Data-Driven Advice For Applying Machine Learning To Bioinformatics Problems.

[20] Samantaray, R., Das, H. (2023). Performance Analysis of Machine Learning Algorithms Using Bagging Ensemble Technique for Software Fault Prediction. In: 2023 6th International Conference on Information Systems and Computer Networks (ISCON). Presented at the 2023 6th International Conference on Information Systems and Computer Networks (ISCON), IEEE, Mathura, Hindistan, 1–7.

[21] Wang, W., Chakraborty, G., Chakraborty, B. (2020). Predicting the risk of chronic kidney disease (ckd) using machine learning algorithm. Applied Sciences, 11(1), 202.

[22] Chen, T., Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 785-794.

[23] Navlani, A. (2018). Datacamp, https://www.datacamp.com/tutorial/adaboost-classifier-python (27.10.2023)

[24] Deogade, K. R., Thorat, D. B., Kale, S. V., Rajput, S., & Kaur, H. (2022). Credit Card Fraud Detection using Bagging and Boosting Algorithm. In 2022 International Conference on Signal and Information Processing, IConSIP 2022. Institute of Electrical and Electronics Engineers Inc.

[25] Srinilta, C. and Kanharattanachai, S. (2021). "Application of Natural Neighbor-based

Algorithm on Oversampling SMOTE Algorithms," 2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST), Pattaya, Tayland, 217-220

[26] Das, R. Biswas, S. K., Devi, D., Sarma, B. (2020). "An Oversampling Technique by Integrating Reverse Nearest Neighbor in SMOTE: Reverse-SMOTE," 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, Hindistan, 1239-1244

[27] Andreaperlato, Synthetic Minority Oversampling Technique, https://www.andreaperlato.com/theorypost/synthetic-minority-oversampling-technique/ (27.10.2023)

[28] Imbalanced learn, Under-sampling, https://imbalanced-learn.org/stable/under_sampling.html#controlled-under-sampling (27.10.2023)

[29] Anderson, J. A., (2009). Introduction to Artificial Neural Networks. Vikas Publishing House Pvt Limited, Hindistan.

[30] Priddy, K. L., Keller, P. E. (2005). Artificial Neural Networks: An Introduction. SPIE Press, ABD.

[31] Towardsdatascience, McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron, https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1 (31.08.2023)

[32] da Silva, I. N., Hernane Spatti, D. Andrade Flauzino, R., Liboni, L. H. B., dos Reis Alves, S. F., (2016). Artificial Neural Networks: A Practical Course. Springer International Publishing, Almanya.

[33] Pomerat, J., Segev, A., Datta, R. (2019). On Neural Network Activation Functions and Optimizers in Relation to Polynomial Regression, 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, ABD, 6183-6185.

[34] Apicella, A., Donnarumma, F., Isgrò, F. & Prevete, R. (2021). A survey on modern trainable activation functions. Neural Netw. **138**, 14–32.

[35] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[36] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[37] Oreilly, Feed-forward and feedback networks, https://www.oreilly.com/library/view/neural-networks-with/9781788397872/eb49931a-7122-4b97-a843-bb6e1817cc12.xhtml (03-09-2023)

[38] Mei, B. X., Chen, L. S., Sun, S. J., Chen, P. Y., & Huang, W. L. (2023). A New Multi-Feature Recommendation Model Based on Recurrent Neural Network. 2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom) 235-240.

[39] Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 12(7), e0180944.

[40] How, D. N. T., Loo, C. K., & Sahari, K. S. M. (2016). Behavior recognition for humanoid robots using long short-term memory. International journal of advanced robotic systems, 13(6), 1729881416663369.

[41] Palangi, H., Ward, R., & Deng, L. (2016). Distributed compressive sensing: A deep learning approach. IEEE Transactions on Signal Processing, 64(17), 4504-4518.

[42] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[43] Salihoğlu, S. (2023). Enhancing next destination prediction: An application of LSTM using real-world airline data. Yüksek Lisans Tezi, Orta Doğu Teknik Üniversitesi, Ankara, Türkiye, 1-62.

[44] Kang, R., Park, B., Ouyang, Q., & Ren, N. (2021). Rapid identification of foodborne bacteria with hyperspectral microscopic imaging and artificial intelligence classification algorithms. Food Control, 130, 108379.

[45] Yang, L., Liu, G., Wang, J., Zhai, J., & Dai, Y. (2022). A semantic element representation model for malicious domain name detection. Journal of Information Security and Applications, 66, 103148.

[46] Cucchiarelli, A., Morbidoni, C., Spalazzi, L., & Baldi, M. (2021). Algorithmically generated malicious domain names detection based on n-grams features. Expert Systems with Applications, 170, 114551.

[47] Chiong, R., Wang, Z., Fan, Z., & Dhakal, S. (2022). A fuzzy-based ensemble model for improving malicious web domain identification. Expert Systems with Applications, 204, 117243.

[48] Sun, Y., Jian, K., Cui, L., Jiang, G., Zhang, S., Zhang, Y., & Pei, D. (2022). Online malicious domain name detection with partial labels for large-scale dependable

systems. Journal of Systems and Software, 190, 111322.

[49] Palaniappan, G., Sangeetha, S., Rajendran, B., Goyal, S., & Bindhumadhava, B. S. (2020). Malicious domain detection using machine learning on domain name features, host-based features and web-based features. Procedia Computer Science, 171, 654-661.

[50] Yang, L., Liu, G., Wang, J., Bai, H., Zhai, J., & Dai, Y. (2021). Fast3DS: A real-time full-convolutional malicious domain name detection system. Journal of Information Security and Applications, 61, 102933.

[51] Sun, X., Wang, Z., Yang, J., & Liu, X. (2020). Deepdom: Malicious domain detection with scalable and heterogeneous graph convolutional networks. Computers & Security, 99, 102057.

[52] Raja, A. S., Vinodini, R., & Kavitha, A. (2021). Lexical features based malicious URL detection using machine learning techniques. Materials Today: Proceedings, 47, 163-166.

[53] Selvi, J., Rodríguez, R. J., & Soria-Olivas, E. (2019). Detection of algorithmically generated malicious domain names using masked N-grams. Expert systems with applications, 124, 156-163.

[54] Li, T., Kou, G., & Peng, Y. (2020). Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. Information Systems, 91, 101494.

[55] Liang, J., Chen, S., Wei, Z., Zhao, S., & Zhao, W. (2022). HAGDetector: Heterogeneous DGA domain name detection model. Computers & Security, 120, 102803.

[56] Cil, A. E., Yildiz, K., & Buldu, A. (2021). Detection of DDoS attacks with feed forward based deep neural network model. Expert Systems With Applications, 169.

[57] Raybaut, P. (2009). Spyder IDE, https://www.spyder-ide.org/ (27.11.2023)

[58] McKinney, W. (2009). Pandas, https://pandas.pydata.org/ (27.11.2023)

[59] Chollet, F. (2015). Keras, https://keras.io/ (27.11.2023)

[60] The Google Brain Team (2015). TensorFlow, https://www.tensorflow.org/ (27.11.2023)

[61] Abacı, İ., & Yıldız, K. (2023). SMOTE vs. KNNOR: An evaluation of oversampling techniques in machine learning. Gümüşhane Üniversitesi Fen Bilimleri Dergisi, 13(3), 767-779.

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı : Ahmet AYDIN
Yabancı Dili : İngilizce

Öğrenim Durumu

Derece	Eğitim Birimi	Mezuniyet Tarihi
Lisans	Ankara Üniversitesi Elektrik-Elektronik Mühendisliği	Haziran 2014
Lise	Elazığ Anadolu Lisesi	Haziran 2008

İş Deneyimi

Yıl	Yer	Görev
2015 - Halen	TÜBİTAK BİLGEM Kamu SM	Yazılım Geliştirme Mühendisi