T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AĞ TABANLI SALDIRI TESPİT SİSTEMLERİNDE TOPLULUK ÖĞRENME YÖNTEMLERİNİN KARŞILAŞTIRMALI PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

Anıl KURT

Enstitü Anabilim Dalı	**:**	**BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Prof. Dr. İsmail Hakkı CEDİMOĞLU**

Mart 2021

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AĞ TABANLI SALDIRI TESPİT SİSTEMLERİNDE TOPLULUK ÖĞRENME YÖNTEMLERİNİN KARŞILAŞTIRMALI PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

Anıl KURT

Enstitü Anabilim Dalı : BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

Bu tez 03.03.2021 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Anıl KURT

03.03.2021

TEŞEKKÜR

Yüksek lisans eğitimim boyunca bilgi ve deneyimlerinden yararlandığım, bu tez çalışmasında da bilgi ve yardımlarını esirgemeyen değerli danışman hocam Prof. Dr. İsmail Hakkı CEDİMOĞLU'na teşekkürlerimi ve saygılarımı sunarım.

Bu teze desteği ve sağladığı katkılardan dolayı Burak BULDU ve Arş. Gör. Murat VARUL'a teşekkür ederim. Bu tez çalışmasına sağladığı katkılardan dolayı İrem FIRAT'a teşekkür ederim. Tüm hayatım boyunca bana maddi manevi destek olan eğitim hayatına devam etmemi sağlayan aileme sonsuz sevgilerimi ve şükranlarımı sunarım. Son olarak bu süreçte manevi desteğini esirgemeyen ve hep yanımda olduklarını hissettiren tüm arkadaşlarıma şükranlarımı sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

ABD	: Amerika Birleşik Devletleri
ADFA	: Avusturalya Savunma Kuvvetleri Akademisi
AUC	: Area Under Curve (Eğri Altında Kalan Alan)
CART	: Classification And Regression Trees
CPU	: Merkezî İşlem Birimi (Central Process Unit)
DNN	: Derin Sinir Ağları (Deep Neural Network)
DN	: Doğru Negatif (True Negative)
DoS	: Hizmet Engelleme Saldırısı (Denial of Service)
DP	: Doğru Pozitif (True Positive)
DRF	: Dağıtılmış Rastgele Orman
DVM	: Destek Vektör Makineleri (Support Vector Machine)
GBM	: Gradient Boosting Machine
GPU	: Grafik İşlemci Birimi (Graphics Processing Unit)
HIDS	: Host-Based Intrusion Detection System
IG	: Bilgi Kazancı (Information Gain)
KNN	: K-En Yakın Komşu (K-Nearest Neighbors)
MIT	: Massachusetts Institute of Technology
NIDS	: Network-Based Intrusion Detection System
OCSVM	: One-Class Support Vector Machine
OOB	: Out-of-Bag
RAM	: Rastgele Erişimli Hafıza (Random Access Memory)
RNN	: Tekrarlayan Sinir Ağları (Recurrent Neural Network)
ROC	: İşlem Karakteristik Eğrisi (Receiver Operating Characteristic)
R2L	: Yönetici Hesabı ile Yerel Oturum Açma Saldırısı (Remote to Local Attack)

TCP	: Transmission Control Protocol
U2R	: Kullanıcı Hesabının Yönetici Hesabına Yükseltilmesi Saldırısı (User to Root Attack)
YN	: Yanlış Negatif (False Negative)
YP	: Yanlış Pozitif (False Positive)
YSA	: Yapay Sinir Ağları

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: CatBoost, NSL-KDD, Saldırı Tespit Sistemleri, Makine Öğrenmesi, Sınıflandırma Algoritmaları

Teknolojinin gelişmesine paralel olarak internet kullanımı büyük bir artış göstermektedir. İnternetin bu denli yaygın kullanımı büyük miktarda verinin üretilmesine ve böylece bilgi güvenliği unsurlarını tehdit eden ağ saldırılarının daha da yaygınlaşmasına neden olmuştur. Bu bilgi güvenliği unsurlarının tehditlere karşı güvenliğinin sağlanması amacıyla birtakım uygulamalar geliştirilmekte ve kullanılmaktadır. Bu uygulamalardan biri olan saldırı tespit sistemlerinin son yıllarda makine öğrenmesi, derin öğrenme gibi yöntemlerle geliştirilmesi araştırmacılar tarafından ilgi görmektedir. Bu çalışmada, makine öğrenmesi yöntemiyle topluluk öğrenme algoritmaları kullanılarak bir saldırı tespit sistemi modeli önerilmiştir. Bu modelde, belirlenen algoritmaların performansı NSL-KDD veri kümesi üzerindeki KDDTest+ veri kümesini kullanarak test edilmiştir. Buna ek olarak, KDDTrain+ eğitim veri setinde bilgi kazancı yöntemi ile öznitelik seçimi yapılmış, bu algoritmaların KDDTest+ veri seti üzerindeki performansları da incelenmiştir. Bu modelde kullanılmak üzere, henüz literatürde az sayıda çalışılmış bir topluluk öğrenme algoritması olan, CatBoost algoritması seçilmiştir. Ardından, CatBoost algoritmasının performansı, diğer topluluk öğrenme algoritmalarından Random Forest ve AdaBoost algoritmalarının performanslarıyla karşılaştırılmıştır. Bu uygulamada deneysel ortam Python programlama dili, Scikit-learn ve CatBoost kütüphaneleri kullanılarak oluşturulmuştur. Deneysel sonuçlar incelendiğinde, önerilen model ile CatBoost algoritmasının performansı bilgi kazancı yöntemi ile seçilen öznitelikler kullanılarak KDDTest+ veri kümesi üzerinde %79,43 doğruluk (accuracy), %68,44 kesinlik (precision), %96,95 duyarlılık (recall), %80,24 f-ölçütü (f-measure) ve eğri altındaki alan (AUC) değeri 0,9678 olarak elde edilmiştir. Bu ampirik çalışmada, CatBoost algoritmasının hem öznitelik seçimi yapılan hem de öznitelik seçimi yapılmayan veriler üzerinde diğer algoritmalara göre daha iyi saldırı tespiti performansı gösterdiği anlaşılmıştır.

COMPARATIVE PERFORMANCE ANALYSIS OF ENSEMBLE LEARNING METHODS IN NETWORK-BASED INTRUSION DETECTION SYSTEMS

SUMMARY

Keyword: CatBoost, NSL-KDD, Intrusion Detection Systems, Machine Learning, Classification Algorithms

The internet usage has increased significantly in parallel with the development of technology. Such widespread use of the Internet has led to the production of large amounts of data and thus, cause more to the spread of network attacks that threaten information security elements. In order to ensure the security of these information security elements against threats, some methods are developed and used. The fact that the intrusion detection system as one of these methods has been developed using machine learning and deep learning algorithms gathers interest among the researchers. In this paper, a model of intrusion detection system is proposed by using ensemble learning algorithms in machine learning method. In this model, the performance of the designated algorithms is tested using the KDDTest + dataset on the NSL-KDD dataset. In addition, feature selection was made with the information gain method in the KDDTrain + training data set, the performances on the KDDTest+ dataset of these algorithms were also analysed. CatBoost algorithm, an ensemble learning algorithm that has newly been limited number of studies in the literature, has been chosen to be used in this model. After, the performance of the CatBoost algorithm was compared with the performances of Random Forest algorithm and AdaBoost algorithm of other ensemble learning algorithms. In this paper, the experimental environment has been generated by Python programming language, Scikit-learn, and CatBoost libraries. In the analysis of the experiment's results, the performance of CatBoost algorithm with the suggested model that used features selected by the information gain method has produced %79,43 accuracy, %68,44 precision, %96,95 recall, %80,24 f-measure and area under curve value is 0.9678 on KDDTest+ data set. In this empirical study, it was arrived that the CatBoost algorithm has better performance of intrusion detection compared to other algorithms on both executed feature selection data and unexecuted feature selection data.

BÖLÜM 1. GİRİŞ

İnternetin yaygınlaşması ve buna bağlı olarak her geçen gün katlanarak artan internete bağlı cihaz sayısı büyük bir artış göstermektedir. Dünya çapında internetin bu denli yaygın kullanımı e-postalar ve sosyal ağlar gibi platformlarında gelişmesine büyük katkı sağlamaktadır. Bunun sonucu olarak üretilen veri miktarlarındaki artış büyük veri adı verilen bir kavramın ortaya çıkmasına ön ayak olmuştur (Khammassi ve Krichen, 2017). Khammassi ve Krichen görüşlerindeki gibi, Domo tarafından her yıl yayınlanan analizlere bakıldığında bahsedilenleri kanıtlar nitelikte dakika başına üretilen veri miktarlarının korkutucu boyutları gözler önüne serilmektedir (Domo, 2019).

İnternet teknolojilerinin getirdiği birtakım avantajlar firmaların bu teknolojiyi benimsemesine, kullanıcılarına daha iyi bir deneyim sunma ve kendilerini dünyaya tanıtmalarının önünü açmaktadır. İnternetin bu denli gelişmesi buna benzer birtakım faydalar sağladığı gibi beraberinde ağ güvenlik tehditlerinin de ortaya çıkmasına neden olmaktadır. Bu durum veri kayıplarına ve ekonomik kayıplara neden olabilmektedir (Verma ve ark., 2018; Devan ve Khare, 2020).

Ağ güvenliğini tehdit eden saldırıların yaygınlaşması verilerin bütünlüğünün, gizliliğinin ve erişilebilirliğinin korunmasını zorunlu hale getirmektedir. Ağ güvenliğini tehdit eden saldırıların önlenmesi amacıyla birtakım araçlar da geliştirilmiştir. Bunlardan bir tanesi olan güvenlik duvarı sistemleri belirli bir koruma seviyesi sağlamakla birlikte gelişen saldırıları tespit etmek ve engellemek için ağ paketlerini analiz etme kapasiteleri sınırlıdır (Husain ve ark., 2019). Ağ güvenliğinin sağlanması adına geliştirilmiş bir başka araç olan saldırı tespit sistemleri, mevcut saldırıların tespitinde ve yeni geliştirilen saldırıların tespit edilmesinde önemli bir rol oynamaktadır.

Saldırı tespit sistemleri, tespit modellerine göre ikiye ayrılmaktadır. Bunlar: imza tabanlı tespit modeli ve anormallik tabanlı tespit modelidir. Bu iki saldırı tespit modeli de her türden saldırı için etkin olmamakla birlikte her birinin güçlü veya zayıf olduğu yönler bulunmaktadır (Alamiedy ve ark., 2019).

İmza tabanlı saldırı tespit sistemi bilinen saldırıları tespit etmede başarılı olmakla birlikte bilinmeyen saldırıların tespitinde zayıf kalmaktadır. Buna karşın anormallik tabanlı saldırı tespit sistemi bilinmeyen saldırıların tespitinde etkin bir rol oynamakla birlikte yüksek bir yanlış alarm oranına sahip olmaktadır. Bu sebeplerden ötürü bazı saldırı tespit sistemleri hem imza tabanlı saldırı tespiti hem de anormallik tabanlı saldırı tespiti tekniklerini bütünleştiren karma bir yaklaşım kullanmaktadır (Khammassi ve Krichen, 2017; El Boujnouni ve Jedra, 2018).

Literatür incelendiğinde, saldırı tespit sistemlerindeki araştırmaların çoğu anormallik tespitine yoğunlaşmıştır (Khammassi ve Krichen, 2017). Bununla birlikte makine öğrenmesini kullanan ağ tabanlı saldırıların tespitinde anormallik tabanlı tespit yoğun ilgi görmektedir. Random Forest, Destek Vektör Makineleri (DVM) ve XGBoost gibi sınıflandırma algoritmaları saldırı tespit sistemleri geliştirilmek için kullanılmıştır (Pattawaro ve Polprasert, 2018).

Ağ tabanlı saldırı tespit sistemlerinin tespit etmeye çalıştığı saldırılar literatürde Hizmet Engelleme Saldırısı (DoS), Yönetici Hesabı ile Yerel Oturum Açma Saldırısı (R2L), Bilgi Tarama Saldırısı (ing. Probe Attack) ve Kullanıcı Hesabının Yönetici Hesabına Yükseltilmesi Saldırısı (U2R) olmak üzere dört farklı kategoriye ayrılmaktadır (Rai, 2020). Sınıflandırma problemleri kategori sayılarına göre ikili sınıflandırma ve çok sınıflı sınıflandırma olmak üzere ikiye ayrılabilmektedir. Ağ tabanlı saldırı tespiti alanında ikili sınıflandırma, normal trafiğin anormal trafikten ayırt edilmesi şeklinde gerçekleştirilir. Çoklu sınıflandırma probleminde ise saldırı tespitinin yapıldığı yöneticiye bildirilmekle birlikte yapılan saldırının kategorisi hakkında daha ayrıntılı bilgi sağlamaktadır (Jin ve ark., 2020). Literatür incelendiğinde çoğu çalışmanın ikili sınıflandırmaya odaklandığı görülmektedir. Ayrıca ağ trafiği içerisinde farklı kategorilerde bulunan saldırı örneklerinin nadir gerçekleşmesinden

ötürü ağ trafiği veri kümeleri saldırı sınıflarına göre genellikle bir dengesizlik barındırmaktadır. Bu durum makine öğrenmesi modelleri tarafından belli bir kategoriye ait örneklerin yanlış sınıflandırılmasına neden olabilmektedir (Jin ve ark., 2020).

Bunların dışında ağ tabanlı verilerin yüksek boyutlu olması sebebiyle makine öğrenmesi tekniklerinin uygulanmasında hem daha fazla zaman almakta hem de modelin sınıflandırma başarısına olumsuz katkıları olmaktadır. Bu sebepten ötürü boyut azaltma tekniklerinden öznitelik seçimi, tüm veri kümesini temsil eden öznitelik alt kümesinin seçilmesinde kullanılmaktadır. Bu sayede veri kümesindeki gereksiz öznitelikler ortadan kaldırılarak algoritmaların hızlanmasına ve sınıflandırma başarısına katkı sağlamaktadır (Khammassi ve Krichen, 2017).

Bu çalışmada, ağ tabanlı saldırın tespit edilip sınıflandırılması için makine öğrenmesi tabanlı bir model önerilmiştir. Önerilen modelde saldırıların sınıflandırması için literatürde çok az çalışmada kullanılan topluluk öğrenme yöntemlerinden CatBoost algoritması kullanılmıştır. Bu algoritmanın kullanılmasının nedeni, yeni geliştirilen bir algoritma olmasından dolayı saldırı tespiti alanında çok az çalışılmış olması ve literatürde kullanılan birçok topluluk öğrenmesi altındaki güçlendirme algoritmalarından daha hızlı olduğunun belirtilmesidir (Dorogush, Ershov ve Gulin, 2018). Akademik araştırmalardaki saldırı tespit sistemlerinin performanslarının değerlendirilmesinde KDD Cup 99 (University of California, 1999) veri kümesi sıkça kullanılmaktadır. Ancak bu veri kümesinin olumsuz yönleri bulunmaktadır. Bu yönlerin ortadan kaldırılıp geliştirilmiş bir versiyonu olan ve yine sıkça kullanılan NSL-KDD veri kümesi bu çalışmada tercih edilmiştir. Veri kümesindeki boyutun azaltılmasında El Boujnouni ve Jedra'dan (2018) esinlenilerek öznitelik seçim yöntemlerinden bilgi kazancı yöntemi tercih edilmiştir. Bu yöntem saldırı tespit sisteminin doğruluğunu arttırmakla kalmaz aynı zamanda algoritmanın çalışma süresini de azaltmaktadır.

Bu çalışmada, CatBoost algoritması modelde gerçekleştirildikten sonra algoritmanın performansı ayrıca Random Forest ve AdaBoost algoritmalarının performansıyla da

karşılaştırılacaktır. Performans karşılaştırmasında doğruluk (accuracy), kesinlik (precision), duyarlılık (recall), f-ölçütü (f-measure), hız ve işlem karakteristik eğrisi (ROC) gibi standart sınıflandırma performans metrikleri kullanılarak analiz edilecektir.

Araştırmanın amacı: Bu çalışmanın temel amacı; makine öğrenimindeki sınıflandırma algoritmalarının saldırı tespit sistemlerinde kullanılmasına yol göstermesi istenmektedir. Bu amaca göre 3 temel hedef belirlenmiştir. Bunlardan birincisi literatürdeki yeni bir sınıflandırma algoritması olarak CatBoost makine öğrenimi yöntemi ile saldırı tespit sistemi geliştirmektir. İkincisi bu geliştirilen saldırı tespit sisteminin başarısını literatürde sıkça kullanılan Random Forest ve AdaBoost algoritmaları ile oluşturulan saldırı tespit sistemleriyle karşılaştırmaktır. Son olarak üçüncüsü CatBoost algoritmasının performansını literatürde yapılmış benzer çalışmalarla karşılaştırarak ampirik bir çalışma gerçekleştirmektir.

Araştırmanın önemi: Bu çalışma ağ tabanlı saldırı tespiti sistemlerindeki sınıflandırma yöntemi olarak CatBoost algoritmasının kullanıldığı literatürdeki öncü çalışmalardan bir tanesidir. Ayrıca bu çalışmada boyut azaltma yöntemlerinden öznitelik seçimi olarak bilgi kazancı yöntemiyle diğer araştırmalardan ayrışmaktadır. Bu sayede hem bilgi kazancı yöntemi kullanılarak öznitelik seçiminin yapıldığı hem de sınıflandırma algoritması olarak CatBoost algoritmasının kullanıldığı literatürdeki özgün ve yenilikçi çalışmalardan biri olması beklenmektedir.

Araştırmanın yöntemi ve kısıtları: Bu tez çalışmasında ağ tabanlı saldırı tespit sistemleri için bir sınıflandırma temelli makine öğrenmesi modeli önerilmiştir. Bu kapsamda önerilen model kullanılarak algoritmaların performansının ölçülmesinde ve karşılaştırılmasında NSL-KDD veri kümesi tercih edilmiştir. Bu veri kümesi üzerinde bir takım veri ön işlem adımları gerçekleştirilmiştir. Uygulamada varsayılan hiper parametreler kullanılmıştır. Deneysel ortamın hazırlanmasında Python programlama dili, CatBoost kütüphanesi ve Scikit-learn kütüphanesi kullanılmıştır.

Araştırmanın organizasyonu: Çalışma şu şekilde düzenlenmiştir. Bölüm 1'de yapılan çalışma hakkında literatürden genel bilgiler verilmiştir. Bölüm 2'de saldırı tespit sistemleri, saldırı tespit sistemlerinin geliştirilmesinde kullanılan makine öğrenmesinin detayları ve öznitelik seçimi yöntemlerinden bahsedilmiştir. Bölüm 3'te önerilen saldırı tespit sisteminin ayrıntıları ve bulgularına yer verilmiştir. Bölüm 4'te çalışmanın sonuçlarının değerlendirilmesinden ve gelecekteki çalışma önerilerinden bahsedilmiştir.

BÖLÜM 2. SALDIRI TESPİT SİSTEMLERİNDE MAKİNE ÖĞRENİMİ

Bu bölümde saldırı tespit sistemlerinin ne olduğu hakkında genel bir giriş yapılmakta ve saldırı tespit sistemlerinin sınıflarından bahsedilmektedir. Bununla birlikte literatürdeki makine öğrenme algoritmaları kullanılarak geliştirilen saldırı tespit sistemlerine değinilmiştir. Daha sonra makine öğrenmesi tanıtılmış ve türlerinden bahsedilmiştir. Son olarak saldırı tespit sisteminde öznitelik seçiminin ne olduğundan bahsedilmektedir.

2.1. Saldırı Tespit Sistemleri

Anderson saldırıyı bilgiye erişme veya bilgiyi değiştirme ya da bir sistemin işleyişini bozma amacı taşıyan her türlü girişim olarak tanımlamaktadır (Anderson, 1980). Son yıllarda internet ve bilgisayar sistemlerinin kullanımındaki artış, beraberinde çok sayıda güvenlik sorunu ortaya çıkartmıştır. Bununla ilgili her yıl düzenli olarak rapor ortaya koyan Symantec Global İnternet Güvenliği Tehdit Raporu'nda saldırılar her yıl bir önceki yıla göre giderek artış göstermektedir (Brigid ve ark., 2019). İnternet kullanımının yaygınlaşması ve sosyal medya platformlarının artan kullanımı ağ saldırı türlerinin çoğalmasına yeni saldırı parametrelerinin ortaya çıkmasına neden olmuştur. Bu saldırıların önlenebilmesi amacıyla güvenlik duvarları, saldırı önleme sistemleri ve saldırı tespit sistemleri gibi çevre güvenlik cihazlarının kullanım alanları genişlemektedir (Vasudevan ve Selvakumar, 2016). Bunlardan saldırı tespit sistemleri bilgi güvenliğinin sağlanmasında etkin bir rol oynamaktadır. Buna bağlı olarak saldırı tespit sistemleri günümüz kurum, kuruluş ve organizasyonlarının ayrılmaz bir parçası haline gelmiştir. Şekil 2.1.'de Anderson'un saldırı tanımında bahsettiği bilgi güvenliğinin 3 temel unsuru gösterilmektedir (Anderson, 1980). Bunlardan kısaca aşağıda bahsedilmiştir;

- Gizlilik: Bilginin yetkisiz kişiler tarafından ele geçirilmesinin engellenmesi amaçlanmaktadır.
- Bütünlük: Bilginin olduğu gibi korunması ve yetkisiz kişilerce değiştirilmesinin engellenmesi amaçlanmaktadır.
- Erişilebilirlik: Bilginin her zaman erişilebilir olmasının sağlanmasını amaçlamaktadır.

Şekil 2.1. Bilgi güvenliği unsurları

Saldırı tespit sistemi, kısaca bilgisayar sistemindeki bilgi güvenliği unsurlarını ihlal etmeye yönelik girişimleri tanımlayabilen ve bunları sistem yöneticisine bildiren yazılım veya donanım tabanlı sistemler şeklinde tanımlanabilmektedir (Malhotra, Bali ve Paliwal, 2017; Khraisat ve ark., 2020). Şekil 2.2.'de saldırı tespit sistemlerinin bulundukları yere göre ve tespit modeline göre genel olarak iki sınıfa ayrıldığı görülmektedir.

Şekil 2.2. Saldırı tespit sistemlerinin sınıflandırılması (Alamiedy ve ark., 2019)

2.1.1. Bulundukları yere göre saldırı tespit sistemleri

2.1.1.1. Host tabanlı saldırı tespit sistemi (HIDS)

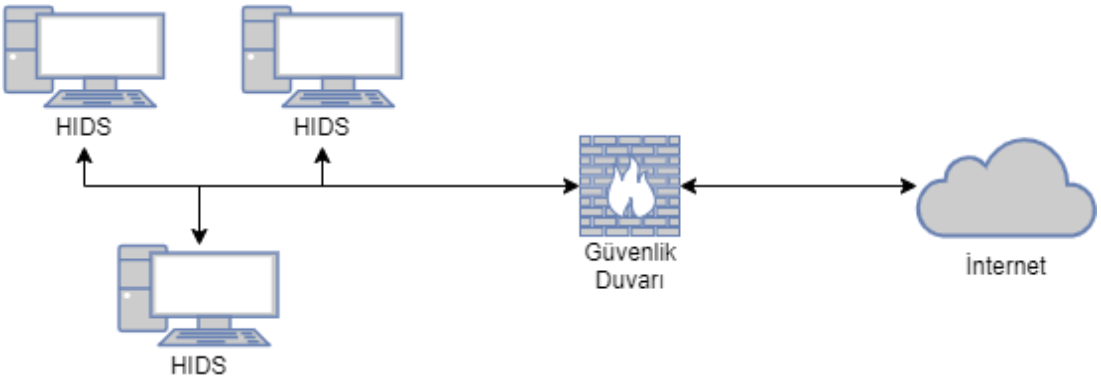
Host (Ana Bilgisayar) tabanlı saldırı tespit sistemi bilgisayarlar üzerine kurulu bir yazılım şeklindedir. Saldırıları tespit edebilmek için sistem üzerindeki oturum açma isteklerini, Log (Günlük) dosyaları üstündeki değişiklikleri, komut dizinlerini ve sistem çağrıları gibi sistemin tamamını takip edip analiz etmektedir. Herhangi bir tutarsızlıkla karşılaşılması sonucunda sistem yöneticisi bilgilendirilir (Malhotra, Bali ve Paliwal, 2017; Noorbehbahani ve ark., 2017; Verma ve ark., 2018; Alamiedy ve ark., 2019; Bhati ve ark., 2020; Bhattacharya ve ark., 2020). Şekil 2.3.'de Host tabanlı saldırı tespit sisteminin şeması gösterilmiştir. Burada HIDS bahsedildiği üzere bilgisayarların üzerine kurulmuştur. HIDS'in avantajları ve dezavantajlarından aşağıda bahsedilmiştir (Bhati ve ark., 2020).

Avantajları:

- İç güvenlik sorunlarını önler.
- Bilgisayarlar üzerinde kurulu olduğu için güvenlik duvarlarından daha etkindir.

Dezavantajları:

- Bilgisayarın veya sistemin performansını düşürür.
- Çevrimdışı çalıştığında güvenlik ihlali olabilir.
- Yönetilmesi zayıftır.



Şekil 2.3. Host tabanlı saldırı tespit sistemi

2.1.1.2. Ağ tabanlı saldırı tespit sistemi (NIDS)

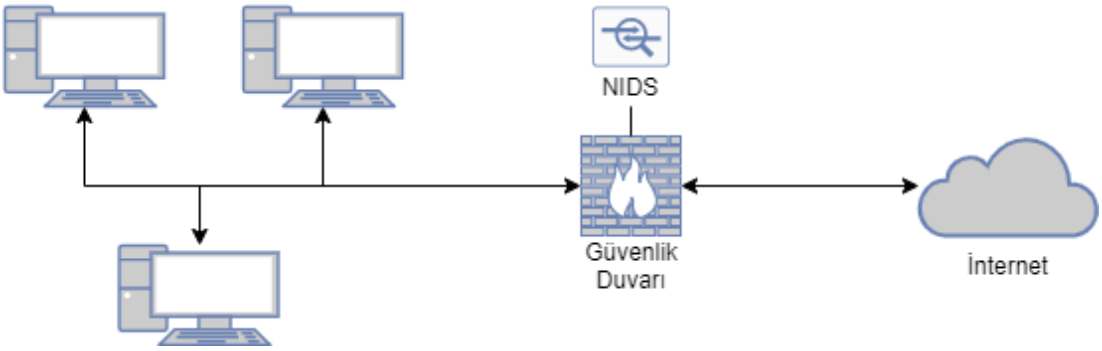
Ağ tabanlı saldırı tespit sistemi, sistemin ağ trafiğini takip edebileceği merkezi bir konuma yerleştirilmektedir. Ağ tabanlı saldırı tespit sistemi ağ üzerinden gelen/giden paketleri takip edip inceler ve paketler üzerindeki şüpheli kalıpları bulmaya çalışmaktadır. Ayrıca, yerel trafik üzerindeki saldırılar hakkında bilgi sağlamaktadır. Herhangi bir şüpheli etkinlik gerçekleştiğinde sistem yöneticisi bilgilendirilir. Ağ tabanlı saldırıların tespiti, izlenen ağda meydana gelen olayların analizine dayanılarak gerçekleştirilmektedir. NIDS için en önemli zorluk büyük hacimli trafiği kabul edilebilir bir doğrulukla analiz edebilmesidir (El Boujnouni ve Jedra, 2018; Pattawaro ve Polprasert, 2018; Verma ve ark., 2018; Bhati ve ark., 2020). Şekil 2.4.'de Ağ tabanlı saldırı tespit sisteminin şeması gösterilmiştir. Burada NIDS ağ trafiğinin sisteme giriş yaptığı merkezi bir konum olan güvenlik duvarının üstüne yerleştirilmiştir. NIDS'in avantajları ve dezavantajından aşağıda bahsedilmiştir (Bhati ve ark., 2020).

Avantajları:

- Çok sayıda bilgisayarı idare eden verimli tasarımı bulunmaktadır.
- NIDS pasif modda çalışabilmektedir. Bu nedenle daha fazla bilgisayar eklemek herhangi bir soruna neden olmamaktadır.
- NIDS, doğrudan saldırılarla çökertilememektedir.

Dezavantajları:

- NIDS, şifreli saldırılarda başarısız olmaktadır.



Şekil 2.4. Ağ tabanlı saldırı tespit sistemi

2.1.2. Tespit modeline göre saldırı tespit sistemleri

2.1.2.1. İmza tabanlı tespit

Ticari olarak kullanılan çoğu saldırı tespit sistemlerinde bulunan imza tabanlı saldırı tespit yöntemi, önceden tanımlanmış saldırı kalıpları ve saldırı imzalarını kullanarak bilinen saldırıları tespit etmeyi amaçlamaktadır (El Boujnouni ve Jedra, 2018). İmza tabanlı saldırı tespit sistemi aynı zamanda bilgi tabanlı tespit sistemi ve yanlış kullanım tespit sistemi olarak da adlandırılmaktadır (Khraisat ve ark., 2020). İmza tabanlı saldırı tespit sistemi saldırıların imzalarından oluşan kapsamlı bir veri tabanına sahiptir. Veri tabanındaki imzaların her biri daha önce meydana gelmiş saldırı türlerine karşılık gelen bir dizi kural içermektedir (Khafajeh, 2020). Şüpheli trafiğin veya etkinliğin imzası, imza tabanlı saldırı tespit sistemindeki imza veri tabanı ile karşılaştırılır. Eğer şüpheli trafik veya etkinlik imza veri tabanındaki kayıtlardan biriyle eşleşiyorsa saldırı olarak nitelendirilir (Verma ve ark., 2018). Bu yöntemle ilgili en büyük sorun daha önce imza veri tabanında kayıtlı olmayan yeni bir saldırı olduğunda, bu saldırıyı tespit etme olasılığının çok düşük olmasından kaynaklanmaktadır (Bhati ve ark., 2020). Bir diğer sorun, imza veri tabanındaki kayıtlar ne kadar fazla ise büyük hacimli verileri analiz etmek ve işlemek bir o kadar uzun sürmektedir (Khraisat ve ark., 2020). İmza tabanlı saldırı tespit sistemi veri tabanında kayıtlı saldırı imzaları sayesinde saldırıları kesin bir şekilde tanımlayabildiği için yüksek doğruluk oranına sahiptir. Bununla birlikte yanlış alarm sayısı çok düşüktür (Khraisat ve ark., 2020). Bu sistemin El

Boujnouni ve Jedra (2018) bazı avantaj ve dezavantajlarından bahsetmektedir (El Boujnouni ve Jedra, 2018).

Avantajları:

- Saldırıları yüksek yanlış alarmlar oluşturmadan tespit edebilmektedir.

Dezavantajları:

- Bilinen saldırılar ile sınırlıdır.
- İmza veri tabanının yeni çıkan saldırıları öğrenmesi için sık sık güncellenmesi gereklidir.
- Büyük hacimli verileri işleyip analiz etmek uzun sürmektedir (Khraisat ve ark., 2020).

2.1.2.2. Anormallik tabanlı tespit

Anormallik tabanlı saldırı tespit sistemlerinde, sistemin normal davranışını açıklayan bir model öğrenilir ve model üzerindeki herhangi bir sapma ayırt edilerek saldırı amaçlı etkinlikler tespit edilebilmektedir (Hua, 2020). Anormallik tabanlı saldırı tespit sistemleri bilinen veya bilinmeyen saldırıları algılayabilme potansiyeline sahiptir. Anormallik tabanlı saldırı tespit sistemi, sistemin normal davranışını öğrenebilmek için makine öğrenimi ve veri madenciliği tekniklerini kullanmaktadır. Öğrenilen sistemin normal davranışında meydana gelen sapmalar anormallik olarak algılanmaktadır. Anormallik tabanlı saldırı tespit sistemlerinin en büyük zorluğu normal davranışların kesin olarak tanımlanmasının mümkün olmamasıdır. Eğitilen modelin daha önce hiç tanımadığı normal bir davranış anormal olarak değerlendirilebilir. Bu durum anormallik tabanlı saldırı tespit sistemlerinin yanlış alarm sayısının yüksek olmasına sebep olmaktadır (El Boujnouni ve Jedra, 2018; Verma ve ark., 2018; Tang, Luktarhan ve Zhao, 2020).

Avantajları (El Boujnouni ve Jedra, 2018):

- Bilinen ve bilinmeyen saldırıları algılayabilme potansiyeline sahiptir.

Dezavantajları (El Boujnouni ve Jedra, 2018):

- Yeni normal bir davranış anormallik olarak algılanabilmektedir. Bu durum yanlış alarm sayısının yükselmesine sebep olmaktadır.

2.1.3. Literatürdeki saldırı tespit sistemleri çalışmaları

Rai (2020) tarafından topluluk öğrenme yöntemleri kullanılarak geliştirilen saldırı tespit sistemlerinin performansları incelenmiştir. Bu doğrultuda NSL-KDD veri kümesi üzerinde öznitelik seçimi olarak genetik algoritma kullanılmış. Daha sonra ilgili veri kümesi üzerinde hem öznitelik seçimi yapılmış hali hem de yapılmamış hali Dağıtılmış Rastgele Orman (DRF), Gradient Boosting Machine (GBM), XGBoost topluluk öğrenme yöntemleri ve Derin Sinir Ağları (DNN)'nın ikili sınıflandırma üzerindeki performansları karşılaştırılmıştır. Bu yöntemlerden topluluk tabanlı yöntemlerin diğer makine öğrenmesi yöntemlerinden daha iyi performans gösterebileceği sonucuna ulaşılmıştır (Rai, 2020).

Khammassi ve Krichen (2017) tarafından KDD Cup 99 ve UNSW-NB15 veri kümeleri üzerinde öznitelik seçimi olarak genetik algoritma ve lojistik regresyonun birleşiminden oluşan bir yaklaşım kullanılmış. Daha sonra ilgili öznitelikler seçilerek karar ağacı temelli C4.5, Random Forest ve NBTree algoritmaları üzerinde hem ikili hem de çoklu sınıflandırma performansları karşılaştırılmış. Ayrıca literatürdeki başka çalışmalarla da önerilen modelin sonuçları karşılaştırılmıştır. KDD Cup 99 veri kümesi üzerinde 18 öznitelik seçilerek %99,90 doğruluk (accuracy) oranı elde edilirken UNSW-NB15 veri kümesi üzerinde 20 öznitelik seçilerek %81,42 doğruluk (accuracy) oranı elde edilmiştir (Khammassi ve Krichen, 2017).

Pattawaro ve Polprasert (2018) tarafından NSL-KDD veri kümesi üzerinde öznitelik seçimi yöntemi olarak öznitelik oranı (ing. Attribute Ratio) yöntemi kullanılmıştır. Daha sonra seçilen öznitelikler K-Means kümeleme ve XGBoost algoritması kombinasyonu şeklinde önerilen model üzerinde kullanılarak önerilen modelin ikili sınıflandırma performansı ortaya konmaktadır. Önerilen modelin tekrarlayan sinir ağları (RNN) tabanlı derin sinir ağı (DNN) ve ağaç tabanlı sınıflandırıcılar kullanılarak

elde edilen sonuçlardan daha iyi performans gösterdiği belirtilmiştir. Önerilen model NSL-KDD veri kümesi üzerinde %84,41 doğruluk (accuracy) oranı elde etmiştir (Pattawaro ve Polprasert, 2018).

El Boujnouni ve Jedra (2018) tarafından NSL-KDD veri kümesi üzerinde öznitelik seçimi yöntemi olarak bilgi kazancı yöntemi kullanılmıştır. Daha sonra ilgili öznitelikler seçilerek Destek Vektör Makineleri (DVM)'nden esinlenen yeni bir sınıflandırma modeli olan Destek Vektör Alan Açıklaması algoritmasının SSPV-SVDD adı verilen geliştirilmiş bir versiyonu kullanılarak çoklu sınıflandırma performansı ortaya konulmaktadır. Önerilen model NSL-KDD veri kümesi üzerinde %77,5 doğruluk (accuracy) oranı elde etmiştir (El Boujnouni ve Jedra, 2018).

Jin ve arkadaşları (2020) tarafından KDD Cup 99 veri kümesi üzerinde K-En Yakın Komşular (KNN) ve CatBoost algoritmalarının avantajlarının birleştirildiği karma bir model önerilmiştir. Önerilen modelin çoklu sınıflandırma performansı ortaya konulmaktadır. Önerilen model duyarlılık performans metriği açısından DoS (%99.4), Probing (%94.14), U2R (%50) ve R2L (%55,23) doğruluk (accuracy) oranı sonuçlarını elde etmiştir (Jin ve ark., 2020).

Khraisat ve arkadaşları (2020) tarafından NSL-KDD ve Avusturalya Savunma Kuvvetleri Akademisi (ADFA) veri kümesi üzerinde C5 karar ağacı algoritması ve Tek Sınıf Destek Vektör Makinesinin birleştirildiği karma bir model önerilmiştir. Önerilen modelin ikili sınıflandırma performansı ortaya konmaktadır. Önerilen model NSL-KDD veri kümesi üzerinde %83,24 doğruluk (accuracy) oranı elde etmiştir (Khraisat ve ark., 2020).

2.2. Makine Öğrenmesi

Makine öğrenmesinden bahsedebilmek için öncelikle öğrenme kavramının tanımına bakılmalıdır. Öğrenme kavramının farklı şekillerde tanımının bulunmasıyla birlikte genellikle Simon tarafından önerilen tanım üzerinde değişiklikler yapılmaktadır. Simon'a göre öğrenme, zaman içinde yeni bilgilerin keşfedilmesi yoluyla

davranışların iyileştirilmesi süreci olarak tanımlanmaktadır (Simon, 1983; Öztemel, 2012).

Makine öğrenmesi ise mevcut verilerin kullanılarak bir model oluşturulmasına ve bu model kullanılarak yeni veriler üzerinde tahminde bulunulmasıyla ilgilenen bir yapay zekâ alanı olarak adlandırılır (Bilgin, 2018; Vieira, Lopez Pinaya ve Mechelli, 2019). Bir başka deyişle makine öğrenmesi, bir olay ile ilgili verilerdeki bilgi ve tecrübeyi öğrenerek gelecekte karşılaşılacak benzeri olaylar hakkında kararlar verebilmek ve problemlere çözümler üretebilmek olarak tanımlanmaktadır (Öztemel, 2012). Makine öğrenmesi kavramı ilk olarak Samuel tarafından ortaya atılmıştır. Samuel makine öğrenmesini, bilgisayarın yeniden programlanmasına gerek duyulmadan görev yapmasını sağlayan bilim olarak tanımlamaktadır (Samuel, 1959; Bilgin, 2018).

Makine öğrenimi geleceğe yönelik bilgi tahmininde kullanıldığında buna tahmin edici model denilmektedir. Tahmin edici model eğitim ve test aşamasından oluşmaktadır. Öncelikle mevcut veriler eğitim ve test olmak üzere ikiye bölünür. Eğitim aşamasında etiketli eğitim verisi kullanılarak bir model oluşturulur. Daha sonra oluşturulan model kullanılarak test verileri üzerinde performansı değerlendirilir. Başarılı bulunan model sisteme entegre edilir (Bilgin, 2018).

Makine öğrenmesi görevinin gerçekleştirilebilmesi için 5 temel uygulama adımı bulunmaktadır (Lantz, 2013; Bilgin, 2018). Bunlar;

- Veri toplama: Veriler birçok yerde (veri tabanı, web sayfası, görüntüler, vb.) ve formatta bulunabilmektedir. Bu verilerin analize uygun bir formatta toplanması gerekmektedir. Bu veriler model oluşturulması için kullanılan eğitim verileri olmaktadır (Lantz, 2013; Bilgin, 2018).

- Verilerin keşfedilmesi/hazırlanması: Makine öğrenmesi projelerinin kalitesi, büyük ölçüde kullanılan verilerin kalitesine bağlıdır. Makine öğrenimindeki çabanın %80'i bu adıma harcanmaktadır. Bu adım model eğitiminde

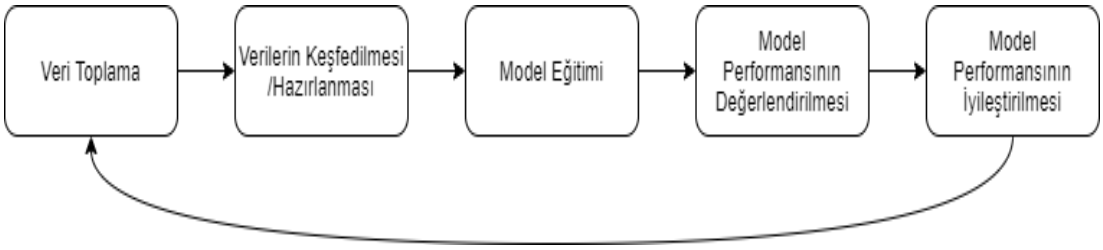
kullanılmadan önce veriler üzerinde gerçekleştirilen işlemleri kapsamaktadır (Lantz, 2013; Bilgin, 2018).

- Model eğitimi: Bu adım verilerden ne öğrenmek istediğimizle bağlantılı olarak uygun bir algoritmanın seçimini ve ilgili modelin oluşturulması sürecini içermektedir. Oluşturulan model elimizdeki verilerin temsilini içermektedir (Lantz, 2013; Bilgin, 2018).

- Modelin performansının değerlendirilmesi: Bu adım oluşturulan makine öğrenimi modelinin benzer veriler karşısında nasıl performans gösterdiğinin değerlendirildiği aşamadır. Burada bir test veri kümesi kullanılarak modelin doğruluğu ve hedeflenen uygulamaya özgü performans ölçümleri gerçekleştirilmektedir (Lantz, 2013).

- Modelin performansının iyileştirilmesi: Bu adım kullanılan modelin performansını arttırmak için daha gelişmiş stratejiler kullanmayı gerektirmektedir. Bunlar arasında farklı bir model seçmek, mevcut verilere ek veriler eklemek ve verilerin keşfedilmesi/hazırlanması aşamasında olduğu gibi daha fazla ek hazırlık çalışmaları yapılması gerekebilmektedir (Lantz, 2013; Bilgin, 2018).

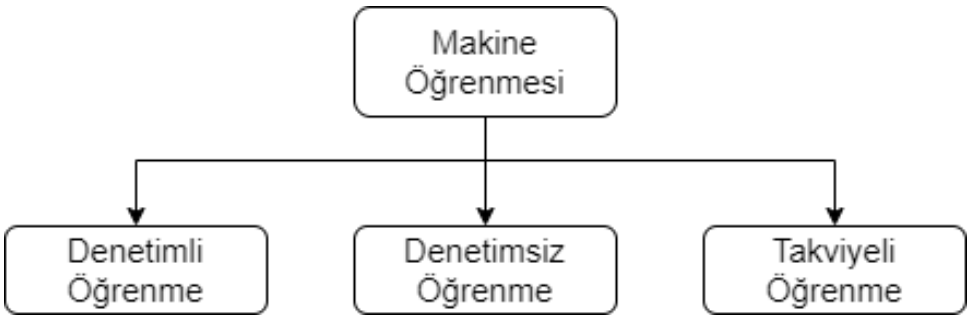
Şekil 2.5.'de makine öğrenmesi uygulama adımlarının şeması gösterilmiştir.



Şekil 2.5. Makine öğrenmesi uygulama adımları

2.2.1. Makine öğrenmesi türleri

Makine öğrenmesi, verilerdeki ilgili kalıpları öğrenmeyi ve ardından bunları tahmin yapabilmek için kullanmayı içermektedir. Bunu başarmanın birden fazla yolu vardır ve bu durum aralarında seçim yapılabilecek çok sayıda makine öğrenimi algoritmasıyla sonuçlanmaktadır (Vieira, Lopez Pinaya ve Mechelli, 2019). Araştırma problemiyle uyumlu olan makine öğrenmesi yöntemini seçebilmek için Şekil 2.6.'da makine öğrenmesi yöntemleri 3 farklı türe ayrılmaktadır.



Şekil 2.6. Makine öğrenmesi türleri

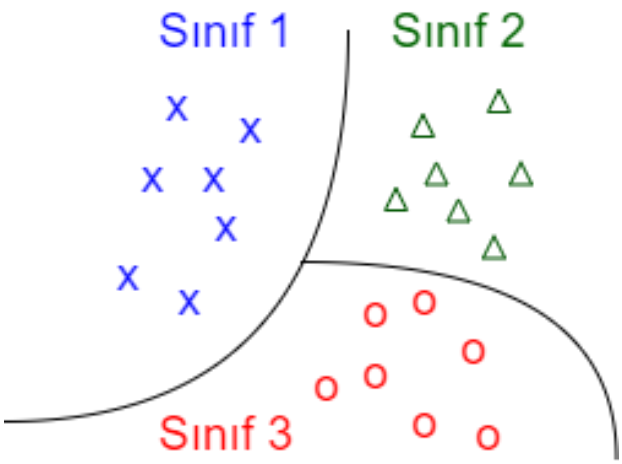
2.2.1.1. Denetimli öğrenme

Denetimli öğrenme yöntemi, algoritmaları eğitilirken kullanılan veri kümesi hem girdi verilerinin değerlerini hem de ilgili çıktı verilerinin hedef değerlerini içermektedir (Vieira, Lopez Pinaya ve Mechelli, 2019; Subasi, 2020). Bu öğrenme yöntemindeki temel amaç girdi verilerine karşılık gelen çıktı yani etiketli verilerin ilişkisi öğrenilerek modelin eğitilmesidir. Bu eğitilen model daha sonra model eğitimi sürecinde kullanılmamış olan girdi verisi örnek kümesinin çıktısını yani etiketini tahmin etmek için kullanılmaktadır (Subasi, 2020). Kullanılan modelin performansı tahmin edilen çıktı değerlerinin gerçek çıktı değerleriyle karşılaştırılması sonucu ölçülür. Bu tür öğrenmenin denetimli olarak adlandırılmasının nedeni, eğitilen algoritmanın çıktı değerlerinin ne olması gerektiğine dair önceden bilgisi olmasıdır. Tahmin edilecek çıktı verilerinin kategorik veya sürekli değişken olmalarına bağlı olarak denetimli öğrenme yöntemleri sırasıyla sınıflandırma veya regresyon problemi olmak üzere iki ana sınıftan oluşmaktadır (Vieira, Lopez Pinaya ve Mechelli, 2019; Subasi, 2020). En

bilinen denetimli öğrenme algoritmaları Lojistik Regresyon, Yapay Sinir Ağları (YSA), Destek Vektör Makineleri (DVM) ve Karar Ağaçları algoritmaları olarak ifade edilebilmektedir (Bilgin, 2018). Denetimli öğrenme anormallik tespiti, yüz tanıma, hava durumu tahmini vb. alanlarda kullanılmaktadır (Subasi, 2020).

2.2.1.1.1. Sınıflandırma algoritmaları

Sınıflandırma algoritmaları daha önce bahsedildiği gibi denetimli öğrenme yönteminin bir alt alanıdır (Subasi, 2020). Sınıflandırma algoritmaları; etiketleri yani sınıfları (hedef değerleri) belirli olan verileri kullanarak yeni etiketli olmayan verilerin sınıflarının bulunması olarak tanımlanabilir. Sınıflandırma algoritmalarının kullanılmasıyla oluşturulan modelin ürettiği her bir çıktı, sınıf (ing. class) olarak adlandırılmaktadır. Sınıflandırma algoritmalarının kullanılmasındaki amaç, etiketli veriler kullanılarak eğitilen modelin, etiketleri olmayan ve model tarafından daha önceden görülmemiş veriler üzerinde yüksek doğrulukla sınıflandırma yapılmasının sağlanmasıdır. Sınıflandırma algoritmaları bazı diğer kaynaklarda örüntü tanıma olarak da isimlendirilmektedir. Sınıflandırma algoritmaları makine öğrenmesinin en bilinen problem türlerinden birisidir. En bilinen sınıflandırma algoritmaları Destek Vektör Makineleri (DVM), K-En Yakın Komşu (KNN), Yapay Sinir Ağları (YSA) ve Karar Ağaçları gibi algoritmalardır (Bilgin, 2018). Sınıflandırma algoritmaları anormallik tespiti, tıbbi tanıların teşhisinde, yüz tanıma sistemlerinde ve kredi derecelendirilmesi gibi daha pek çok alanda kullanılmaktadır (Alpaydın, 2014b). Şekil 2.7.'de sınıflandırma örneği gösterilmektedir.



Şekil 2.7. Sınıflandırma örneği (Sugiyama, 2016b)

2.2.1.2. Denetimsiz öğrenme

Denetimsiz öğrenme yöntemi algoritmaları eğitilirken kullanılan veri kümesi sadece girdi verilerinin değerlerini içermektedir (Vieira, Lopez Pinaya ve Mechelli, 2019; Subasi, 2020). Çıktı yani etiketli verilerin değerleri model eğitilirken bulunmamaktadır. Denetimsiz öğrenme yöntemindeki amaç girdi verileri arasındaki ilişkilerin, düzenlerin ve benzerliklerin bulunmasıdır (Alpaydın, 2014b; Bilgin, 2018; Subasi, 2020). Girdi verilerinde belirli kalıpların diğerlerinden daha sık meydana geldiği bir yapı vardır. Denetimsiz öğrenmede genel olarak bu yapıların ne olup ne olmadığı görülmek istenmektedir. İstatistik alanında buna yoğunluk tahmini denilmektedir (Alpaydın, 2014b). Denetimsiz öğrenme yöntemleri kümeleme ve boyut azaltma olmak üzere iki sınıfa ayrılmaktadır (Vieira, Lopez Pinaya ve Mechelli, 2019; Subasi, 2020). En bilinen denetimsiz öğrenme algoritmaları K-Means ve Temel bileşen analizi algoritmaları olarak ifade edilebilmektedir (Vieira, Lopez Pinaya ve Mechelli, 2019). Denetimsiz öğrenme müşteri ilişkileri yönetimi, görüntü sıkıştırma vb. alanlarda kullanılmaktadır (Alpaydın, 2014b).

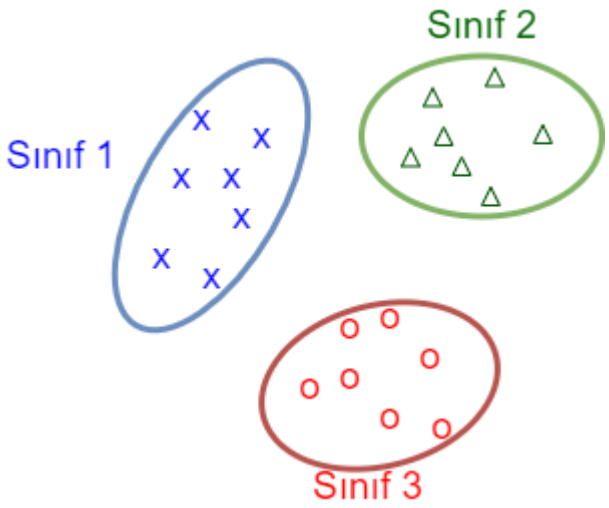
2.2.1.2.1. Kümeleme algoritmaları

Kümeleme algoritması daha önce bahsedildiği gibi denetimsiz öğrenme yönteminin bir alt alanıdır. Kümeleme algoritmaları denetimsiz öğrenme yöntemlerinin en bilinen

sınıfıdır (Kononenko ve Kukar, 2007a). Kümeleme algoritmaları; etiketleri yani sınıfları (hedef değerleri) olmayan girdi verilerinden çıkarımlar yapmak veya gizli kalıpları, benzerlikleri bulmak olarak tanımlanmaktadır. Kümeleme algoritmaları birbirine benzeyen veri örneklerini veya birbirinden çok farklı olan veri örneklerini farklı kümeler halinde gruplandırmaktadır (Shobha ve Rangaswamy, 2018). Kümeleme algoritmalarının iki temel kuralı bulunmaktadır (Bilgin, 2018).

- Küme içerisindeki verilerin benzerliklerinin maksimum yapılması,
- Kümeler arasındaki verilerin benzerliklerinin ise minimum yapılmasıdır.

Kümeleme algoritmalarında girdi verileri kümelere ayrıştırılırken verilerin birbirlerine olan uzaklıkları dikkate alınmaktadır. Girdi verilerinin kesikli-sürekli veya nominal-ordinal olmasına göre uzaklıkların ölçülmesinde kullanılan metriklerde değişiklik göstermektedir. Kümeleme algoritmalarında en sık kullanılan ölçüm metrikleri: Öklit uzaklığı, Manhattan uzaklığı, Pearson uzaklığı, Spearsman sıralama korelasyonu ve Jaccard benzerlik ölçütü olmaktadır. En bilinen kümeleme algoritmaları K-Means ve Hiyerarşik kümeleme gibi algoritmalardır (Bilgin, 2018). Kümeleme algoritmaları tıbbi tanıların teşhisinde ve konuşma tanıma gibi alanlarda kullanılmaktadır (Alpaydın, 2014b; Vieira, Lopez Pinaya ve Mechelli, 2019). Şekil 2.8.'de kümeleme örneği gösterilmektedir.



Şekil 2.8. Kümeleme örneği (Sugiyama, 2016b)

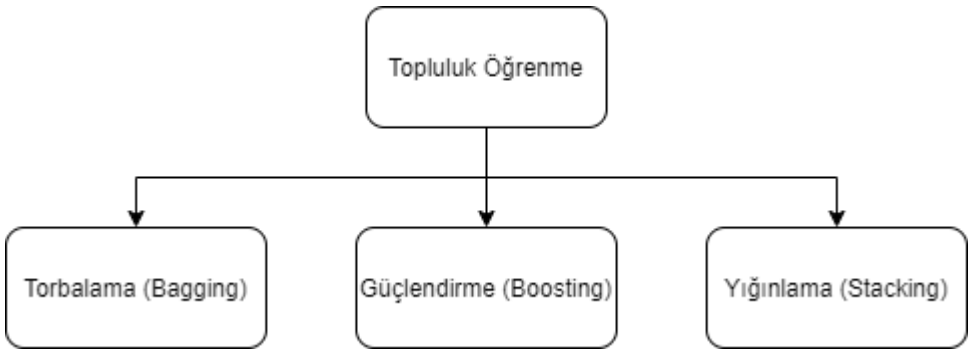
2.2.1.3. Takviyeli öğrenme

Takviyeli öğrenme yöntemi algoritmaları eğitilirken kullanılan veri kümesi denetimsiz öğrenmede olduğu gibi sadece girdi verilerinin değerlerini içermektedir (Yılmaz, 2019). Takviyeli öğrenme yönteminde oluşturulan modelin çıktılarını kontrol eden modele geri dönüş sağlayan bir danışman bulunmaktadır (Bilgin, 2018; Yılmaz, 2019). Danışman modele verilen girdi verilerinin beklenen çıktısı ile model tarafından tahmin edilen çıktı değerlerini karşılaştırarak modelin ürettiği çıktı sonucunun doğru ya da yanlış olduğunu modele bildirmektedir. Bu esnada danışman olması gereken çıktıları modele tanıtmaz. Modelin ürettiği çıktıya sadece doğru ya da yanlış geri bildiriminde bulunmaktadır (Hamzaçebi, 2011; Yılmaz, 2019). Modelin hesapladığı ya da oluşturduğu tahmin sonucunun doğru olması durumunda model ödüllendirilirken yanlış olması durumunda model cezalandırılır. Model danışmandan gelen geri dönüşleri göz önünde bulundurarak en uygun adımları atıp öğrenmeye devam etmektedir. Bu şekildeki öğrenmeye takviyeli öğrenme yöntemi denilmektedir (Bilgin, 2018; Shobha ve Rangaswamy, 2018; Yılmaz, 2019). En bilinen takviyeli öğrenme algoritması Q-Learning algoritmasıdır (Bilgin, 2018). Takviyeli öğrenme oyun oynama (satranç ve tavla), çeşitli optimizasyon problemleri ve konuşulan kelimeyi tanıma vb. alanlarda kullanılmaktadır (Kononenko ve Kukar, 2007a; Shobha ve Rangaswamy, 2018).

2.2.2. Makine öğrenmesi süreci: topluluk öğrenme

Topluluk öğrenme yöntemleri Nilsson tarafından denetimli öğrenme yöntemi altındaki sınıflandırma algoritmaları için önerildi (Nilsson, 1965; Yang, 2017). Genellikle makine öğrenmesi algoritmaları ile tek bir sınıflandırıcı kullanılarak oluşturulan modellerden daha iyi performans elde eden topluluk öğrenme yöntemleri, makine öğreniminde en etkili yaklaşımlardandır (Bilgin, 2018; Subasi, 2020). Topluluk öğrenme yöntemleri birçok sınıflandırıcı algoritmanın tahminlerini bir araya getirerek sınıflandırma doğruluğunu geliştirmeyi amaçlamaktadır. Topluluk öğrenme yöntemleri kullanılarak etiketli eğitim verilerinden birkaç tane temel sınıflandırıcılı eğitilmiş modeller oluşturulmaktadır. Daha sonra test verileri üzerinde eğitilen

modeller tarafından yapılan tahminlere göre oylama (sınıf değerleri oluşturmada) veya ortalama (regresyon değerlerini tahmin etmede) alınarak sınıflandırma yapılmaktadır (Talia, Trunfio ve Marozzo, 2016). Topluluk öğrenme yöntemleri hem regresyon hem de sınıflandırma problemlerine uygulanabilmektedir (Sugiyama, 2016a). Üç popüler topluluk öğrenme yöntemi bulunmaktadır. Bunlar: Torbalama (Bagging), Güçlendirme (Boosting) ve Yığınlama (Stacking) olarak kategorize edilmektedir (Khraisat ve ark., 2020). Şekil 2.9.'da üç popüler topluluk öğrenme yöntemi gösterilmiştir.



Şekil 2.9. Topluluk öğrenme kategorileri

2.2.2.1. Torbalama (Bagging)

Torbalama (Bagging) ifadesi, önyükleme kümelenmesinin (ing. bootstrap aggregating) kısaltmasıdır (Sugiyama, 2016a). Torbalama tekniği Breiman tarafından sınıflandırma ve regresyon algoritmalarının tahminlerinin doğruluğunu arttırmak için geliştirilmiş bir tekniktir (Breiman, 1996; Sutton, 2005; Bilgin, 2018). Torbalama tekniğinde etiketli eğitim kümesi içerisinden rastgele örnek seçimleri yapılarak birkaç küçük veri kümesine bölünür ve her alt veri kümesi üzerinde modeller eğitilir. Daha sonra eğitilen her bir model test veri kümesi içindeki her bir örnek ile test edilir ve nihai sonucu elde etmek için regresyon problemlerinde modellerin çıktılarının ortalama değerleri veya medyan değerleri hesaplanırken, sınıflandırma problemlerinde modellerin çıktıları basit oylama yoluyla birleştirilmektedir (Sutton, 2005; Witten, Frank ve Hall, 2011; Alpaydın, 2014a; Subasi, 2020). Torbalama yönteminde orijinal veri kümesinden rastgele örnek seçilerek oluşturulan küçük veri kümelerinde bazı örneklerden birden fazla bulunurken, bazı örneklerden hiç bulunmaması mümkün olabilmektedir (Alpaydın, 2014a). Sınıflandırma algoritmaları kullanılarak yapılan torbalama

yöntemlerinde daha fazla oy dikkate alındıkça daha güvenilir sonuçlar ortaya çıkmaktadır (Witten, Frank ve Hall, 2011). Torbalama yönteminde modellerin verdiği kararlar eşit derecede önemlidir (Witten, Frank ve Hall, 2011; Talia, Trunfio ve Marozzo, 2016).

Torbalama yöntemi, sınıflandırma algoritmaları kullanılarak eğitilen modellerdeki eğitim veri kümesine aşırı uyum gösterme (ing. overfitting) problemlerinin engellenmesine ve modellerin tahminlerindeki önyargı (sapma) ve varyans hatalarını azaltmaktadır (Bilgin, 2018; Subasi, 2020). Torbalama yöntemi, orijinal veri kümesinin farklı rastgele seçilen örneklerle oluşturulan küçük veri kümeleri üzerinde çeşitli modeller ile eğitilen son derece başarılı bir topluluk öğrenme yöntemidir. Torbalama yöntemindeki orijinal veri kümesinden rastgele örnekler seçilerek oluşturulan küçük veri kümeleri arasındaki farklılıklar topluluk öğrenme yöntemlerindeki modeller arasında çeşitlilik sağlayacağından modellerin başarısında tam olarak ihtiyaç duyulan durumu ortaya çıkarmaktadır (Subasi, 2020). Torbalama yöntemi, orijinal veri kümesindeki kayıp verilerin olduğu durumlarda da sınıflandırma ve regresyon algoritmalarındaki başarıları oldukça yüksektir (Bilgin, 2018).

Torbalama yöntemi sinir sistemi, karar ağaçları ve regresyon ağaçları gibi algoritmalarda çok iyi çalışmaktadır (Subasi, 2020). Karar ağaçları algoritmalarında uygulanan torbalama yöntemi Random Forest algoritması olarak adlandırılmaktadır (Sugiyama, 2016a).

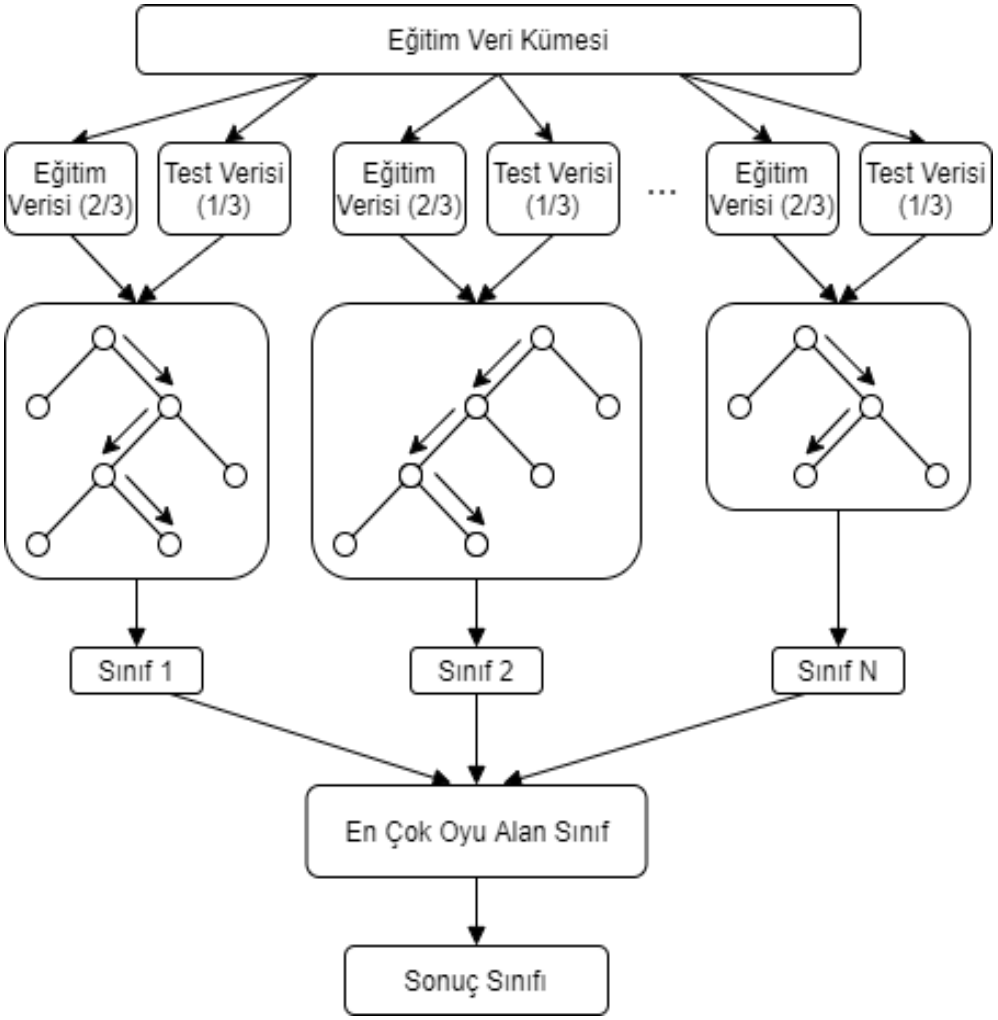
2.2.2.1.1. Random Forest

Random Forest algoritması Breiman tarafından 2001 yılında tanıtılmıştır (Breiman, 2001). Random Forest algoritması başlangıçta sınıflandırma problemi için geliştirildi, ancak regresyon problemleri içinde kullanılabilmektedir (Kononenko ve Kukar, 2007b; Horning, 2010). Random Forest algoritması torbalama yöntemini kullanan topluluk öğrenme içerisinde yer almaktadır (Bilgin, 2018). Random Forest algoritması birbirinden bağımsız olarak eğitilen çok sayıda karar ağacı algoritması modellerinden oluşmaktadır (Lindner, 2017). Random Forest algoritmasının başarısının anahtarı

oluşturulan her bir karar ağacının nasıl oluşturulduğudur (Horning, 2010). Random Forest algoritması oluşturulan her bir karar ağacı modelinin eğitilmesinde orijinal etiketli eğitim veri kümesi içerisinden üçte ikisi oranında rastgele örnekler seçerek orijinal etiketli eğitim veri kümesiyle aynı boyutta başka veri kümeleri kullanmaktadır. Veri kümelerinin oluşturulmasında kullanılmayan üçte birlik kısım oluşturulan eğitim modellerinin doğruluğunu test etmek için kullanılmaktadır. Test için kullanılan bu üçte birlik kısma "out-of-bag (OOB)" denilmektedir. Random Forest algoritması ile oluşturulan karar ağaçları modellerinde düğümler dallara ayrılırken her bir karar ağacına özgü veri kümesi içerisindeki tüm öznitelikleri kullanarak en iyi dal ayrımı gerçekleştirmek yerine, her bir düğümde rastgele olarak seçilen öznitelikler arasından en iyisi kullanılarak düğüm dallarına ayırılmaktadır. Şayet karar ağaçları oluşturulurken düğümlerin dallara ayrılmasında veri kümesi içerisindeki tüm öznitelikler kullanılsaydı bu durum oluşturulan bütün karar ağaçları modellerinin aynı olmasına neden olmaktadır. Random Forest algoritması kullanılarak eğitilen modeller test veri kümesi içindeki her bir örnek ile test edilir ve nihai sonucu elde etmek için sınıflandırma problemlerinde modellerin çıktılarından en çok oyu alan sınıf seçilmektedir. Diğer bir deyişle 500 tane eğitilmiş karar ağacı modeli varsa ve bunlardan 350 tanesi bir test örneği için saldırı şeklinde sınıflandırıp 150 tanesi saldırı değil şeklinde sınıflandırıyorsa, bu test örneğinin sınıf çıktısı saldırı şeklinde olmaktadır. Regresyon problemlerinde ise modellerin çıktılarının ortalama değeri alınmaktadır (Kononenko ve Kukar, 2007b; Horning, 2010; Bilgin, 2018). Random Forest algoritması eğitim veri kümesinde eğitilen eğitim modelleri üzerinde önemli etkiye sahip olan önemli özniteliklerin belirlenmesine yardımcı olmaktadır. Bu sayede önemsiz gözüken öznitelikler eğitim veri kümesinden çıkartılarak başarı oranında bir artış sağlanabilmektedir (Horning, 2010).

Random Forest algoritması ile üretilen karar ağaçlarının budama gerçekleştirilmeden mümkün olan en geniş ölçüde büyümesine izin verilmektedir (Breiman ve Cutler, 2004; Bilgin, 2018). Random Forest algoritmasının güçlü bir hale gelmesi için algoritma içerisinde oluşturulan modeller arasındaki ilişkinin azaltılması gerekmektedir (Breiman ve Cutler, 2004; Subasi, 2020). Random Forest algoritması veri kümesine karşı aşırı uyum (ing. overfitting) göstermemektedir (Breiman, 2001).

Ayrıca hızlı ve istenildiği kadar karar ağacıyla çalışabilmektedir (Breiman ve Cutler, 2004; Bilgin, 2018). Random Forest algoritması eğitim ve test veri kümesindeki eksik veriler üzerinde de çalışabilmektedir (Horning, 2010). Şekil 2.10.'da Random Forest algoritmasının sınıflandırma yöntemi şeması gösterilmektedir.



Şekil 2.10. Random Forest sınıflandırma yöntemi şeması (Bilgin, 2018)

2.2.2.2. Güçlendirme (Boosting)

Güçlendirme (Boosting) tekniği ilk olarak Schapire tarafından torbalama yönteminde olduğu gibi sınıflandırma ve regresyon algoritmalarının tahminlerinin doğruluğunu arttırmak için güçlü bir model oluşturmak amacıyla önerilmiştir (Schapire, 1990; Sutton, 2005; Alpaydın, 2014a; Subasi, 2020). Güçlendirme yönteminde öncelikle etiketli eğitim kümesindeki her bir örneğin ilk modelin eğitilmesinde kullanılan

rastgele seçilerek oluşturulan küçük eğitim veri kümesinin içerisinde bulunabilmesi için her bir örneğe eşit ağırlık değeri atanmaktadır. İlk model eğitildikten sonra modelin eğitim veri kümesinde hatalı tahmin ettiği örnekler için ağırlık değeri arttırılırken doğru tahmin ettiği örnekler için ağırlık değerleri azaltılmaktadır. Daha sonra ikinci model eğitilirken ilk model sonucu ağırlıkları artan örneklerin seçilme olasılığının arttığı ilk modelin eğitildiğinden farklı bir etiketli eğitim veri kümesi ile eğitilmektedir. Tüm modeller eğitildikten sonra test veri kümesi içindeki her bir örnek ile test edilir ve nihai sonucu elde etmek için regresyon problemlerinde modellerin çıktılarının ortalama değeri alınarak hesaplanırken, sınıflandırma problemlerinde modellerin çıktıları basit oylama yoluyla birleştirilmektedir. Güçlendirme yöntemi modelleri eğitirken önceki modelin yapmış olduğu hatalı örneklere ağırlık vererek bir sonraki modelin bu örnekleri doğru tahmin etmesini sağlamaya çalışmaktadır (Sutton, 2005; Witten, Frank ve Hall, 2011; Subasi, 2020). Güçlendirme yöntemi ile genel bir model oluşturulurken eğitim aşamasında oluşturulan her yeni model daha önce inşa edilmiş eğitim modellerinin performansından etkilenmektedir (Witten, Frank ve Hall, 2011). Güçlendirme yönteminde kullanılan veri seti yinelemeli olacak şekilde bir önceki modelin başarısına bağlı olarak bir sonraki modelin eğitilmesinde kullanılmak üzere güncellenmektedir (Subasi, 2020).

Güçlendirme yöntemi genellikle zayıf öğrencilere uygulanmaktadır (örneğin, karar ağacı gibi basit bir sınıflandırıcı algoritma) ancak torbalama için durum böyle değildir (Sutton, 2005). Güçlendirme yönteminde karar ağaçları ile sadece bir veya iki seviye büyüyen ağaçlar olan karar kütükleri kullanılmaktadır. Bu karar kütükleri kullanılarak modellerin eğitim veri kümesine aşırı uyum göstermesinin (ing. overfitting) önüne geçilmektedir (Alpaydın, 2014a). Güçlendirme yöntemi ve torbalama yönteminin kullanıldığı literatürdeki birçok çalışmada güçlendirme yönteminin torbalama yönteminden daha üstün başarı sağladığı gösterilmektedir (Kononenko ve Kukar, 2007b; Bilgin, 2018). Güçlendirme yönteminde modellerin paralel olarak eğitildikleri torbalama yönteminin aksine modeller sırayla eğitilmektedir (Rahmat ve ark., 2019).

Güçlendirme yöntemini kullanan AdaBoost (ing. Adaptive Boost) ve CatBoost gibi algoritmalar geliştirilmiştir (Rahmat ve ark., 2019).

2.2.2.2.1. CatBoost

CatBoost ifadesi, kategorik özellik desteği ile gradyan güçlendirmenin kısaltmasıdır. CatBoost algoritması Dorogush ve arkadaşları tarafından 2018 yılında tanıtılmıştır (Dorogush, Ershov ve Gulin, 2018). Topluluk öğrenme yöntemlerinin altında güçlendirme tekniğinin kullanıldığı bir algoritmadır. CatBoost algoritması hem sınıflandırma problemlerine hem de regresyon problemlerine uygulanabilmektedir (CatBoost, 2020). Bununla birlikte CatBoost algoritması gradyan güçlendirme makine öğrenimi tekniğini kullanmaktadır. Bu teknik genellikle sabit boyuttaki bir dizi zayıf karar ağaçları sınıflandırıcılarını birleştirerek güçlü modeller geliştirilmesine yardımcı olmaktadır. Gradyan güçlendirme tekniğinin kullanılması veri kümesine karşı aşırı uyum (ing. overfitting) göstermesi problemine neden olabilmektedir. Bu sorun CatBoost algoritması tarafından ele alınmış ve aşırı uyum (ing. overfitting) gösterme probleminin önüne geçilmesi amacıyla gradyan güçlendirme tekniğinin geliştirilmiş bir versiyonu kullanılmıştır (Dorogush, Ershov ve Gulin, 2018).

CatBoost algoritması kullanılarak model geliştirilirken veri kümesi içerisindeki tüm özniteliklerin sayısal değerler içermesine ihtiyaç duyulmamaktadır. Yani CatBoost algoritmasının veri kümesi içerisindeki kategorik veriler üzerinde hiçbir ön işlem yapılmasına ihtiyaç duymadan çalışabilme özelliği bulunmaktadır. CatBoost algoritması istenildiğinde CPU dışında GPU üzerinde çalışabilmektedir. Ayrıca veri kümesi üzerindeki eksik verilerle de çalışabilmektedir (Dorogush, Ershov ve Gulin, 2018).

2.2.2.2.2. AdaBoost

AdaBoost ifadesi, adaptif güçlendirmenin (ing. Adaptive Boosting) kısaltmasıdır. AdaBoost algoritması Freund ve Schapire tarafından 1996 yılında tanıtılmıştır. Ayrıca topluluk öğrenme yönteminin altında güçlendirme tekniğinin kullanıldığı ilk pratik algoritmadır (Freund ve Schapire, 1997; Verma ve ark., 2018). AdaBoost algoritması hem sınıflandırma problemlerinde hem de regresyon problemlerinde kullanılabilmektedir (Polikar, 2006). Bir dizi karar kütükleri gibi zayıf sınıflandırıcıları

güçlü bir sınıflandırıcıya dönüştürmektedir (Verma ve ark., 2018). AdaBoost algoritması oluşturulacak model sayısının belirtildiği ardışık modellerden oluşmaktadır. AdaBoost algoritması kullanılırken öncelikle etiketli eğitim kümesindeki her bir örneğin ilk modelin eğitilmesinde kullanılacak rastgele seçilerek oluşturulan küçük eğitim veri kümesinin içerisinde bulunabilmeleri için her bir örneğe eşit ağırlık değeri atanmaktadır. İlk model eğitildikten sonra modelin eğitim veri kümesinde hatalı tahmin ettiği örnekler için ağırlık değeri arttırılırken doğru tahmin ettiği örnekler için ağırlık değerleri azaltılmaktadır. Daha sonra ikinci model eğitilirken ilk model sonucu ağırlıkları artan örneklerin seçilme olasılığının arttığı ilk modelin eğitildiğinden farklı bir etiketli eğitim veri kümesi ile eğitilmektedir. Önceden belirtilen en son model de eğitildikten sonra AdaBoost algoritması etiketlenmemiş test örnekleri üzerinde sınıflandırma problemini çözmeye hazırdır. AdaBoost algoritması nihai sınıfların belirlenmesinde ağırlıklı çoğunluk oyu olarak adlandırılan oylama yöntemini kullanmaktadır. Bu sayede eğitim sırasında iyi performans gösteren modeller, diğer modellerden daha yüksek oy ağırlıkları ile ödüllendirilmektedir. Tüm bunların sonucunda test aşamasında kullanılan tüm modellerden en yüksek toplam oyu alan sınıf ilgili örneğin sınıf değeri olarak atanmaktadır (Polikar, 2006).

AdaBoost algoritması eğitim veri kümesinde eğitilen eğitim modelleri üzerinde önemli etkiye sahip olan önemli özniteliklerin belirlenmesine yardımcı olabilmektedir (Wu ve ark., 2008). AdaBoost algoritmasını kullanan birçok çalışma AdaBoost algoritmasının eğitim veri kümesine aşırı uyum göstermediğini (ing. overfitting) göstermektedir (Wu ve ark., 2008).

2.2.2.3. Yığınlama (Stacking)

Yığınlama (Stacking) ifadesi, yığınlanmış genellemenin (ing. stacked generalization) kısaltmasıdır (Witten, Frank ve Hall, 2011). Yığınlama yöntemi ilk olarak Wolpert tarafından 1992 yılında ortaya atılmıştır (Wolpert, 1992). Yığınlama yöntemi, torbalama ve güçlendirme yöntemlerinden farklı olarak aynı türdeki modelleri örneğin karar ağaçlarının sonuçlarını birleştirmek için kullanılmamaktadır. Bunun yerine farklı algoritmalar tarafından oluşturulan modelleri birleştirmek için kullanılmaktadır

(Witten, Frank ve Hall, 2011). Yığınlama yöntemi kullanılan farklı algoritmaların modellerini birleştiren genel öğrenici diye adlandırılan başka bir modeldir ve eğitilmektedir. Yığınlama yönteminin amacı farklı algoritmalar tarafından oluşturulan modellerin tahminlerini nihai sonuca ulaşmak için birleştirmektedir. Yığınlama yönteminde farklı algoritmaların modellerinin eğitilmesinde kullanılan eğitim veri kümesi modeller tarafından ezberlenebileceğinden dolayı genel öğrenici modeli eğitilirken farklı bir veri kümesi kullanılarak eğitim yapılmalıdır. Yığınlama yönteminde etiketli eğitim veri kümesi öncelikle iki kısma ayrılmaktadır. Birinci kısım veri kümesi farklı modellerin eğitilmesinde kullanıldıktan sonra çıkan tahmin sonuçları ikinci veri kümesi ile birleştirilerek genel öğrenici modelinin eğitilmesinde kullanılmaktadır. Bunun sonucunda yığınlama yöntemini kullanan farklı modellerin tek bir çatı altında birleştirildiği eğitilmiş bir model elde edilmektedir (Witten, Frank ve Hall, 2011; Alpaydın, 2013; Khraisat ve ark., 2020).

2.3. Saldırı Tespit Sisteminde Öznitelik Seçimi

Günümüzde büyük veri içerisinden bilgi işlemek ve bilgi çıkarımı yapabilmek için makine öğrenimi ve veri madenciliği yöntemleri sıklıkla tercih edilmektedir. Bununla birlikte bu yöntemleri çok fazla öznitelik ve çok fazla veri barındıran veri kümeleri üzerinde uygulamak zaman alıcı bir süreçtir ve bilginin doğruluğunu etkilemektedir. Ağ tabanlı saldırı tespit sistemlerindeki öznitelik seçimi aşaması, ağ üzerinden gelen büyük veri yığınının idare edilmesi konusunda saldırı tespit sistemlerinde zorlu bir görev olmaktadır. Alakasız öznitelikler algoritmanın sınıflandırma performansını azaltan ve desteklenmeyen bellek gereksinimine neden olmaktadır. Bununla birlikte saldırı tespit sistemi modellerinin eğitim ve test aşamalarında ihtiyaç duyulan hesaplama kaynağını arttırmaktadır (Khammassi ve Krichen, 2017; Siddique ve ark., 2017; Alamiedy ve ark., 2019).

Veri kümelerinde N adet bulunan öznitelik değerlerine yabancı literatürde “Curse of dimensionality” denilmektedir. Bu ifadenin Türkçe karşılığı Ozyer ve arkadaşları (2006) tarafından “boyutsal bela” olarak isimlendirilmiştir (Ozyer, Akbas ve Vural, 2006). Boyutsal bela (ing. curse of dimensionality) problemini çözebilmek ve

algoritmaların sınıflandırma performanslarının iyileştirilmesi amacıyla, aynı zamanda öznitelikleri seçmek, çıkarmak ve oluşturmak için bir ön işleme aşamasına gereksinim duyulmaktadır (Khammassi ve Krichen, 2017).

Saldırı tespit sistemlerinde karşılaşılan veri kümeleri çok fazla veri ve çok fazla öznitelik içermeleriyle bilinmektedir. Bu tür veri kümeleriyle başa çıkabilmek, sınıflandırma doğruluğunu iyileştirmek ve sınıflandırma süresini azaltmak için boyut azaltma yöntemleri kullanılmaktadır. Boyut azaltma yöntemleri iki ana kategoriye ayrılmaktadır. Bunlar: öznitelik dönüşümü ve öznitelik seçimidir (Wu ve ark., 2008; Khammassi ve Krichen, 2017).

- Öznitelik dönüştürme: Mevcut özniteliklerden yeni öznitelikler oluşturarak verilerin boyutsal bela problemini (ing. curse of dimensionality) azaltmaktadır. Bu işlem veri kümesindeki öznitelikleri uygun bir öznitelik uzayına dönüştürmeye çalışan matematiksel yöntemlere dayanmaktadır. Bu işlem sonucunda özniteliklerin orijinal anlamları genellikle kaybolmaktadır.

- Öznitelik seçimi: Yalnızca ilgili ve bilgilendirici özniteliklerin seçilmesini, ilgisiz ve gereksiz olanların veri kümesinden kaldırılmasını amaçlamaktadır. Bu işlem sonucunda öznitelikler orijinal anlamlarını koruyabilmektedirler.

Öznitelik seçimi ile kullanılan algoritmanın performansını iyileştirmek için çok az bilgi sağlayan veya hiç sağlamayan veri kümesindeki ilgisiz ve gereksiz özniteliklerin kaldırılarak boyutsal bela (ing. curse of dimensionality) etkisini azaltmayı amaçlamaktadır. Algoritmanın tahmin performansı verilerin verimli bir şekilde temsil edildiği alt kümenin seçimine bağlı olmaktadır. Bir öznitelik hedef veya sınıfla ilgili yararlı bilgiler sunuyorsa bu öznitelik ilgili olarak kabul edilmektedir (Haq, Onik ve Shah, 2015; Khammassi ve Krichen, 2017).

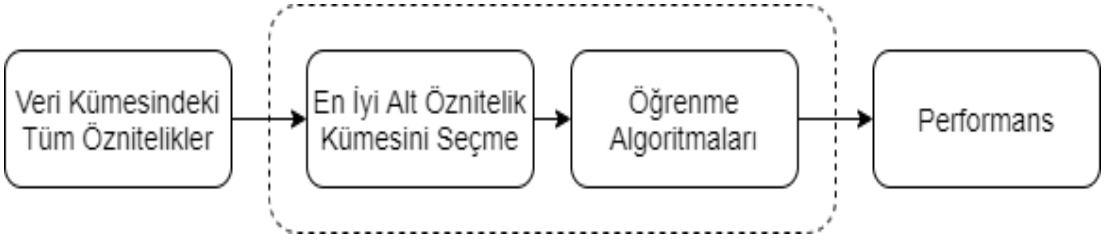
Özellikle sınıflandırma, kümeleme ve regresyon gibi makine öğrenimi problemlerinde kullanılan veri kümelerinin artan boyutları nedeniyle öznitelik seçimine olan ilgi her geçen gün artış göstermektedir. Bunun en temel nedenlerinden bir tanesi öznitelik

seçiminin beraberinde getirdiği faydalardır. Öznitelik seçimi temel olarak model tahmininin iyileştirilmesine, modelin eğitim süresinin kısaltılmasına ve aşırı uyum gösterme (ing. overfitting) problemlerinin önlenmesine yardımcı olmaktadır. Öznitelik seçimi 3 farklı yönteme ayrılmaktadır. Bunlar: filtreleme, sarmal ve gömülü yöntemlerdir (Haq, Onik ve Shah, 2015; Khammassi ve Krichen, 2017).

2.3.1. Öznitelik seçimi yöntemleri

2.3.1.1. Filtreleme yöntemleri

Filtreleme yöntemleri istatistik alanında daha yaygın kullanılmakla birlikte kullanılan makine öğrenme algoritmasından tamamen bağımsız öznitelik seçme yöntemleridir. Filtreleme yöntemleri modelin eğitilmesinde kullanılacak eğitim veri kümesine uygulanmaktadır. Veri kümesinde bulunan özniteliğin alaka düzeyini belirlemek ve bu özniteliği elde tutmaya veya atmaya karar vermek için bir değerlendirme ölçütü ve eşik değeri kullanan öznitelik sıralama tekniklerine dayanmaktadır. İlgili özniteliğin tutulması farklı sınıflar hakkında bilgiler sağlama yeteneğiyle doğru orantılıdır. Filtreleme yöntemleri genellikle diğer yöntemlerden daha yüksek hesaplama verimliliğine sahip olmaktadır (Khammassi ve Krichen, 2017; Alamiedy ve ark., 2019). Şekil 2.11.'de filtreleme yöntemlerinin süreci gösterilmektedir.



Şekil 2.11. Filtreleme yöntemleri süreci (Alamiedy ve ark., 2019)

2.3.1.1.1. Bilgi kazancı (Information gain)

Bilgi kazancı filtreleme yöntemleri içerisinde yer almaktadır. Bilgi kazancı, veri kümesindeki özniteliğin sınıflandırma problemlerindeki sınıf etiketlerini belirlemede ne kadar katkıda bulunduğunu gösteren bir ölçüttür ve 0-1 arasında değer almaktadır.

Bilgi kazancı değerinin hesaplanması entropi kavramına dayanmaktadır. Entropi kavramı, veri kümesi içerisindeki özniteliklerin belirsizlik ve rastgeleliklerinin ölçülmesinde kullanılmaktadır. Veri kümesindeki entropi değeri yüksek olan özniteliklerin sınıflandırma problemlerindeki sınıf etiketlerini belirlemede diğer özniteliklere göre daha çok bilgi içerdiği kabul edilmektedir. Bilgi kazancı yöntemi veri kümesi içerisindeki çok farklı değerlerin bulunduğu öznitelikleri seçme eğilimi göstermektedir (Ünsalan ve Erçil, 1998; Siddique ve ark., 2017; Bilgin, 2018; Kaynar ve ark., 2018). (Denklem 2.1)'de entropi kavramının matematiksel formülü gösterilmektedir.

$$Entropi\ (Y)\ = -\sum_{y \in Y} p(y) * log_2(p(y)) \tag{2.1}$$

(Denklem 2.1)'deki "P(y)" öznitelik içerisindeki sınıfların örnek sayılarının toplam örnek sayısına bölümünü ifade eden olasılığını tanımlamaktadır. Örnek olarak 2 sınıf bulunan, bir sınıf 8 örnek diğer sınıf 2 örnek barındıran toplam 10 örnekli bir veri kümesinin entropi değeri (Denklem 2.2)'deki gibi hesaplanır.

$$\text{Entropi (Y)}\ = -\frac{2}{10} * log_2\left(\frac{2}{10}\right) - \frac{8}{10} * log_2\left(\frac{8}{10}\right) \approx 0{,}72 \tag{2.2}$$

Entropi değeri sınıf özniteliği için hesaplandıktan sonra her bir özniteliğin sınıf özniteliği üzerindeki entropi değeri (Denklem 2.3) kullanılarak hesaplanır.

$$\text{Entropi(Y\X)}\ = -\sum_{x \in X} p(x) * \sum_{y \in Y} p(y\backslash x) * log_2(p(y\backslash x)) \tag{2.3}$$

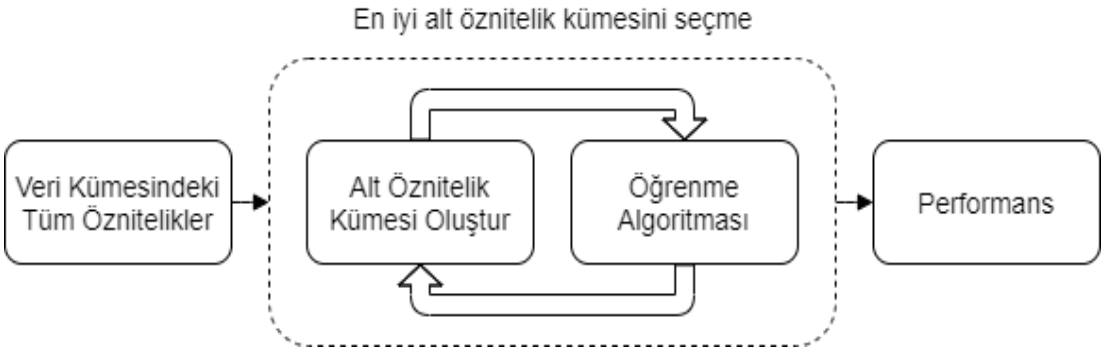
(Denklem 2.3)'teki "p(x)" öznitelik içerisindeki sınıfların örnek sayılarının toplam örnek sayısına bölümünü ifade eden olasılığı tanımlamaktadır. "p(y/x)" ise x özniteliğindeki sınıfın y özniteliği üzerindeki sınıfların örnek sayılarının toplam x sınıfının örnek sayısına bölümünü ifade eden olasılığı tanımlamaktadır. (Denklem 2.1) ve (Denklem 2.3) kullanılarak X özniteliğinin Y özniteliği hakkındaki bilgi kazancı

değeri hesaplanabilmektedir. (Denklem 2.4)'te bilgi kazancı matematiksel hesabı gösterilmiştir.

$$\text{IG} = E(Y) - E(Y\backslash X) \tag{2.4}$$

2.3.1.2. Sarmal (Wrapper) yöntemler

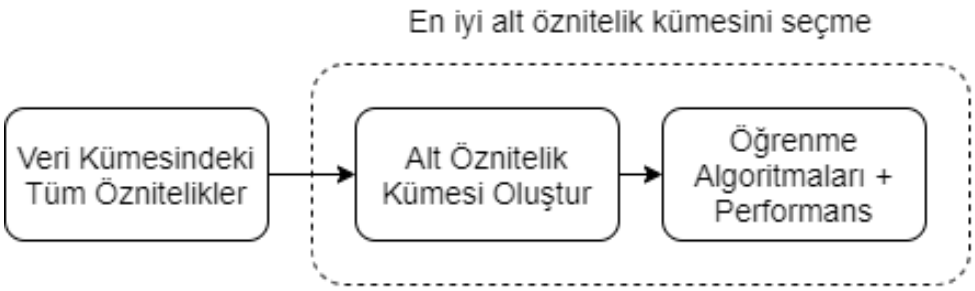
Sarmal yöntemler seçilen öznitelik alt kümesinin değerlendirilmesinde öğrenme prosedürü olarak kullanılmaktadır. Sarmal yöntemler filtreleme yöntemlerinden farklı olarak üç birleşene dayalı öznitelik seçimidir: bir arama stratejisi, bir tahminci ve bir değerlendirme işlevidir. Arama stratejisi veri kümesindeki değerlendirilecek özniteliklerin alt kümesini belirlemektedir. Tahminci herhangi bir sınıflandırma yöntemi olmakla birlikte performans ve en iyi doğruluğu veren optimum alt kümeyi bulabilmek için arama stratejisi tarafından belirlenen öznitelik alt kümesi üzerinde değerlendirme işlevini kullanmaktadır. Değerlendirme işlevi doğruluk oranı vb. amaca yönelik kriterler olabilmektedir. Başka bir ifadeyle sarmal yöntemler en iyi sınıflandırma doğruluğunu sağlayan ve en az hata oranını üreten öznitelik alt kümesinin seçilmesi amaçlanmaktadır. Sarmal yöntemler veri kümesinde bulunan özniteliklerden oluşan çok sayıda alt küme ile eğitildiğinden çok fazla zaman almaktadır. Bununla birlikte sarmal yöntemler filtreleme yöntemlerinden daha iyi performans göstermektedir (Haq, Onik ve Shah, 2015; Khammassi ve Krichen, 2017; Alamiedy ve ark., 2019). Şekil 2.12.'de sarmal yöntemin süreci gösterilmektedir.



Şekil 2.12. Sarmal (Wrapper) yöntem süreci (Alamiedy ve ark., 2019)

2.3.1.3. Gömülü (Embedded) yöntemler

Gömülü yöntemler, sarmal yöntemlerden farklı olarak model oluşturma süreci içerisinde öznitelik seçimi gerçekleştirmektedir. Sarmal yöntemlerde olduğu gibi her alt öznitelik kümesinde tekrar tekrar tahmin modelini eğitmekle uğraşmadığından öznitelik seçimi sarmal yöntemlerden daha hızlı olarak seçilmektedir. Seçilen öznitelikler belirli bir sınıflandırma modeline özgüdür (Haq, Onik ve Shah, 2015). Şekil 2.13.'de gömülü yöntemin süreci gösterilmektedir.



Şekil 2.13. Gömülü (Embedded) yöntem süreci (Alamiedy ve ark., 2019)

2.3.2. Öznitelik seçiminin avantajları ve dezavantajları

Öznitelik seçiminin avantajları (Haq, Onik ve Shah, 2015; Siddique ve ark., 2017; Ahmad ve Aziz, 2019):

- Boyutsal bela (ing. curse of dimensionality) etkilerini azaltmaktadır.
- Oluşturulacak modelin hesaplama süresinin azaltılmasına katkı sağlamaktadır.
- Oluşturulacak modelin sınıflandırma performansının iyileştirilmesine katkı sağlamaktadır.
- Veri kümesi içerisindeki gereksiz ve ilgisiz verilerin kaldırılmasını sağlamaktadır.
- Veri kümesinin hesaplama karmaşıklığını azaltmaktadır.
- Aşırı uyum gösterme (ing. overfitting) probleminin engellenmesine yardımcı olmaktadır.

Öznitelik seçiminin dezavantajları (Bilgin, 2018):

- Veri kümesi içerisindeki en iyi alt öznitelik kümesinin seçilmesinin zaman alıcı olmasıdır.
- Ek işlem maliyeti oluşturmasıdır.

BÖLÜM 3. KULLANILAN VERİ KÜMESİ VE DENEYSEL ÇALIŞMA

Bu bölümde, çalışmada kullanılan veri kümesi tanıtılmıştır. Ardından önerilen modelden ve önerilen modelin uygulanma aşamalarından bahsedilmiştir. Bununla birlikte önerilen model üzerinde algoritmaların performanslarının ölçümünden, uygulama ortamından ve araştırmanın bulgularından bahsedilmektedir.

3.1. Veri Kümesi

NSL-KDD veri kümesi, orijinal KDD Cup 99 (University of California, 1999) veri kümesinin geliştirilmiş bir versiyonudur. Aynı zamanda KDD Cup veri kümesinin devamı olarak da adlandırılır. NSL-KDD veri kümesinin literatürde bulunan saldırı tespit sistemi çalışmalarının birçoğunda kıyaslama veri kümesi olarak kullanılması, bu çalışmada da tercih edilme sebeplerinden biri olmuştur. Bu KDD Cup 99 veri kümesi MIT Lincoln Laboratuvarları tarafından tipik bir ABD Hava Kuvvetleri yerel ağı simüle edilip 9 hafta boyunca ham TCP verileri kaydedilerek oluşturulmuştur. İlk 7 hafta eğitim kümesini, son 2 hafta ise test kümesini oluşturmak için kullanılmıştır. NSL-KDD (Canadian Institute for Cybersecurity, 2009) veri kümesi, eğitim ve test kümesindeki çok sayıda fazlalık ve tekrarlanan kayıtların varlığı gibi KDD Cup 99 veri kümesindeki bazı doğal sorunları çözmektedir. NSL-KDD veri kümesi 41 öznitelik ve 1 sınıf etiketi içermektedir. Bu 41 öznitelikten “protocol-type”, “service” ve “flag” öznitelikleri kategorik, 6 öznitelik ikili ve diğer öznitelikler ise süreklidir. NSL-KDD veri kümesi “KDDTrain+” ve “KDDTest+” olarak belirtilen sırasıyla 125,973 ve 22,544 satır içeren eğitim ve test veri kümesine sahiptir. Ek olarak NSL-KDD veri kümesi sınıf etiketi beş farklı sınıf içermektedir. Bunların dördü saldırı sınıfı: Probe, DoS, R2L ve U2R diğeri de normal kullanım sınıfıdır. Tablo 3.1.’de beş sınıfın “KDDTrain+” ve “KDDTest+” veri kümelerindeki dağılımları gösterilmiştir.

Bu normal sınıf ve dört saldırı sınıfı aşağıda açıklanmıştır (Haq, Onik ve Shah, 2015; Gadal ve Mokhtar, 2017; Rai, 2020; Tang, Luktarhan ve Zhao, 2020).

- Bilgi tarama saldırısı (Probe attack): Hedef sistemin güvenlik kontrollerini atlatmak için bilgisayar ağı hakkında bilgi toplama girişimidir (Travallaee ve ark., 2009).
- Hizmet engelleme saldırısı (Denial of service attack - DoS): Sunucu sistemin üzerinde bulunan kaynakların (Ağ, CPU veya Bellek) işleyebileceğinden fazla ağ trafiği göndererek makinenin yavaşlamasına veya kapanmasına neden olan saldırıları içermektedir. Meşru kullanıcıların ağ trafiği ve sunulan hizmetlere erişimi DoS saldırılarından etkilenmektedir (Travallaee ve ark., 2009; Khafajeh, 2020; Tang, Luktarhan ve Zhao, 2020).
- Yönetici hesabı ile yerel oturum açma saldırısı (Remote to local attack - R2L): Bir ağ üzerinden hedef sisteme paket gönderebilen ancak hedef sistemde hesabı olmayan bir saldırganın, hedef sistemde erişim kazanmak için bir güvenlik açığından yararlandığında meydana gelmektedir (Travallaee ve ark., 2009).
- Kullanıcı hesabının yönetici hesabına yükseltilmesi saldırısı (User to root attack - U2R): Saldırganın sistemdeki normal bir kullanıcı hesabına erişimi üzerinden sistemdeki bazı güvenlik açıklarını kullanarak yetkili erişim elde ettiği saldırı çeşididir. (Travallaee ve ark., 2009).
- Normal: Sistem içerisindeki kullanıcılarının herhangi bir saldırgan eylemde bulunmadığı ağ trafiği şeklinde tanımlanır.

Tablo 3.1. NSL-KDD veri kümesindeki sınıf bazında verilerin dağılımı

		KDDTrain+ (125,973)	KDDTest+ (22,544)
Normal	Normal	67,343	9,711
Saldırı	Probe	11,656	2,421
	DoS	45,927	7,460
	R2L	995	2,885
	U2R	52	67

Tablo 3.2.'de NSL-KDD veri kümesi içerisindeki 41 öznitelik ve tipleri gösterilmektedir. NSL-KDD veri kümesi Tablo 3.3.'de gösterildiği gibi, her biri dört sınıfa (Probe, DoS, R2L ve U2R) ayrılmış toplam 39 saldırı türü içermektedir. Bunlardan 22'si "KDDTrain+" veri setinde bulunurken "KDDTest+" veri setinde ise "KDDTrain+" veri setinde bulunanlar dâhil ek olarak 17 farklı saldırı türü içermektedir. Ayrıca Tablo 3.3.'de sadece "KDDTest+" veri setinde bulunan saldırı türleri koyu olarak gösterilmektedir.

Tablo 3.2. NSL-KDD veri kümesi içerisindeki öznitelikler ve tipleri

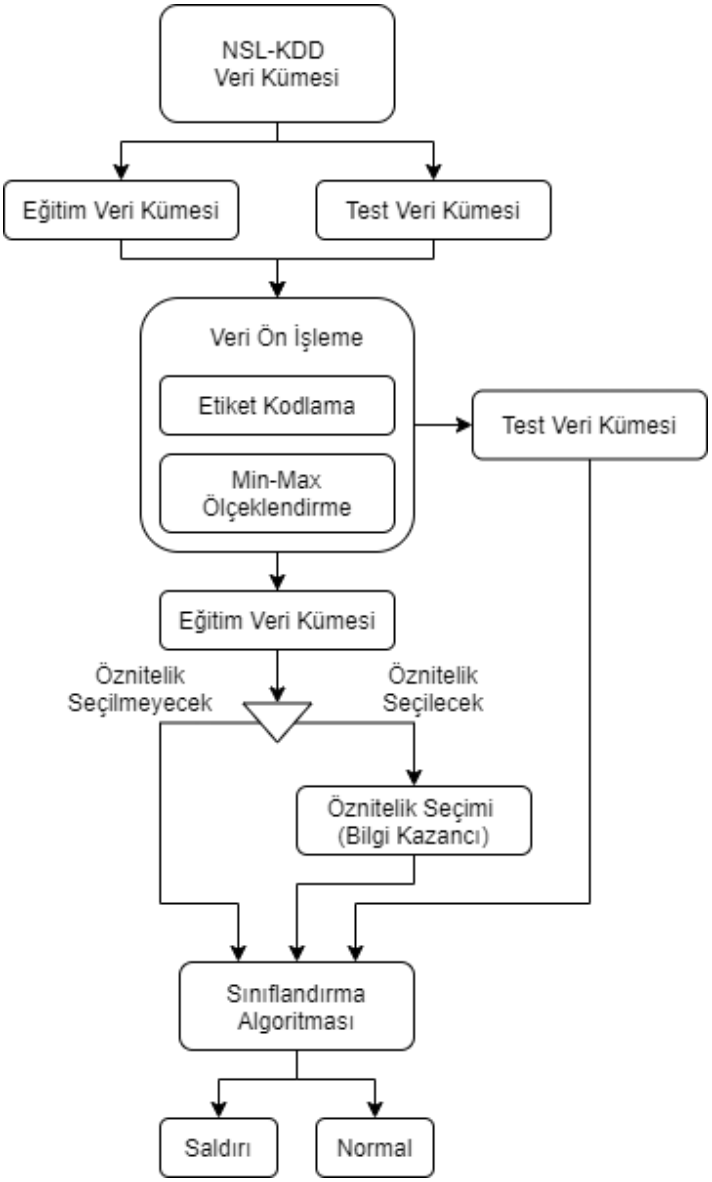
No.	Öznitelik İsmi	Tipi	No.	Öznitelik İsmi	Tipi
1	Duration	Sürekli	22	Is_guest_login	İkili
2	Protocol_type	Kategorik	23	Count	Sürekli
3	Service	Kategorik	24	Srv_count	Sürekli
4	Flag	Kategorik	25	Serror_rate	Sürekli
5	Src_bytes	Sürekli	26	Srv_serror_rate	Sürekli
6	Dst_bytes	Sürekli	27	Rerror_rate	Sürekli
7	Land	İkili	28	Srv_rerror_rate	Sürekli
8	Wrong_fragment	Sürekli	29	Same_srv_rate	Sürekli
9	Urgent	Sürekli	30	Diff_srv_rate	Sürekli
10	Hot	Sürekli	31	Srv_diff_host_rate	Sürekli
11	Num_failed_logins	Sürekli	32	Dst_host_count	Sürekli
12	Logged_in	İkili	33	Dst_host_srv_count	Sürekli
13	Num_compromised	Sürekli	34	Dst_host_same_srv_rate	Sürekli
14	Root_shell	İkili	35	Dst_host_diff_srv_rate	Sürekli
15	Su_attempted	İkili	36	Dst_host_same_src_port_rate	Sürekli
16	Num_root	Sürekli	37	Dst_host_srv_diff_host_rate	Sürekli
17	Num_file_creations	Sürekli	38	Dst_host_serror_rate	Sürekli
18	Num_shells	Sürekli	39	Dst_host_srv_serror_rate	Sürekli
19	Num_access_files	Sürekli	40	Dst_host_rerror_rate	Sürekli
20	Num_outbound_cmds	Sürekli	41	Dst_host_srv_rerror_rate	Sürekli
21	Is_host_login	İkili			

Tablo 3.3. NSL-KDD Veri kümesindeki saldırı türleri ve ilgili sınıfları

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

3.2. Önerilen Model

Bu başlıkta Şekil 3.1.'de gösterilen önerilen modelin mimarisi tanıtılmaktadır. NSL-KDD veri kümesi kullanılmadan önce ön işlemden geçirilir. Burada kategorik veriler etiket kodlama yöntemiyle sayısal verilere dönüştürülür. Daha sonra veriler min-max ölçeklendirme yöntemi kullanılarak [0, 1] aralığına dönüştürülür. Daha sonra bilgi kazancı algoritması kullanılarak eğitim verileri üzerinde öznitelik seçimi gerçekleştirilir. Son olarak sınıflandırma algoritmaları kullanılarak önerilen model test verilerinin ait olduğu sınıfları tahmin etmek için çalıştırılır ve elde edilen bulgular performans metrikleri aracılığıyla analiz edilir.



Şekil 3.1. Önerilen saldırı tespit sistemi modeli

3.3. Veri Ön İşleme

Bu adım önerilen modelin ilk aşamasıdır. Veri ön işleme aşaması 3 alt başlık halinde incelenmektedir. Bu kısımda veri temizleme, veri dönüşümü ve veri ölçeklendirme yapılmaktadır.

3.3.1. Veri temizleme

Bu adımda NSL-KDD veri kümesi içerisinde bulunan “num_outbound_cmds” özniteliğinin sadece 0 değeri barındırdığı görülmüştür. Bu değer önerilen model üzerinde kullanılan algoritmalara katkı sağlamadığı için kaldırılmıştır. Bu öznitelik kaldırıldıktan sonra NSL-KDD veri kümesinde 40’ı öznitelik 1’i de sınıf etiketi kalmıştır.

3.3.2. Veri dönüşümü

Veri temizleme yapıldıktan sonra verilere veri dönüşümü adımı uygulanır. Bu adımda NSL-KDD veri kümesi Tablo 3.3.’de gösterildiği gibi 39 saldırı türü içermektedir. Bu saldırı türleri 4 sınıfa ayrılmıştır. NSL-KDD veri kümesi içerisinde bulunan 4 saldırı sınıfı tek bir “Saldırı” sınıfında birleştirilmiştir. Bunun sonucunda veri kümesi “Saldırı” ve “Normal” olmak üzere 2 sınıflı hale getirilmiştir. Bununla birlikte Random Forest ve AdaBoost algoritmaları, veri setinde bulunan kategorik öznitelikleri direkt olarak kullanamamaktadır. Önceki bölümlerde CatBoost algoritmasının kategorik öznitelikleri kullanabildiğinden bahsedilmişti. Fakat burada CatBoost algoritmasını Random Forest ve AdaBoost algoritmalarının performansıyla karşılaştırabilmek için NSL-KDD veri kümesinde bulunan “protocol_type”, “service” ve “flag” kategorik öznitelikleri, etiket kodlama yöntemi aracılığıyla sayısal hale dönüştürülmüştür.

Etiket kodlama yöntemi kategorik değerler içeren özniteliklerdeki ifadelere sırasıyla {0, 1, 2, …, N-1} şeklinde sayısal değerler atayarak kategorik öznitelikler üzerinde matematiksel işlem yapılabilecek özniteliklere, yani sayısal özniteliklere dönüştürmek için kullanılan bir yöntemdir. Tablo 3.4., Tablo 3.5. ve Tablo 3.6.’da etiket kodlama yöntemi kullanıldıktan sonra kategorik özniteliklerin aldığı değerler verilmektedir.

Tablo 3.4. NSL-KDD veri kümesindeki protocol_type özniteliğinin sayısal değerleri

Değerler	Protocol_type Adı
0	icmp
1	tcp
2	udp

Tablo 3.5. NSL-KDD veri kümesindeki flag özniteliğinin sayısal değerleri

Değerler	Flag Adı	Değerler	Flag Adı
0	OTH	6	S1
1	REJ	7	S2
2	RSTO	8	S3
3	RSTOS0	9	SF
4	RSTR	10	SH
5	S0		

Tablo 3.6. NSL-KDD veri kümesindeki service özniteliğinin sayısal değerleri

Değerler	Service Adı	Değerler	Service Adı	Değerler	Service Adı
0	IRC	24	http	48	printer
1	X11	25	http_2784	49	private
2	Z39_50	26	http_443	50	red_i
3	aol	27	http_8001	51	remote_job
4	auth	28	imap4	52	rje
5	bgp	29	iso_tsap	53	shell
6	courier	30	klogin	54	smtp
7	csnet_ns	31	kshell	55	sql_net
8	ctf	32	ldap	56	ssh
9	daytime	33	link	57	sunrpc
10	discard	34	login	58	supdup
11	domain	35	mtp	59	systat
12	domain_u	36	name	60	telnet
13	echo	37	netbios_dgm	61	tftp_u
14	eco_i	38	netbios_ns	62	tim_i
15	ecr_i	39	netbios_ssn	63	time
16	efs	40	netstat	64	urh_i
17	exec	41	nnsp	65	urp_i
18	finger	42	nntp	66	uucp
19	ftp	43	ntp_u	67	uucp_path
20	ftp_data	44	other	68	vmnet
21	gopher	45	pm_dump	69	whois
22	harvest	46	pop_2		
23	hostnames	47	pop_3		

3.3.3. Veri ölçeklendirme

Veri dönüşümü yapılmış olan verilere veri ölçeklendirme adımı uygulanmaktadır. Veri ölçeklendirme özniteliklerin farklı ölçeklerde değerler barındırdığı veri kümelerinde uygulanmasının yararlı olduğu belirtilmektedir. Bu bağlamda, min-max ölçeklendirme tekniği veriler içerisindeki tüm ilişkileri tam olarak koruma avantajına sahip

olmaktadır (Devan ve Khare, 2020). Bu işlem veri kümesi içerisindeki öznitelikleri aynı ölçeğe getirdiği için eğitim süresinde bir iyileştirme sağlayabilmektedir (Ahmad ve Aziz, 2019). Ayrıca büyük sayısal aralık barındıran özniteliklerin daha küçük sayısal ağırlık barındıran özniteliklere üstünlük sağlamasının önüne geçilmektedir (El Boujnouni ve Jedra, 2018).

Min-max ölçeklendirme yönteminin yukarıdaki paragrafta bahsedilen avantajlarından yararlanmak amacıyla bu çalışmada tercih edilmiştir. Bu amaç doğrultusunda NSL-KDD veri kümesi içerisindeki öznitelikler min-max ölçeklendirme yöntemi kullanarak [0, 1] aralığına ölçeklendirilmiştir. Bunun için (Denklem 3.1) uygulanmaktadır.

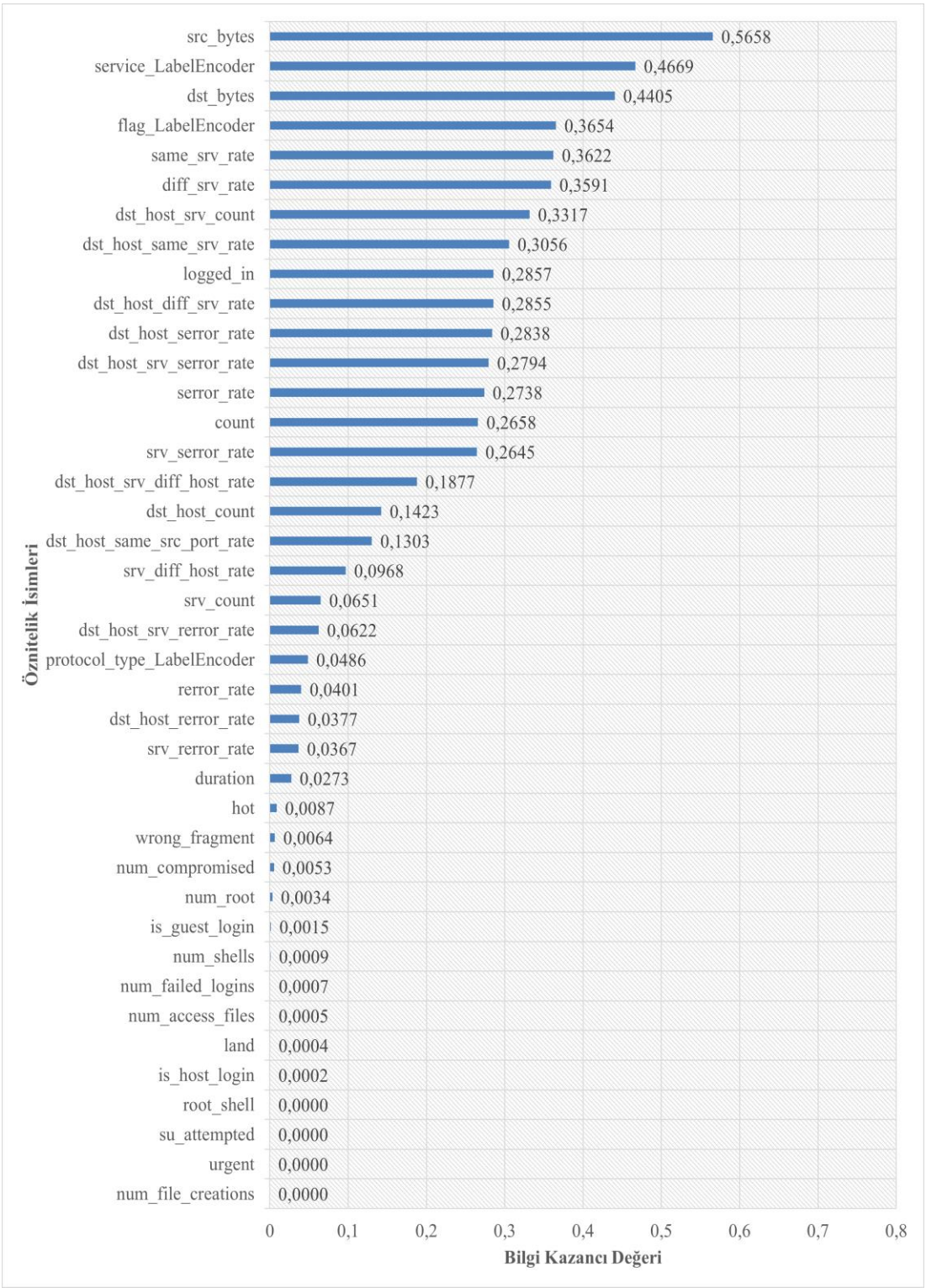
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

(Denklem 3.1)'de X orijinal değerdir. X_{min} ve X_{max} her özniteliğin minimum ve maksimum değerleridir. Bu işlemin sonucunda X orijinal değeri için X_{norm} değeri elde edilmektedir.

3.4. Öznitelik Seçimi

3.4.1. Bilgi kazancı (Information gain)

Veri ölçeklendirme yönteminden sonra verilere öznitelik seçimi adımını uygulanmaktadır. Öznitelik seçiminin getirdiği avantajlardan önceki bölümlerde bahsedilmiştir. Bu avantajlardan yararlanmak amacıyla öznitelik seçimi yöntemi olarak El Boujnouni ve Jedra (2018)'den esinlenilerek filtreleme yöntemlerinden bir tanesi olan bilgi kazancı yöntemi tercih edilmiştir. El Boujnouni ve Jedra (2018)'nın çalışması incelendiğinde bilgi kazancı yöntemi sonucu elde edilen değerlerden IG < 0,001 değerinden küçük olan öznitelikleri veri kümesinden kaldırmayı seçmişlerdir. Bunun sonucunda önerdikleri modellin doğruluğu üzerinde %5'lik bir artış sağlandığı görülmektedir (El Boujnouni ve Jedra, 2018). Bu nedenle, bu çalışmada da IG < 0,001 değerinden küçük olan özniteliklerin kaldırılması tercih edilmiştir. Şekil 3.2.'de NSL-KDD veri kümesi içerisindeki 40 özniteliğin bilgi kazancı değerleri gösterilmektedir.



Şekil 3.2. NSL-KDD veri kümesindeki 40 özniteliğin bilgi kazancı değeri.

Şekil 3.2.'ye bakıldığında bilgi kazancı değeri IG ≥ 0,001 değerinden büyük 31 öznitelik olduğu görülmektedir. Bu 31 öznitelik kullanılarak deneysel sonuçlar sonraki bölümlerde verilmektedir.

3.5. Deneysel Çalışma

Uygulamanın bu aşamasında deneysel çalışma olarak önerilen modelde kullanılacak sınıflandırma algoritması seçilecektir. Bu bağlamda, öznitelik seçimi yapıldıktan sonra veriler topluluk öğrenme yöntemlerinden sınıflandırma algoritmaları ile sınıflandırılmaktadır. Şekil 3.1.'deki önerilen modelde ikili sınıflandırma yöntemiyle algoritmaların başarıları incelenecektir. Bu doğrultuda CatBoost, Random Forest ve AdaBoost algoritmalarının saldırı tespit başarıları karşılaştırılacaktır.

3.5.1. Performans metrikleri

Makine öğrenimi ve veri madenciliği algoritmaları içerisindeki sınıflandırma modellerini değerlendirmek için birçok faklı performans ölçüsü kullanılmaktadır. Bu başlıkta deneysel çalışmada kullanılan sınıflandırıcı algoritmaların performanslarının ölçümünde kullanılan performans metriklerinden bahsedilecektir. Bu performans metriklerinin hesaplanması karışıklık matrisi (ing. confusion matrix) üzerinden açıklanmaktadır. Karışıklık matrisindeki her bir sütun gerçek bir sınıftaki örnekleri temsil ederken, her bir satır modelin tahmin ettiği sınıftaki örnekleri temsil etmektedir. Tablo 3.7.'de karışıklık matrisi gösterilmektedir.

Tablo 3.7. Karışıklık Matrisi

		Gerçek Değer	
		Pozitif	Negatif
Tahmin Edilen	Pozitif	Doğru Pozitif (DP)	Yanlış Pozitif (YP)
	Negatif	Yanlış Negatif (YN)	Doğru Negatif (DN)

- Doğru pozitif (DP): Gerçekte saldırı olarak etiketlenen örneklerin oluşturulan model tarafından da saldırı olarak sınıflandırılmasıdır.

- Doğru negatif (DN): Gerçekte saldırı değil şeklinde etiketlenen örneklerin oluşturulan model tarafından da saldırı değil şeklinde sınıflandırılmasıdır.
- Yanlış pozitif (YP): Gerçekte saldırı değil şeklinde etiketlenen örneklerin oluşturulan model tarafından saldırı olarak sınıflandırılmasıdır.
- Yanlış negatif (YN): Gerçekte saldırı olarak etiketlenen örneklerin oluşturulan model tarafından saldırı değil şeklinde sınıflandırılmasıdır.

Doğruluk (Accuracy) ölçütü: Sınıflandırma algoritmaları tarafından doğru tahmin edilen örnek sayısının veri kümesi içerisinde bulunan tüm örnek sayısına oranı şeklinde tanımlanmaktadır (Bilgin, 2018). (Denklem 3.2)'de matematiksel formülü gösterilmektedir.

$$\text{Doğruluk} = \frac{DP + DN}{DP + DN + YP + YN} \tag{3.2}$$

Kesinlik (Precision) ölçütü: Sınıflandırma algoritmaları tarafından doğru sınıflandırılan pozitif örnek sayısının yine sınıflandırma algoritmaları tarafından doğru ve yanlış sınıflandırılan pozitif örnek sayısına bölümüdür. Bir başka deyişle, modelin saldırı olarak sınıflandırdığı örneklerin kaçının gerçekte saldırı olduğunun oranını vermektedir (Bilgin, 2018; Arslan, 2019). (Denklem 3.3)'te matematiksel formülü gösterilmektedir.

$$\text{Kesinlik} = \frac{DP}{DP + YP} \tag{3.3}$$

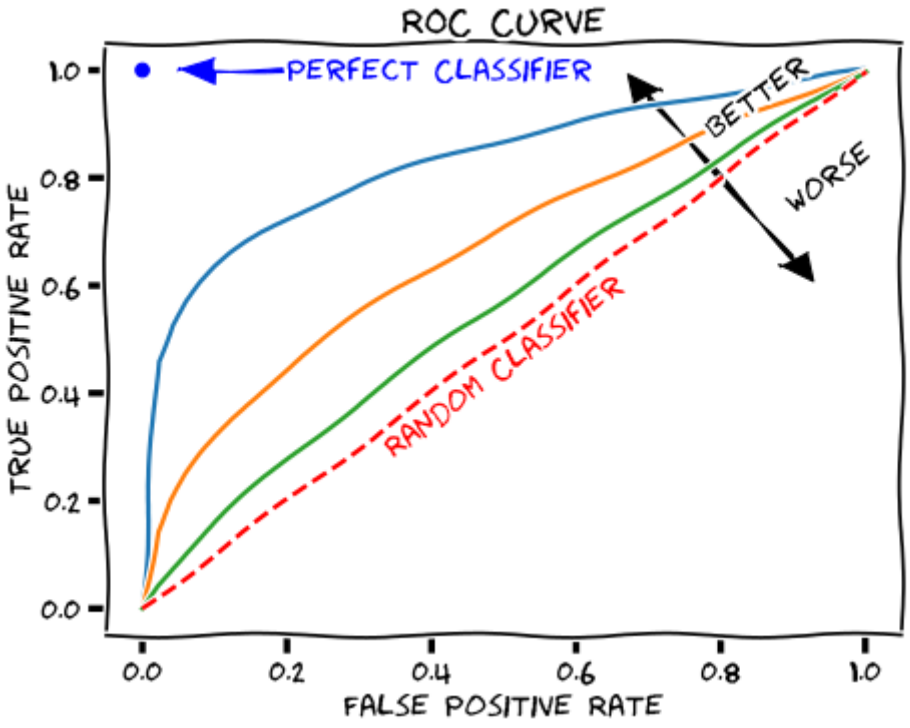
Duyarlılık (Recall) ölçütü: Sınıflandırma algoritmaları tarafından doğru sınıflandırılan pozitif örnek sayısının, gerçekte saldırı olan, model tarafından saldırı olarak sınıflandırdığı pozitif örneklerin ve gerçekte saldırı olan ama modelin saldırı olarak tanımlamadığı negatif örneklerin toplam sayısına bölümüdür. Bir başka deyişle, gerçekte saldırı olan örneklerin ne kadarının doğru sınıflandırıldığının oranı verilmektedir (Bilgin, 2018; Arslan, 2019). (Denklem 3.4)'te matematiksel formülü gösterilmektedir.

$$\text{Duyarlılık} = \frac{DP}{DP + YN} \tag{3.4}$$

F-Ölçütü (F-Measure): F-Ölçütü, duyarlılık ve kesinlik değerlerinin harmonik ortalamasıdır. Her iki ölçütün birlikte değerlendirilip daha doğru sonuç alınmasına katkıda bulunmaktadır (Bilgin, 2018). (Denklem 3.5)'te matematiksel formülü gösterilmektedir.

$$\text{F} - \text{Ölçütü} = \frac{2 * (Kesinlik * Duyarlılık)}{(Kesinlik + Duyarlılık)} \tag{3.5}$$

İşlem karakteristik eğrisi (Receiver operating characteristic (ROC)): İlk olarak 2. Dünya Savaşında savaş alanındaki düşman nesneleri tespit etmek için geliştirilmiştir. Daha sonra makine öğrenmesi ve veri madenciliği modellerinin değerlendirilmesinde kullanılan bir parametre haline gelmiştir (Bilgin, 2018; Wikipedia, 2020).



Şekil 3.3. Örnek ROC eğrisi (Thoma, 2018)

ROC eğrisinde X-ekseni doğru pozitif oranını gösterirken Y-ekseni yanlış pozitif oranını göstermektedir. ROC eğrisi sınıflandırıcının doğru pozitif için hesapladığı değer ile yanlış pozitif için hesapladığı değeri gösteren bir grafiktir. ROC eğrisi Şekil 3.3.'de görüldüğü üzere (0,0) ve (1,1) noktaları arasında çizilen köşegen şeklindedir (Bilgin, 2018).

ROC eğrisi kullanılarak hesaplanabilecek başka bir performans göstergesi de eğri altındaki alanın toplam alana oranını veren AUC (ing. Area Under Curve) değeridir. Bu değerin sonucu 1'e ne kadar yakınsa modelin sınıflandırma başarısı o kadar yüksektir. İdeal bir modelin ROC eğrisi (0,0) noktasından başlar (0,1) noktasına gider ve oradan (1,1) noktasına giderek bütün grafiği kaplar ve eğrinin altındaki alanın dikdörtgenin alanına oranı yani AUC değeri 1 olmaktadır. ROC eğrisinin köşegen üzerinde olması eğitilen modelin aslında rastgele seçim yaptığını yani eğitilemediğini göstermektedir (Arslan, 2019).

3.5.2. Uygulama

Önerilen modelin hem eğitilmesi hem de test edilmesi için kullanılan NSL-KDD veri kümesinin büyüklüğü ve bu veri kümesi üzerinde gerçekleştirilen veri dönüşümü, veri ölçeklendirme ve öznitelik adımları büyük miktarda RAM tüketmektedir. Bu nedenle deneyler Google tarafından sunulan CPU ve GPU tabanlı bir çerçeve barındıran Google Colab üzerinde gerçekleştirilmiştir. Google Colab üzerinde barındırdığı ek hesaplama gücü ve veri işleme kapasitesi sayesinde verilerin analiz edilmesinde kullanılmıştır. Deneysel ortam Google Colab ortamında geliştirilmiştir. Önerilen model Python programlama dili (versiyon 3.6.9), Scikit-learn (version 0.22.2) ve CatBoost (versiyon 0.24.3) kütüphaneleri kullanılarak ortaya çıkartılmıştır.

Önerilen model üzerindeki sınıflandırma algoritmalarının birbirleriyle kıyaslanabilmeleri amacıyla algoritmalar üzerinde kullanılan parametreler;

- CatBoost algoritmasının varsayılan hiper parametrelerinde ağaç sayısı 1000 olarak gelmektedir. Bu çalışmada algoritmanın diğer algoritmalarla

performansını kıyaslayabilmek amacıyla ağaç sayısı parametresi 100 olarak değiştirilmektedir. CatBoost algoritmasının NSL-KDD veri kümesi üzerindeki performans sonuçları ağaç sayısı dışındaki CatBoost kütüphanesindeki varsayılan (loss_function = "Logloss", Ağaç sayısı= 100, Öğrenme oranı= 0,03, L2 düzenlileştirme= 3, random_state= 0) parametreleri kullanılarak elde edilmektedir.

- Random Forest algoritmasının NSL-KDD veri kümesi üzerindeki performans sonuçları Scikit-learn kütüphanesindeki varsayılan (Ağaç sayısı= 100, random_state= 0) parametreleri kullanılarak elde edilmektedir.
- AdaBoost algoritmasının NSL-KDD veri kümesi üzerindeki performans sonuçları Scikit-learn kütüphanesindeki varsayılan (Ağaç sayısı= 100, Öğrenme oranı= 1, random_state= 0) parametreleri kullanılarak elde edilmektedir.

3.6. Araştırmanın Bulguları

3.6.1. CatBoost, AdaBoost ve Random Forest algoritmalarının bulguları

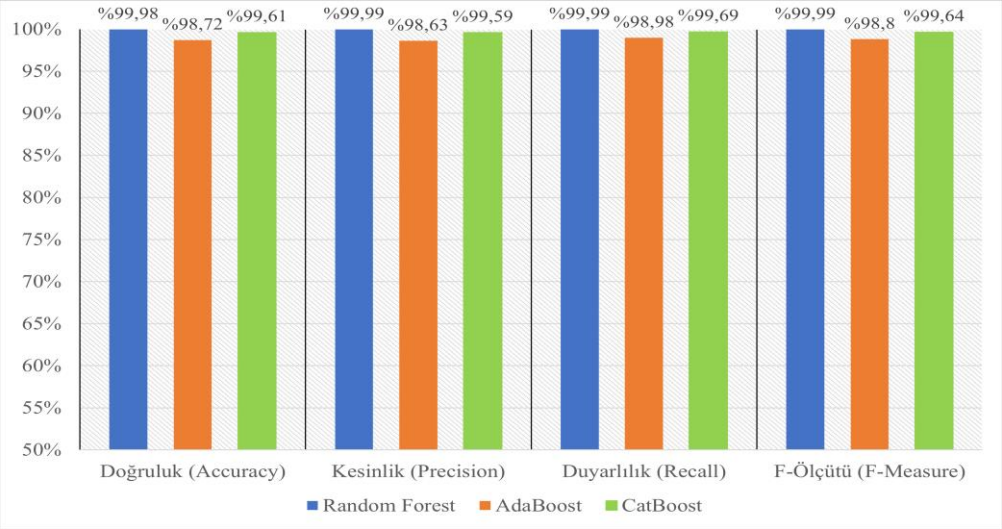
3.6.1.1. Öznitelik seçimi yapılmamış sınıflandırma algoritmalarının sonuçlarının karşılaştırılması

Bu bölümde sınıflandırma algoritmalarının NSL-KDD veri kümesi içerisindeki öznitelik seçimi yapılmamış 40 öznitelik içeren KDDTrain+ ve KDDTest+ veri kümeleri üzerindeki saldırı tespit performansları, performans metrikleri kullanılarak Tablo 3.8.'de gösterilmektedir.

Tablo 3.8. Sınıflandırma algoritmalarının 40 öznitelik üzerindeki sınıflandırma karşılaştırması

		Random Forest	AdaBoost	CatBoost
KDDTrain+	Doğruluk (Accuracy)	**% 99,98**	% 98,72	% 99,61
	Kesinlik (Precision)	**% 99,99**	% 98,63	% 99,59
	Duyarlılık (Recall)	**% 99,99**	% 98,98	% 99,69
	F-Ölçütü (F-Measure)	**% 99,99**	% 98,80	% 99,64
KDDTest+	Doğruluk (Accuracy)	% 76,96	% 74,77	**% 79,42**
	Kesinlik (Precision)	% 65,75	% 64,44	**% 68,40**
	Duyarlılık (Recall)	**% 97,08**	% 92,43	% 97,07
	F-Ölçütü (F-Measure)	% 78,40	% 75,94	**% 80,25**

Tablo 3.8.'deki KDDTrain+ veri kümesinin performans metrikleri değerlerinin görsel olarak incelenmesi Şekil 3.4.'te gösterilmektedir.



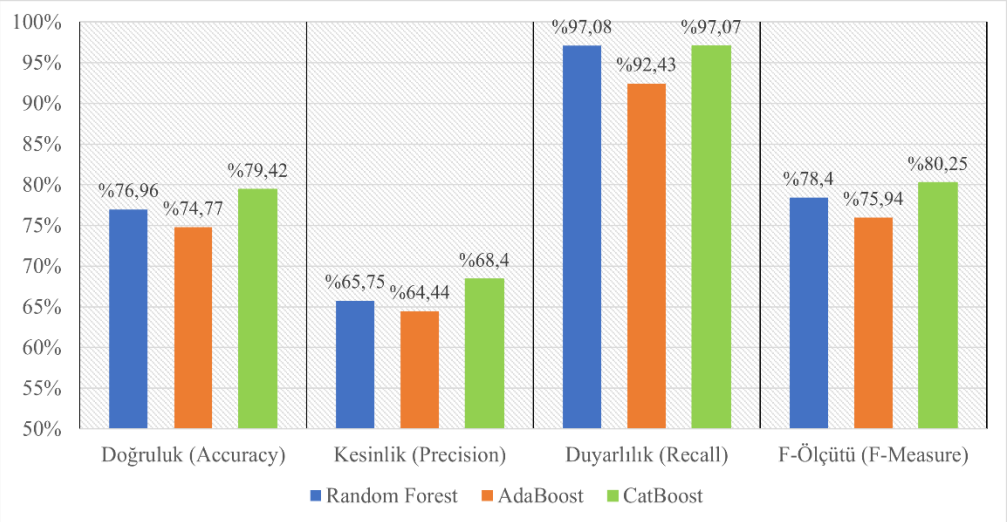
Şekil 3.4. Sınıflandırma algoritmalarının 40 öznitelik içeren KDDTrain+ veri kümesi üzerindeki performans karşılaştırmaları

Tablo 3.8. ve Şekil 3.4.'e bakıldığında kullanılan sınıflandırma algoritmalarının eğitilmesinde kullanılan 40 öznitelik içeren KDDTrain+ veri kümesindeki performans metrikleri değerleri gösterilmektedir. Performans metrikleri sonuçlarına bakıldığında

sınıflandırma algoritmalarının 40 öznitelik içeren KDDTrain+ veri kümesi üzerindeki saldırı tespitinde yaklaşık %99 oranında başarılı sonuçlar elde ettikleri görülmektedir.

Tüm kullanılan sınıflandırma algoritmaları arasındaki performans metriklerine bakıldığında büyük bir fark göze çarpmamaktadır. Bununla birlikte en yüksek performans metrik değerleri Random Forest algoritması tarafından elde edilirken, en düşük performans metrik değerlerinin ise AdaBoost algoritması tarafından elde edildiği görülmektedir. Doğruluk (accuracy) metriğine bakıldığında Random Forest algoritması %99,98, AdaBoost algoritması %98,97 ve CatBoost algoritması %99,61 değerlerini elde etmişlerdir. Kesinlik (precision) metriğine bakıldığında Random Forest algoritması %99,99, AdaBoost algoritması %98,63 ve CatBoost algoritması %99,59 değerlerini elde etmişlerdir. Duyarlılık (recall) metriğine bakıldığında Random Forest algoritması %99,99, AdaBoost algoritması %98,98 ve CatBoost algoritması %99,69 değerlerini elde etmişlerdir. F-ölçütü (f-measure) metriğine bakıldığında Random Forest algoritması %99,99, AdaBoost algoritması %98,8 ve CatBoost algoritması %99,64 değerlerini elde etmişlerdir.

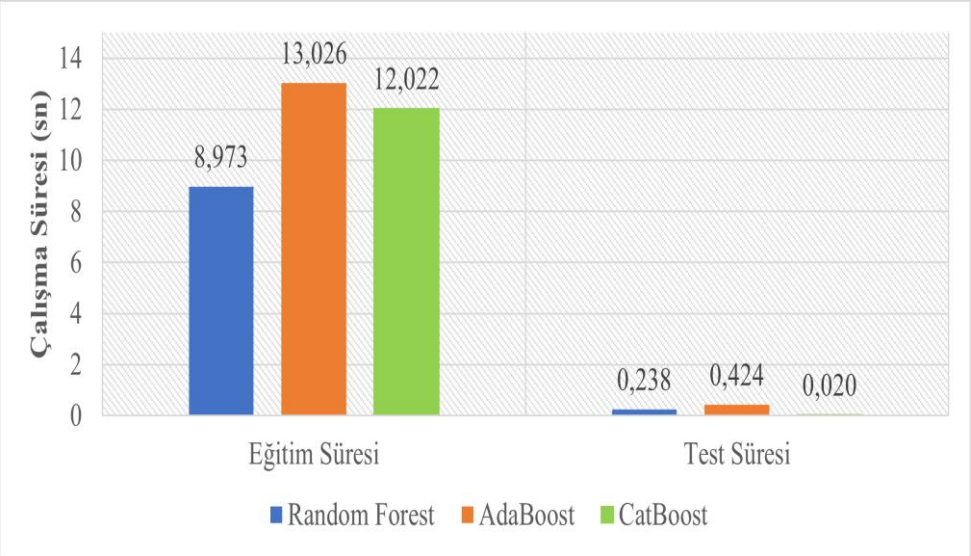
Bununla birlikte Tablo 3.8.'deki KDDTest+ veri kümesinin performans metrikleri değerlerinin görsel olarak incelenmesi Şekil 3.5.'te gösterilmektedir.



Şekil 3.5. Sınıflandırma algoritmalarının 40 öznitelik içeren KDDTest+ veri kümesi üzerindeki performans karşılaştırmaları

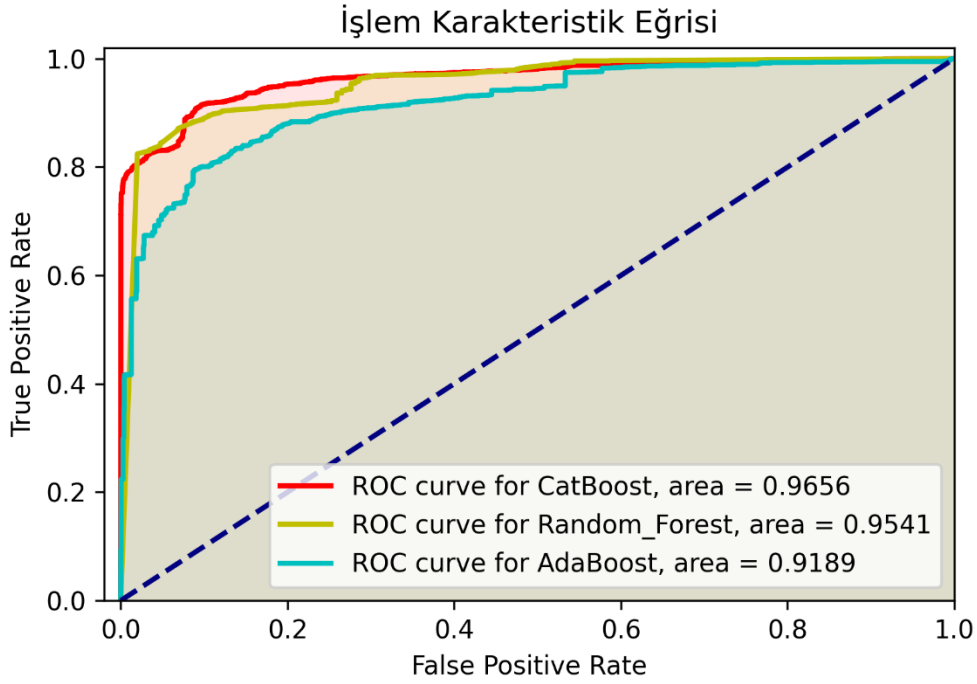
Şekil 3.5.'e bakıldığında kullanılan sınıflandırma algoritmalarının eğitim aşamasında görmedikleri saldırı türlerini barındıran 40 öznitelik içeren KDDTest+ veri kümesindeki performans metrik değerleri gösterilmektedir. Performans metrikleri sonuçlarına bakıldığında sınıflandırma algoritmalarının 40 öznitelik içeren KDDTest+ veri kümesi üzerindeki saldırı tespitinde %70-%80 aralığında başarılı sonuçlar elde ettikleri görülmektedir.

Tüm kullanılan sınıflandırma algoritmaların performans metrikleri bazında birbirleri arasında KDDTest+ veri kümesi üzerinde büyük bir fark göze çarpmamaktadır. Hemen hemen kullanılan bütün sınıflandırma algoritmaları aynı performans metriğinde %1-%5'lik farklarla benzer sonuçları elde etmişlerdir. Bununla birlikte duyarlılık (recall) metriği dışında en yüksek performans metriği değerleri CatBoost algoritması tarafından elde edildiği görülmektedir. En düşük performans metrikleri değerlerinin ise AdaBoost algoritması tarafından elde edildiği görülmektedir. Doğruluk (accuracy) metriğine bakıldığında Random Forest algoritması %76,96, AdaBoost algoritması %74,77 ve CatBoost algoritması %79,42 değerlerini elde etmişlerdir. Kesinlik (precision) metriğine bakıldığında Random Forest algoritması %65,75, AdaBoost algoritması %64,44 ve CatBoost algoritması %68,4 değerlerini elde etmişlerdir. Duyarlılık (recall) metriğine bakıldığında Random Forest algoritması %97,08, AdaBoost algoritması %92,43 ve CatBoost algoritması %97,07 değerlerini elde etmişlerdir. F-ölçütü (f-measure) metriğine bakıldığında Random Forest algoritması %78,4, AdaBoost algoritması %75,94 ve CatBoost algoritması %80,25 değerlerini elde etmişlerdir.



Şekil 3.6. Sınıflandırma algoritmalarının 40 öznitelik içeren KDDTrain+ ve KDDTest+ veri kümeleri üzerindeki çalışma süreleri

Şekil 3.6.'da sınıflandırma algoritmalarının 40 öznitelik içeren KDDTrain+ ve KDDTest+ veri kümeleri üzerindeki çalışma süreleri gösterilmektedir. Bu şekle bakıldığında Random Forest algoritmasının KDDTrain+ veri kümesi üzerinde çok daha hızlı bir eğitim sürecine sahip olduğu görülmektedir. Bununla birlikte KDDTest+ veri kümesinin test edilmesinde CatBoost algoritmasının çok daha hızlı olduğu görülmektedir.



Şekil 3.7. Sınıflandırma algoritmalarının 40 öznitelik içeren KDDTest+ veri kümesi üzerindeki ROC eğrisi

Şekil 3.7.'deki işlem karakteristik eğrisine (ROC) bakıldığında CatBoost algoritmasının 40 öznitelik içeren KDDTest+ veri kümesi üzerinde diğer algoritmalardan saldırıları tespit etmede daha başarılı olduğu görülmektedir.

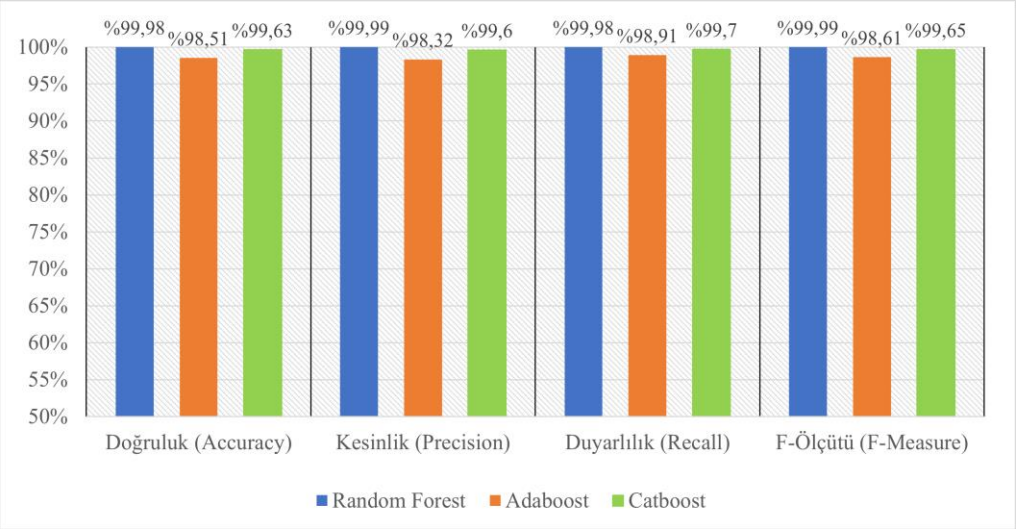
3.6.1.2. Bilgi kazancı yöntemi kullanılarak öznitelik seçimi yapılmış sınıflandırma algoritmalarının karşılaştırılması

Bu bölümde NSL-KDD veri kümesi üzerinde bilgi kazancı öznitelik seçimi yöntemi uygulanmasıyla birlikte 40 olan öznitelik sayısı, bilgi kazancı değeri 0,001'den büyük özniteliklerin seçilmesiyle 31 özniteliğe indirgenmektedir. Bu 31 özniteliği içeren KDDTrain+ ve KDDTest+ veri kümeleri kullanılarak sınıflandırma algoritmalarının saldırı tespit performansları, performans metrikleri kullanılarak Tablo 3.9.'da gösterilmektedir.

Tablo 3.9. Sınıflandırma algoritmalarının 31 öznitelik üzerindeki sınıflandırma karşılaştırması

		Random Forest	AdaBoost	CatBoost
KDDTrain+	Doğruluk (Accuracy)	**% 99,98**	% 98,51	% 99,63
	Kesinlik (Precision)	**% 99,99**	% 98,32	% 99,60
	Duyarlılık (Recall)	**% 99,98**	% 98,91	% 99,70
	F-Ölçütü (F-Measure)	**% 99,99**	% 98,61	% 99,65
KDDTest+	Doğruluk (Accuracy)	% 78,33	% 74,74	**% 79,43**
	Kesinlik (Precision)	% 67,20	% 64,40	**% 68,44**
	Duyarlılık (Recall)	**% 97,07**	% 92,48	% 96,95
	F-Ölçütü (F-Measure)	% 79,42	% 75,93	**% 80,24**

Tablo 3.9.'daki 31 öznitelik içeren KDDTrain+ veri kümesinin performans metrikleri değerlerinin görsel olarak incelenmesi Şekil 3.8.'de gösterilmektedir.



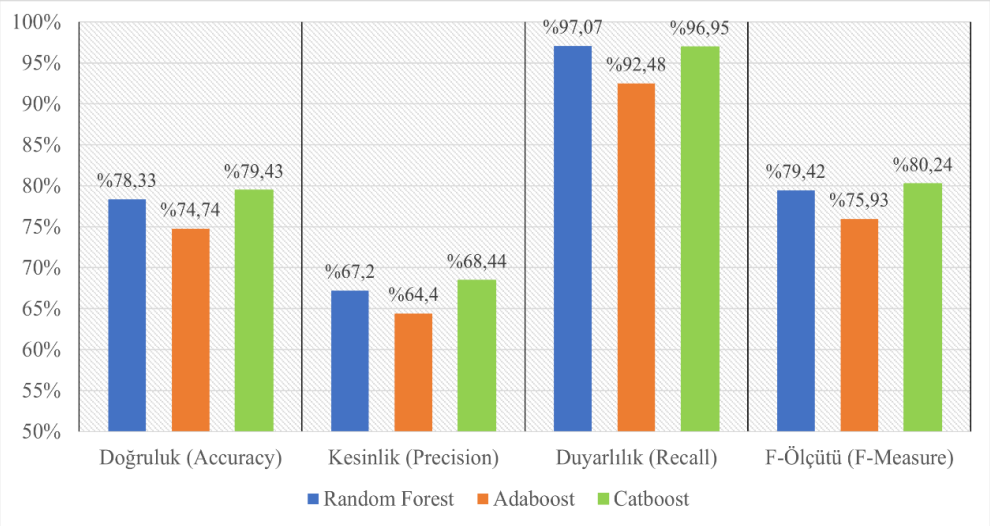
Şekil 3.8. Sınıflandırma algoritmalarının 31 öznitelik içeren KDDTrain+ veri kümesi üzerindeki performans karşılaştırmaları

Tablo 3.9. ve Şekil 3.8.'e bakıldığında kullanılan sınıflandırma algoritmalarının eğitilmesinde kullanılan 31 öznitelik içeren KDDTrain+ veri kümesindeki performans metrikleri değerleri gösterilmektedir. Performans metrikleri sonuçlarına bakıldığında sınıflandırma algoritmalarının 31 öznitelik içeren KDDTrain+ veri kümesi üzerindeki

saldırı tespitinde Tablo 3.8. ve Şekil 3.4.'tekilere benzer başarılı sonuçlar elde ettikleri görülmektedir.

Tüm kullanılan sınıflandırma algoritmaları arasındaki performans metriklerine bakıldığında büyük bir fark göze çarpmamaktadır. Sonuçlar öznitelik seçimi yapılmamış olan Tablo 3.8. ve Şekil 3.4.'teki performans metriklerinin sonuçlarıyla karşılaştırıldığında en yüksek performans metrikleri değerleri yine Random Forest algoritması tarafından elde edilirken, en düşük performans metrikleri değerlerinin ise yine AdaBoost algoritması tarafından elde edildiği görülmektedir. Doğruluk (accuracy) metriğine bakıldığında Random Forest algoritması %99,98, AdaBoost algoritması %98,51 ve CatBoost algoritması %99,63 değerlerini elde etmişlerdir. Kesinlik (precision) metriğine bakıldığında Random Forest algoritması %99,99, AdaBoost algoritması %98,32 ve CatBoost algoritması %99,6 değerlerini elde etmişlerdir. Duyarlılık (recall) metriğine bakıldığında Random Forest algoritması %99,98, AdaBoost algoritması %98,91 ve CatBoost algoritması %99,7 değerlerini elde etmişlerdir. F-ölçütü (f-measure) metriğine bakıldığında Random Forest algoritması %99,99, AdaBoost algoritması %98,61 ve CatBoost algoritması %99,65 değerlerini elde etmişlerdir.

Bununla birlikte Tablo 3.9.'daki 31 öznitelik içeren KDDTest+ veri kümesinin performans metrikleri değerlerinin görsel olarak incelenmesi Şekil 3.9.'da gösterilmektedir.

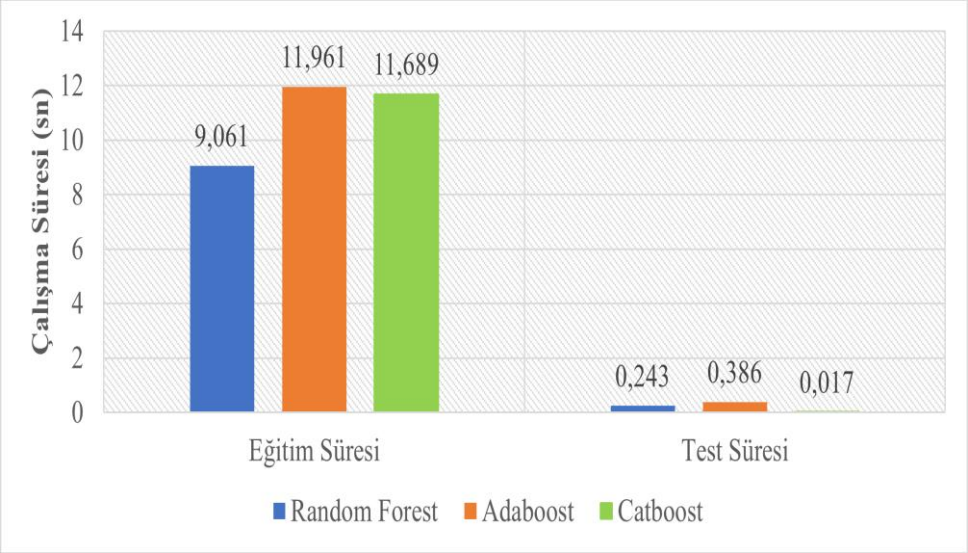


Şekil 3.9. Sınıflandırma algoritmalarının 31 öznitelik içeren KDDTest+ veri kümesi üzerindeki performans karşılaştırmaları

Şekil 3.9.'a bakıldığında kullanılan sınıflandırma algoritmalarının eğitilme aşamasında görmedikleri saldırı türlerini barındıran 31 öznitelik içeren KDDTest+ veri kümesindeki performans metrikleri değerleri gösterilmektedir. Performans metrikleri sonuçlarına bakıldığında sınıflandırma algoritmalarının 31 öznitelik içeren KDDTest+ veri kümesi üzerindeki saldırı tespitinde Tablo 3.8. ve Şekil 3.5.'tekilere benzer %70-%80 aralığında başarılı sonuçlar elde ettikleri görülmektedir.

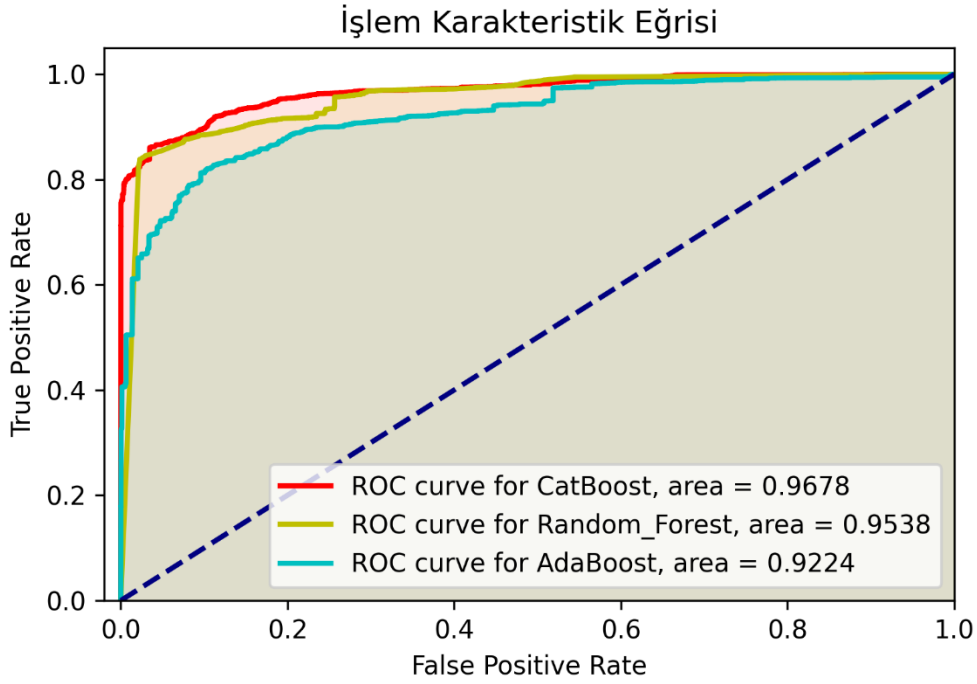
Tüm kullanılan sınıflandırma algoritmaları arasındaki performans metriklerine bakıldığında KDDTest+ veri kümesi üzerinde büyük bir fark göze çarpmamaktadır. Hemen hemen kullanılan bütün sınıflandırma algoritmaları aynı performans metriğinde %1-%5'lik farklarla benzer sonuçları elde etmişlerdir. Sonuçlar öznitelik seçimi yapılmamış olan Tablo 3.8. ve Şekil 3.5.'teki performans metriklerinin sonuçlarıyla karşılaştırıldığında en yüksek performans metrikleri değerleri yine CatBoost algoritması tarafından elde edilirken, en düşük performans metrikleri değerlerinin ise yine AdaBoost algoritması tarafından elde edildiği görülmektedir. Doğruluk (accuracy) metriğine bakıldığında Random Forest algoritması %78,33, AdaBoost algoritması %74,74 ve CatBoost algoritması %79,43 değerlerini elde etmişlerdir. Kesinlik (precision) metriğine bakıldığında Random Forest algoritması %67,2, AdaBoost algoritması %64,4 ve CatBoost algoritması %68,44 değerlerini elde etmişlerdir. Duyarlılık (recall) metriğine bakıldığında Random Forest algoritması

%97,07, AdaBoost algoritması %92,48 ve CatBoost algoritması %96,95 değerlerini elde etmişlerdir. F-ölçütü (f-measure) metriğine bakıldığında Random Forest algoritması %79,42, AdaBoost algoritması %75,93 ve CatBoost algoritması %80,24 değerlerini elde etmişlerdir.



Şekil 3.10. Sınıflandırma algoritmalarının 31 öznitelik içeren KDDTrain+ ve KDDTest+ veri kümeleri üzerindeki çalışma süreleri

Şekil 3.10.'da sınıflandırma algoritmalarının 31 öznitelik içeren KDDTrain+ ve KDDTest+ veri kümeleri üzerindeki çalışma süreleri gösterilmektedir. Bu şekle bakıldığında Random Forest algoritmasının KDDTrain+ veri kümesi üzerinde Şekil 3.6.'daki eğitim süresinden yaklaşık 1 saniye daha yüksek olmasına rağmen diğer algoritmalardan daha hızlı bir eğitim sürecine sahip olduğu görülmektedir. Bununla birlikte, CatBoost algoritmasının test süresi Şekil 3.6.'daki test süresine göre daha hızlı olduğu görülmektedir.



Şekil 3.11. Sınıflandırma algoritmalarının 31 öznitelik içeren KDDTest+ veri kümesi üzerindeki ROC eğrisi

Şekil 3.11.'deki işlem karakteristik eğrisine (ROC) bakıldığında CatBoost algoritmasının Şekil 3.7.'dekine benzer bir sonuçla 31 öznitelik içeren KDDTest+ veri kümesi üzerinde de diğer algoritmalardan saldırı tespitinde daha başarılı olduğu görülmektedir.

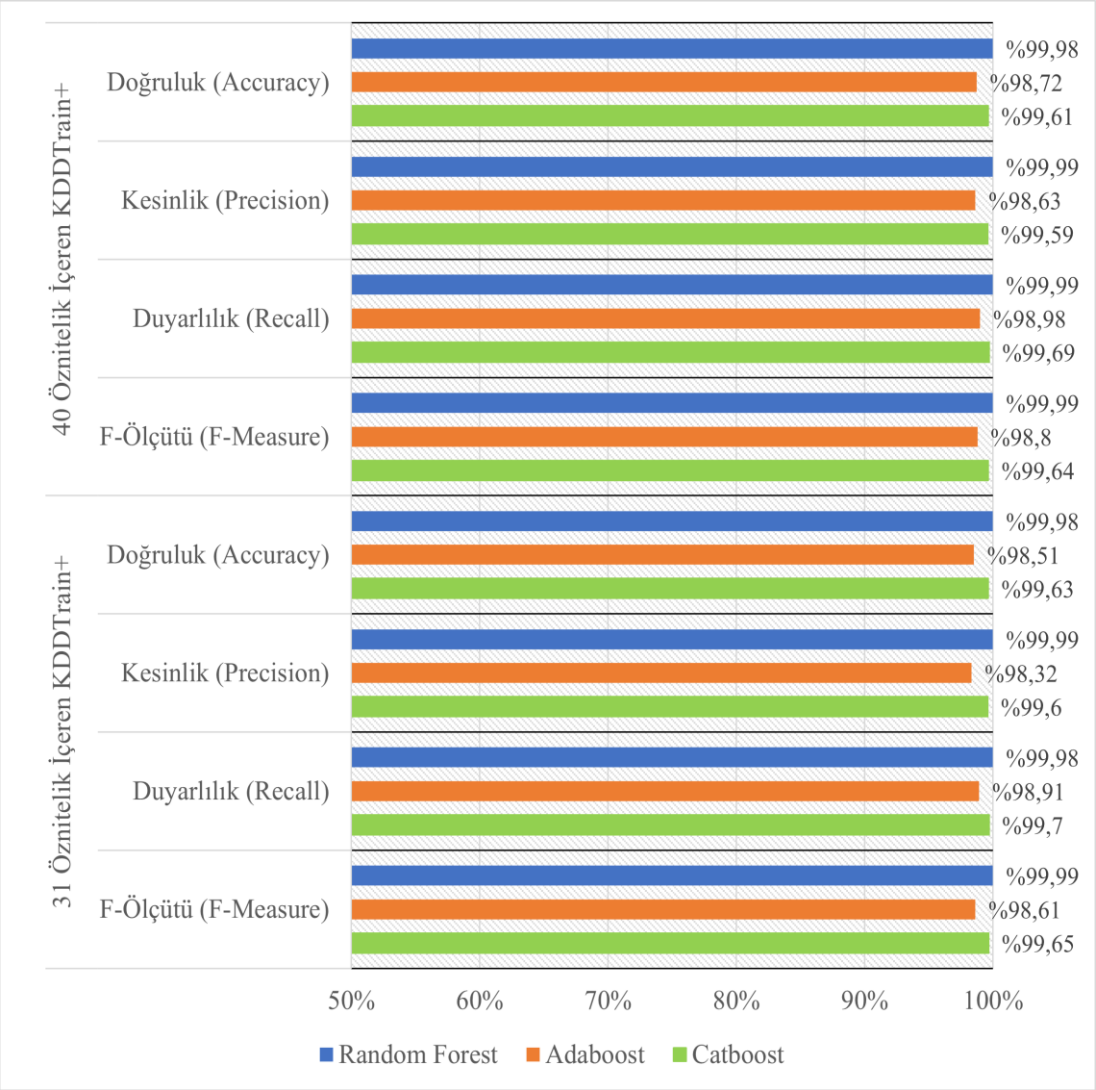
3.6.1.3. Öznitelik seçimi yapılmış ve yapılmamış NSL-KDD veri kümesindeki sınıflandırma algoritmalarının karşılaştırılması

Bu bölümde yukarıdaki bölümlerde ayrı ayrı anlatılan öznitelik seçimi yapılmamış olan NSL-KDD veri kümeleri üzerindeki sınıflandırma algoritmalarının performans metriklerinin analizi ile bilgi kazancı yöntemi kullanılarak öznitelik seçimi yapılan NSL-KDD veri kümeleri üzerindeki sınıflandırma algoritmalarının performans metriklerinin analizi birlikte değerlendirilmektedir. Tablo 3.10.'da öznitelik seçimi yapılmış ve yapılmamış NSL-KDD veri kümelerinden KDDTrain+ veri kümesi üzerindeki sınıflandırma algoritmalarının performans metrik sonuçları gösterilmektedir.

Tablo 3.10. Öznitelik seçimi yapılmış ve yapılmamış KDDTrain+ veri kümesi üzerindeki sınıflandırma algoritmalarının performans metrikleri sonuçları

		Random Forest	AdaBoost	CatBoost
40 Öznitelik İçeren KDDTrain+	Doğruluk (Accuracy)	**% 99,98**	% 98,72	% 99,61
	Kesinlik (Precision)	**% 99,99**	% 98,63	% 99,59
	Duyarlılık (Recall)	**% 99,99**	% 98,98	% 99,69
	F-Ölçütü (F-Measure)	**% 99,99**	% 98,80	% 99,64
31 Öznitelik İçeren KDDTrain+	Doğruluk (Accuracy)	**% 99,98**	% 98,51	% 99,63
	Kesinlik (Precision)	**% 99,99**	% 98,32	% 99,60
	Duyarlılık (Recall)	**% 99,98**	% 98,91	% 99,70
	F-Ölçütü (F-Measure)	**% 99,99**	% 98,61	% 99,65

Tablo 3.10.'daki öznitelik seçimi yapılmış ve yapılmamış NSL-KDD veri kümelerinden KDDTrain+ veri kümesi üzerindeki performans metrikleri değerlerinin görsel olarak incelenmesi Şekil 3.12.'de gösterilmektedir.



Şekil 3.12. Öznitelik seçimi yapılmış ve yapılmamış KDDTrain+ veri kümesi üzerinde sınıflandırma algoritmalarının performans karşılaştırmaları

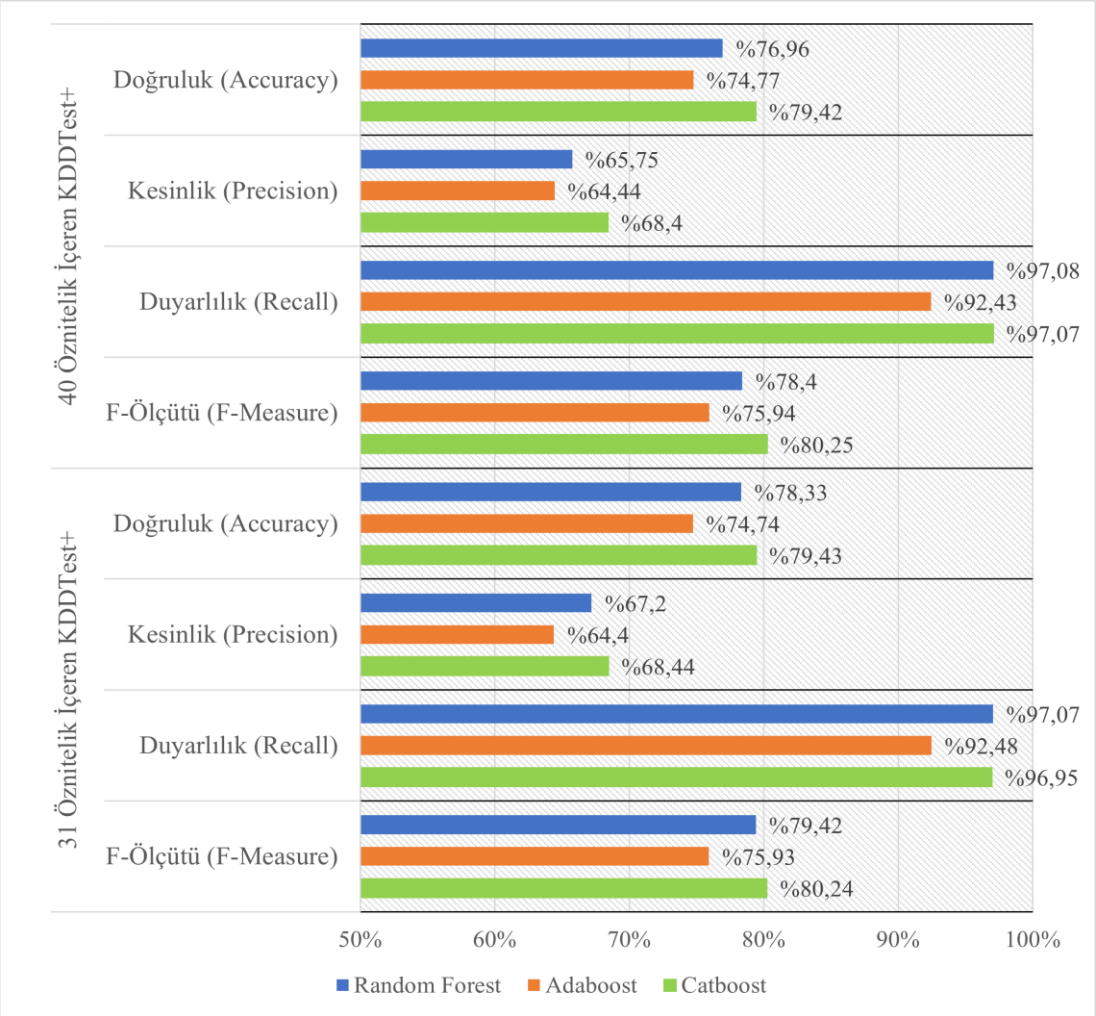
Tablo 3.10. ve Şekil 3.12.'de öznitelik seçimi yapılmış ve yapılmamış KDDTrain+ veri kümesi üzerindeki sınıflandırma algoritmalarının performans metrikleri incelendiğinde bilgi kazancı yöntemi ile seçilen 31 özniteliğin AdaBoost algoritması hariç sınıflandırma performansına olumlu katkı sağladığı görülmektedir. Öznitelik seçimi yapılmış ve yapılmamış veri kümelerinde de en başarılı algoritmanın Random Forest algoritması olduğu görülmektedir. Bununla birlikte önerilen model üzerinde uygulanan AdaBoost algoritmasının 2.3.2.'deki öznitelik seçiminin avantajları ve dezavantajları kısmında bahsedilen, öznitelik seçimi sınıflandırma performansının iyileştirilmesine katkı sağlar maddesini yerine getirmediği gözlemlenmiştir.

Bir diğer öznitelik seçimi yapılmış ve yapılmamış NLS-KDD veri kümelerinden olan KDDTest+ veri kümesi üzerindeki sınıflandırma algoritmalarının performans metrikleri sonuçları Tablo 3.11.'de gösterilmektedir.

Tablo 3.11. Öznitelik seçimi yapılmış ve yapılmamış KDDTest+ veri kümesi üzerindeki sınıflandırma algoritmalarının performans metrikleri sonuçları

		Random Forest	AdaBoost	CatBoost
40 Öznitelik İçeren KDDTest+	Doğruluk (Accuracy)	% 76,96	% 74,77	**% 79,42**
	Kesinlik (Precision)	% 65,75	% 64,44	**% 68,40**
	Duyarlılık (Recall)	**% 97,08**	% 92,43	% 97,07
	F-Ölçütü (F-Measure)	% 78,40	% 75,94	**% 80,25**
31 Öznitelik İçeren KDDTest+	Doğruluk (Accuracy)	% 78,33	% 74,74	**% 79,43**
	Kesinlik (Precision)	% 67,20	% 64,40	**% 68,44**
	Duyarlılık (Recall)	**% 97,07**	% 92,48	% 96,95
	F-Ölçütü (F-Measure)	% 79,42	% 75,93	**% 80,24**

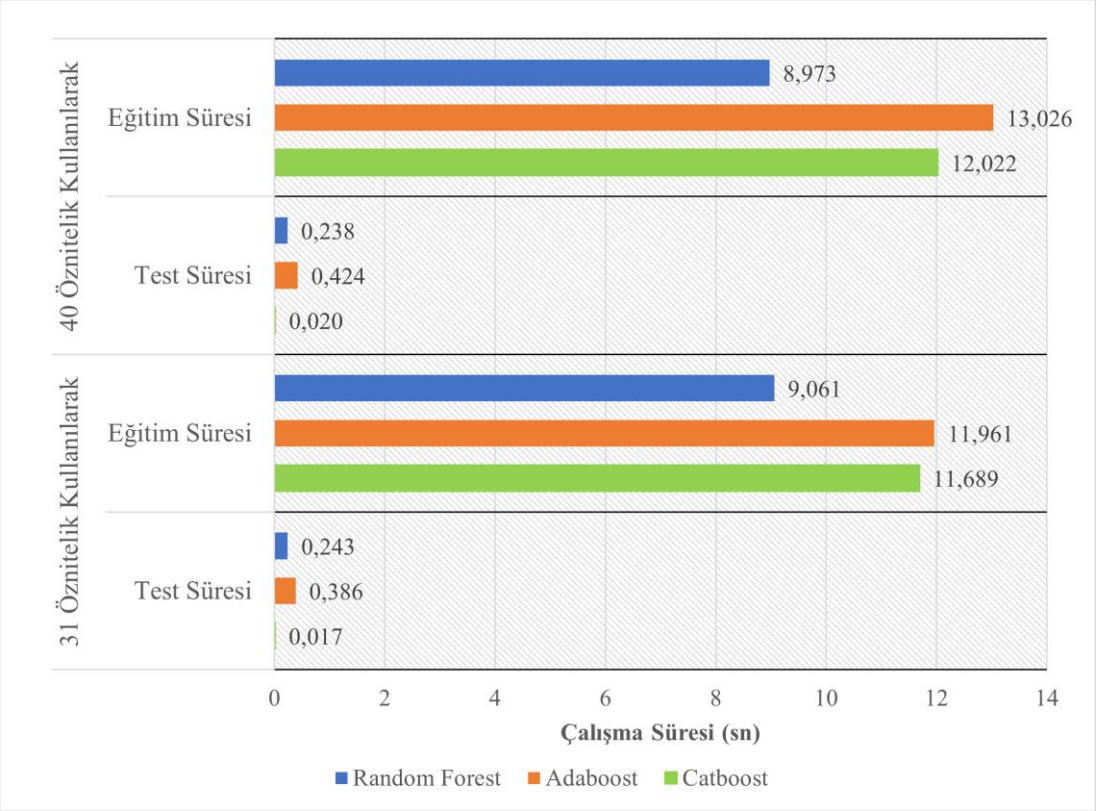
Tablo 3.11.'deki öznitelik seçimi yapılmış ve yapılmamış NSL-KDD veri kümelerinden KDDTest+ veri kümesi üzerindeki performans metrikleri değerlerinin görsel olarak incelenmesi Şekil 3.13.'te gösterilmektedir.



Şekil 3.13. Öznitelik seçimi yapılmış ve yapılmamış KDDTest+ veri kümesi üzerinde sınıflandırma algoritmalarının performans karşılaştırmaları

Tablo 3.11. ve Şekil 3.13.'te öznitelik seçimi yapılmış ve yapılmamış KDDTest+ veri kümesi üzerindeki sınıflandırma algoritmalarının performans metrikleri incelendiğinde bilgi kazancı yöntemi ile seçilen 31 özniteliğin AdaBoost algoritması hariç sınıflandırma performansına olumlu katkı sağladığı görülmektedir. Doğruluk (accuracy) metriği açısından bakıldığında öznitelik seçimi yapılmış ve yapılmamış veri kümelerinde de en başarılı algoritmanın CatBoost algoritması olduğu görülmektedir. Doğruluk (accuracy) metriği açısından bakıldığında önerilen model üzerinde uygulanan AdaBoost algoritmasının 2.3.2'deki öznitelik seçiminin avantajları ve dezavantajları kısmında bahsedilen öznitelik seçimi sınıflandırma performansına katkı sağlar maddesini yerine getirmediği gözlemlenmiştir.

Öznitelik seçimi yapılan ve yapılmamış veri kümeleri üzerindeki sınıflandırma algoritmalarının eğitim ve test süreleri Şekil 3.14.'te gösterilmiştir.



Şekil 3.14. Öznitelik seçimi yapılmış ve yapılmamış NSL-KDD veri kümeleri üzerindeki sınıflandırma algoritmalarının çalışma süreleri

Şekil 3.14.'e bakıldığında bilgi kazancı yöntemi kullanılarak öznitelik seçimi yapıldığında Random Forest algoritması hariç diğer sınıflandırma algoritmalarının eğitim ve test sürelerinde düşüş gözlenmektedir. Bu bulguya göre CatBoost algoritmasının bu algoritmalar arasında en hızlı test süresine sahip olduğu görülmektedir.

3.6.2. CatBoost algoritması bulgularının literatürdeki çalışmalarla karşılaştırılması

Bu bölümde bilgi kazancı yöntemi kullanılarak öznitelik seçimi yapılmış KDDTest+ veri kümesi üzerinde CatBoost sınıflandırma algoritması kullanılarak elde edilen ikili

sınıflandırma doğruluk (accuracy) değerinin saldırı tespit sistemleri literatüründeki çalışmalar ile karşılaştırılması Tablo 3.12.'de verilmektedir.

Tablo 3.12. CatBoost algoritmasının ve literatürdeki çalışmaların KDDTest+ veri kümesi üzerindeki ikili sınıflandırma doğruluk (accuracy) değerlerinin karşılaştırılması

(Khraisat ve ark., 2020)	SVM	% 69,52
(Pattawaro ve Polprasert, 2018)	K-Means+AdaBoost	% 72,64
	K-Means+Random Forest	% 75,67
(Khraisat ve ark., 2020)	Naïve Bayes	% 76,56
	Multi-Layer Perception	% 77,41
	KNN	% 79,40
	CART	% 80,30
	Random Forest	% 80,67
	C5 + OCSVM	% 83,24
(Pattawaro ve Polprasert, 2018)	K-Means+XGBoost	% 84,41
(Rai, 2020)	XGBoost	% 92,74
	CatBoost Algoritması	**% 79,43**

Tablo 3.12.'ye bakıldığında CatBoost algoritmasının doğruluk (accuracy) değeri literatürdeki birçok çalışmayı geride bırakmaktadır. Bununla birlikte en yüksek doğruluk (accuracy) değerinin Rai (2020) tarafından önerilen XGBoost algoritmasının olduğu görülmektedir.

BÖLÜM 4. SONUÇ VE ÖNERİLER

Günümüzde internet kullanımı, teknolojinin gelişmesiyle birlikte çok geniş bir alan bulmaktadır. Bu durum büyük miktarda verinin ortaya çıkmasına ve internet kullanılarak gerçekleştirilen saldırılarda artış yaşanmasına neden olmaktadır. Bu saldırıların önlenmesi amacıyla saldırı tespit sistemlerinin geliştirilmesi teknoloji hizmeti sunan organizasyonlar tarafından kaçınılmaz haline gelmektedir. Saldırı tespit sistemlerinin makine öğrenmesi ve derin öğrenme algoritmaları kullanılarak geliştirilmesi araştırmacılar tarafından ilgi gören bir konu olmaktadır. Dolayısıyla bu tez çalışmasında, saldırı tespit sisteminde makine öğrenimi yöntemlerinin uygulanması temel amaç ve motivasyon kaynağıdır.

Bu çalışmada makine öğrenmesi topluluk öğrenme yöntemlerinden CatBoost sınıflandırma algoritması kullanılarak ağ tabanlı saldırıları tespit sistemi önerilmiştir. Önerilen model üzerinde CatBoost algoritması uygulanmış ayrıca bu algoritma Random Forest ve AdaBoost sınıflandırma algoritmaları ile karşılaştırılmıştır. CatBoost algoritması ve çalışmada kullanılan diğer algoritmaların saldırı tespiti performansları NSL-KDD veri kümesi (KDDTrain+ ve KDDTest+) üzerinde veri ön işleme aşamalarından sonra bilgi kazancı yöntemi ile öznitelik seçimi yapılmış ve öznitelik seçimi yapılmamış veriler üzerinde ayrı ayrı gösterilmiştir.

Literatürdeki çalışmalar incelendiğinde NSL-KDD veri kümesinin içerisindeki saldırı türlerinin farklı sınıflar içerisinde yer aldığı anlaşılmıştır. Tang ve arkadaşları (2020), Verma ve arkadaşları (2018), Devan ve Khare (2020), Gadal ve Mokhtar (2017), El boujnouni ve Jedra (2018) gibi araştırmacıların çalışmalarında saldırı türlerinin başta U2R ve bununla birlikte R2L, Probe ve DoS sınıflarına kategorize edilirken farklılıklar tespit edilmiştir. Bu durum, saldırı tespitindeki sınıflandırma başarısının farklı sonuçlar doğurabileceğini düşündürmüştür.

NSL-KDD veri kümesi üzerinde veri ön işleme adımlarında etiket kodlama ve min-max ölçeklendirme işlemleri gerçekleştirilmiştir. Ardından bilgi kazancı yöntemi kullanılmış ve bu yöntemle öznitelik seçimi yapılan CatBoost sınıflandırma algoritmasının performansının olumlu yönde etkilendiği tespit edilmiştir. Bununla birlikte AdaBoost algoritmasının öznitelik seçimi yapılmış veriler üzerindeki sınıflandırma performansının düştüğü tespit edilmiştir. Buna ek olarak CatBoost ve AdaBoost algoritmalarının bilgi kazancı yöntemiyle öznitelik seçimi yapılan veriler üzerinde hesaplama sürelerinin hızlandığı, Random Forest algoritmasının ise yavaşladığı tespit edilmiştir.

CatBoost algoritmasının eğitiminde daha önce karşılaşmadığı saldırı türlerinde diğer sınıflandırma algoritmalarına göre daha başarılı sonuçlar verdiği görülmüştür. Bu sonuçlar KDDTest+ veri kümesi ile performans metrikleri kullanılarak tespit edilmiştir. Ayrıca CatBoost sınıflandırma algoritmasının diğer sınıflandırma algoritmalarından daha hızlı verileri test edebildiği tespit edilmiştir.

Elde edilen bulgular değerlendirildiğinde CatBoost algoritmasının, %79,43 doğruluk (accuracy), %68,44 kesinlik (precision), %80,24 F-ölçütü (f-measure) ve 0,9678 eğri altındaki alan (AUC) performans metrikleri değerleri ile diğer algoritmalara nazaran daha güvenilir sonuçlar verdiği anlaşılmıştır. Bu güvenilir çıktılardan doğruluk (accuracy) değeri literatürdeki NSL-KDD veri kümesinin (KDDTest+) kullanıldığı çalışmaların doğruluk (accuracy) değeri ile karşılaştırılmıştır. CatBoost algoritmasının literatürdeki NSL-KDD veri kümesini (KDDTest+) kullanan birçok çalışmadan doğruluk (accuracy) değerinin daha yüksek olduğu görülmüştür. Dolayısıyla saldırı tespitinde CatBoost algoritmasının daha iyi sonuçlar ortaya koyabileceği bu çalışmada anlaşılmıştır.

Bunlarla birlikte Kononenko ve Kukar (2007b) ile Bilgin (2018)'in çalışmalarında bahsedildiği gibi bu ampirik çalışmada da KDDTest+ veri kümesi üzerinde güçlendirme yönteminin torbalama yönteminden daha başarılı olduğu görülmüştür. Dorogush ve arkadaşlarının (2018) çalışmasında güçlendirme yöntemi algoritmalarını

karşılaştırdıklarındaki sonuçlara benzer şekilde, bu çalışmada CatBoost algoritmasının AdaBoost algoritmasından da hızlı olduğu bulunmuştur.

El Boujnouni ve Jedra (2018) çalışmasındaki bilgi kazancı yöntemi saldırıların tespitinde %5'lik bir başarı artışı sağlarken CatBoost algoritması üzerinde %0,01'lik bir başarı artışı göstermiştir. El Boujnouni ve Jedra saldırı tespitine Destek Vektör Makinesi yöntemiyle yaklaşmalarına karşın bu çalışmada karar ağaçları tabanlı yaklaşılmıştır ve bu nedenle başarı farkı olabileceği düşünülmektedir. Bu durumun sebepleri başka bir çalışmanın konusu olabilir.

Bu çalışmada CatBoost algoritmasının performansı varsayılan hiper parametreler ile gerçekleştirilmiştir. Gelecekte CatBoost algoritmasının hiper parametrelerin ağ saldırı tespit etme başarımlarına katkıları incelenebilir. Ayrıca CatBoost algoritmasının NSL-KDD veri kümesi yerine literatürdeki rüştünü kanıtlamış yeni ortaya çıkan veri kümeleriyle de başarımları değerlendirilebilir.

Diğer yandan bu çalışmada saldırıların tespiti ikili sınıflandırma yöntemi kullanılarak gerçekleştirilmiştir. Dolayısıyla çoklu sınıflandırma yöntemi bu çalışmanın kapsamı dışında kalmıştır. Araştırmacılar CatBoost algoritmasının farklı saldırı türleri tespit edebilme başarımlarını keşfetmek amacıyla çoklu sınıflandırma performansını da ölçebilirler. Böylece farklı öznitelik seçimi yöntemleri kullanılarak CatBoost algoritmasının ağ saldırı tespiti başarımları daha detaylı incelenebilir.

Bu tez akademik çalışmalara, CatBoost algoritması ele alınarak bir sınıflandırma algoritmasının makine öğrenimi yöntemiyle saldırı tespit sistemlerinde nasıl kullanılabileceği ile ilgili yol göstermektedir. Bununla birlikte makine öğrenimi gibi diğer derin öğrenme, yapay zekâ alanlarında, ayrıca sınıflandırmanın yanı sıra kümeleme ve diğer algoritma kategorileri açısından başta saldırı tespit sistemi olmak üzere bilişim sistemlerinde uygulanması için özgün ve yenilikçi rehber niteliği taşıyan bir yüksek lisans tezidir.

Nihayetinde bu tez çalışması Bilişim Sistemleri Mühendisliği alanında yeni yöntem ve teknolojilerin uygulanması ve karşılaştırılması için önemli bir kaynaktır. Dolayısıyla bilimsel araştırmalara katkısı olduğu gibi, bu akademik tez çalışması, özel sektör ve kamu kurumlarında da yeni teknolojilerin (makine öğrenimi, derin öğrenme, yapay zekâ vb.) hayata geçirilmesi amacıyla değerli bir rehber olarak ortaya çıkarılmıştır.

KAYNAKLAR

Ahmad, T. ve Aziz, M. N. (2019) "Data preprocessing and feature selection for machine learning intrusion detection systems", *ICIC Express Letters*, 13(2), ss. 93–101. doi: https//doi.org/10.24507/icicel.13.02.93.

Alamiedy, T. A., Anbar, M., Al-Ani, A. K., Al-Tamimi, B. N. ve Faleh, N. (2019) "Review on Feature Selection Algorithms for Anomaly-Based Intrusion Detection System", İçinde: *Advances in Intelligent Systems and Computing*. Springer International Publishing, ss. 605–619. doi: 10.1007/978-3-319-99007-1_57.

Alpaydın, E. (2013) *Yapay ögrenme*. 2. Basım. Boğaziçi Üniversitesi Yayınları.

Alpaydın, E. (2014a) "Combining Multiple Learners", İçinde: *Introduction To Machine Learning*. MIT Press, ss. 487–515.

Alpaydın, E. (2014b) "Introduction", İçinde: *Introduction to Machine Learning*. MIT Press, ss. 1–20.

Anderson, J. P. (1980) "Computer security threat monitoring and surveillance", *Technical Report, James P. Anderson Company.*, s. 56. doi: citeulike-article-id:592588.

Arslan, I. (2019) *Python ile Veri Bilimi*. 2. Baskı. İstanbul, Pusula 20 Teknoloji ve Yayıncılık.

Bhati, B. S., Chugh, G., Al-Turjman, F. ve Bhati, N. S. (2020) "An improved ensemble based intrusion detection technique using XGBoost", *Transactions on Emerging Telecommunications Technologies*, (April), ss. 1–15. doi: 10.1002/ett.4076.

Bhattacharya, S., Krishnan, S. R., Maddikunta, P. K. R., Kaluri, R., Singh, S., Gadekallu, T. R., Alazab, M. ve Tariq, U. (2020) "A Novel PCA-Firefly Based XGBoost Classification Model for Intrusion Detection in Networks Using GPU", *Electronics*, 9(2), s. 219. doi: 10.3390/electronics9020219.

Bilgin, M. (2018) *Makine Öğrenmesi*. 2. Basım. İstanbul, Papatya Yayıncılık.

El Boujnouni, M. ve Jedra, M. (2018) "New intrusion detection system based on support vector domain description with information gain metric", *International Journal of Network Security*, 20(1), ss. 25–34. doi: 10.6633/IJNS.201801.20(1).04.

Breiman, L. (1996) “Bagging predictors”, *Machine Learning*, 24(2), ss. 123–140. doi: 10.1007/BF00058655.

Breiman, L. (2001) “Random Forests”, *Machine learning*, 45(1), ss. 5–32. doi: https://doi.org/10.1023/A:1010933404324.

Breiman, L. ve Cutler, A. (2004) *Random forests - classification description*. Erişim Adresi: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm., Erişim Tarihi: 14 Kasım 2020.

Canadian Institute for Cybersecurity, U. N. B. (2009) *NSL-KDD dataset*. Erişim Adresi: https://www.unb.ca/cic/datasets/nsl.html., Erişim Tarihi: 29 Ekim 2020.

CatBoost (2020) *CatBoost - open-source gradient boosting library*. Erişim Adresi: https://catboost.ai/., Erişim Tarihi: 15 Kasım 2020.

Devan, P. ve Khare, N. (2020) “An efficient XGBoost–DNN-based classification model for network intrusion detection system”, *Neural Computing and Applications*. Springer London, 32(16), ss. 12499–12514. doi: 10.1007/s00521-020-04708-x.

Domo (2019) *Data Never Sleeps 7.0 Infographic*. Erişim Adresi: https://www.domo.com/learn/data-never-sleeps-7., Erişim Tarihi: 30 Kasım 2020.

Dorogush, A. V., Ershov, V. ve Gulin, A. (2018) “CatBoost: gradient boosting with categorical features support”, *arXiv*, ss. 1–7. Erişim Adresi: http://arxiv.org/abs/1810.11363.

Freund, Y. ve Schapire, R. E. (1997) “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”, *Journal of Computer and System Sciences*, 55(1), ss. 119–139. doi: 10.1006/jcss.1997.1504.

Gadal, S. M. A. M. ve Mokhtar, R. A. (2017) “Anomaly detection approach using hybrid algorithm of data mining technique”, İçinde: *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*. IEEE, ss. 1–6. doi: 10.1109/ICCCCEE.2017.7867661.

Hamzaçebi, Ç. (2011) *Yapay Sinir Ağları: Tahmin Amaçlı Kullanımı MATLAB ve Neurosolutions Uygulamalı*. Ekin Basım Yayın Dağıtım.

Haq, N. F., Onik, A. R. ve Shah, F. M. (2015) “An ensemble framework of anomaly detection using hybridized feature selection approach (HFSA)”, İçinde: *2015 SAI Intelligent Systems Conference (IntelliSys)*. IEEE, ss. 989–995. doi: 10.1109/IntelliSys.2015.7361264.

Horning, N. (2010) “Random Forests: An algorithm for image classification and generation of continuous fields data sets”, *Proceedings of the International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences, Osaka, Japan*, 911, ss. 1–6.

Hua, Y. (2020) “An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM”, İçinde: *2020 Information Communication Technologies Conference (ICTC)*. IEEE, ss. 125–130. doi: 10.1109/ICTC49638.2020.9123302.

Husain, A., Salem, A., Jim, C. ve Dimitoglou, G. (2019) “Development of an Efficient Network Intrusion Detection Model Using Extreme Gradient Boosting (XGBoost) on the UNSW-NB15 Dataset”, İçinde: *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, ss. 1–7. doi: 10.1109/ISSPIT47144.2019.9001867.

Jin, D., Lu, Y., Qin, J., Cheng, Z. ve Mao, Z. (2020) “KC-IDS : Multi-layer Intrusion Detection System”, İçinde: *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*. IEEE, ss. 1–5. doi: 10.1109/HPBDIS49115.2020.9130573.

Kaynar, O., Arslan, H., Görmez, Y. ve Işık, Y. E. (2018) “Makine Öğrenmesi ve Öznitelik Seçim Yöntemleriyle Saldırı Tespiti”, *Bilişim Teknolojileri Dergisi*, ss. 175–185. doi: 10.17671/gazibtd.368583.

Khafajeh, H. (2020) “An Efficient Intrusion Detection Approach Using Light Gradient Boosting”, *Journal of Theoretical and Applied Information Technology*, 98(5), ss. 825–835.

Khammassi, C. ve Krichen, S. (2017) “A GA-LR wrapper approach for feature selection in network intrusion detection”, *Computers & Security*. Elsevier Ltd, 70, ss. 255–277. doi: 10.1016/j.cose.2017.06.005.

Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J. ve Alazab, A. (2020) “Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine”, *Electronics*, 9(1), s. 173. doi: 10.3390/electronics9010173.

Kononenko, I. ve Kukar, M. (2007a) “Introduction”, İçinde: *Machine Learning and Data Mining*. Elsevier, ss. 1–36. doi: 10.1533/9780857099440.1.

Kononenko, I. ve Kukar, M. (2007b) “Machine Learning Basics”, İçinde: *Machine Learning and Data Mining*. Elsevier, ss. 59–105. doi: 10.1533/9780857099440.59.

Lantz, B. (2013) *Machine learning with R*. Birmingham, UK, Packt Publishing Ltd.

Lindner, C. (2017) “Automated Image Interpretation Using Statistical Shape Models”, İçinde: *Statistical Shape and Deformation Analysis*. Elsevier, ss. 3–32. doi: 10.1016/B978-0-12-810493-4.00002-X.

Malhotra, S., Bali, V. ve Paliwal, K. K. (2017) “Genetic programming and K-nearest neighbour classifier based intrusion detection model”, İçinde: *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*. IEEE, ss. 42–46. doi: 10.1109/CONFLUENCE.2017.7943121.

Nilsson, N. J. (1965) *Learning machines: foundations of trainable pattern-classifying systems*. New York, McGraw Hill.

Noorbehbahani, F., Fanian, A., Mousavi, R. ve Hasannejad, H. (2017) “An incremental intrusion detection system using a new semi-supervised stream classification method”, *International Journal of Communication Systems*, 30(4), s. e3002. doi: 10.1002/dac.3002.

O’Gorman, B., Wueest, C., O’Brien, D., Cleary, G., Lau, H., Power, J., Corpin, M., Cox, O., Wood, P. ve Wallace, S. (2019) *Internet Security Threat Report Vol. 24*. Erişim Adresi: https://docs.broadcom.com/doc/istr-24-2019-en., Erişim Tarihi: 02 Kasım 2020.

Öztemel, E. (2012) “Yapay Zeka ve Makine Öğrenmesine Genel Bakış”, İçinde: *Yapay Sinir Ağları*. 3. Basım. İstanbul: Papatya Yayıncılık, ss. 13–28.

Ozyer, G. T., Akbas, E. ve Vural, F. Y. (2006) “A Comparison of Features Spaces for Face Recognition Problem”, İçinde: *2006 IEEE 14th Signal Processing and Communications Applications*. IEEE, ss. 1–4. doi: 10.1109/SIU.2006.1659818.

Pattawaro, A. ve Polprasert, C. (2018) “Anomaly-Based Network Intrusion Detection System through Feature Selection and Hybrid Machine Learning Technique”, İçinde: *2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE)*. IEEE, ss. 1–6. doi: 10.1109/ICTKE.2018.8612331.

Polikar, R. (2006) “Ensemble based systems in decision making”, *IEEE Circuits and Systems Magazine*, 6(3), ss. 21–45. doi: 10.1109/MCAS.2006.1688199.

Rahmat, S., Niyaz, Q., Mathur, A., Sun, W. ve Javaid, A. Y. (2019) “Network Traffic-Based Hybrid Malware Detection for Smartphone and Traditional Networked Systems”, İçinde: *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, ss. 0322–0328. doi: 10.1109/UEMCON47517.2019.8992934.

Rai, A. (2020) “Optimizing a New Intrusion Detection System Using Ensemble Methods and Deep Neural Network”, İçinde: *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*. IEEE, ss. 527–532. doi: 10.1109/ICOEI48184.2020.9143028.

Samuel, A. L. (1959) “Some Studies in Machine Learning Using the Game of Checkers”, *IBM Journal of Research and Development*, 3(3), ss. 210–229. doi: 10.1147/rd.33.0210.

Schapire, R. E. (1990) “The Strength of Weak Learnability”, *Machine Learning*, 5(2), ss. 197–227. doi: 10.1023/A:1022648800760.

Shobha, G. ve Rangaswamy, S. (2018) “Machine Learning”, İçinde: *Handbook of Statistics*. 1. baskı. Elsevier B.V., ss. 197–228. doi: 10.1016/bs.host.2018.07.004.

Siddique, K., Akhtar, Z., Lee, H., Kim, W. ve Kim, Y. (2017) “Toward Bulk Synchronous Parallel-Based Machine Learning Techniques for Anomaly Detection in High-Speed Big Data Networks”, *Symmetry*, 9(9), s. 197. doi: 10.3390/sym9090197.

Simon, H. A. (1983) “WHY SHOULD MACHINES LEARN?”, İçinde: Michalski, R. S., Mitchell, T. M., ve Carbonell, J. G. (ed.) *Machine Learning: An Artificial Intelligence Approach*. Volume 1. Morgan Kaufmann, ss. 25–37. doi: 10.1016/B978-0-08-051054-5.50006-6.

Subasi, A. (2020) “Machine learning techniques”, İçinde: *Practical Machine Learning for Data Analysis Using Python*. Elsevier, ss. 91–202. doi: 10.1016/B978-0-12-821379-7.00003-5.

Sugiyama, M. (2016a) “Ensemble Learning”, İçinde: *Introduction to Statistical Machine Learning*. Elsevier, ss. 343–354. doi: 10.1016/B978-0-12-802121-7.00041-8.

Sugiyama, M. (2016b) “Statistical Machine Learning”, İçinde: *Introduction to Statistical Machine Learning*. Elsevier, ss. 3–8. doi: 10.1016/B978-0-12-802121-7.00012-1.

Sutton, C. D. (2005) “Classification and Regression Trees, Bagging, and Boosting”, İçinde: *Handbook of Statistics*. Elsevier Masson SAS, ss. 303–329. doi: 10.1016/S0169-7161(04)24011-1.

Talia, D., Trunfio, P. ve Marozzo, F. (2016) “Introduction to Data Mining”, İçinde: *Data Analysis in the Cloud*. Elsevier, ss. 1–25. doi: 10.1016/B978-0-12-802881-0.00001-9.

Tang, C., Luktarhan, N. ve Zhao, Y. (2020) “An Efficient Intrusion Detection Method Based on LightGBM and Autoencoder”, *Symmetry*, 12(9), s. 1458. doi: 10.3390/sym12091458.

Thoma, M. (2018) *The ROC space for a “better” and “worse” classfier.* Erişim Adresi: https://en.wikipedia.org/wiki/Receiver_operating_characteristic., Erişim Tarihi: 01 Kasım 2020.

Travallaee, M., Bagheri, E., Lu, W. ve Ghorbani, A. A. (2009) "A detailed analysis of the KDD CUP 99 data set", *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)*, (Cisda), ss. 1–6.

University of California, I. (1999) *KDD Cup 1999 Data*. Erişim Adresi: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html., Erişim Tarihi: 28 Ekim 2020.

Ünsalan, C. ve Erçil, A. (1998) "Öznitelik Seçme Yöntemlerinin Karşılaştırılması ve Başarı Kriteri", *Proceedings of IEEE SIU'98*, (May), ss. 60–65.

Vasudevan, A. R. ve Selvakumar, S. (2016) "Local outlier factor and stronger one class classifier based hierarchical model for detection of attacks in network intrusion detection dataset", *Frontiers of Computer Science*, 10(4), ss. 755–766. doi: 10.1007/s11704-015-5116-8.

Verma, P., Anwar, S., Khan, S. ve Mane, S. B. (2018) "Network Intrusion Detection Using Clustering and Gradient Boosting", İçinde: *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, ss. 1–7. doi: 10.1109/ICCCNT.2018.8494186.

Vieira, S., Lopez Pinaya, W. H. ve Mechelli, A. (2019) "Introduction to machine learning", İçinde: *Machine Learning: Methods and Applications to Brain Disorders*. Elsevier, ss. 1–20. doi: 10.1016/B978-0-12-815739-8.00001-8.

Wikipedia (2020) *Receiver operating characteristic*. Erişim Adresi: https://en.wikipedia.org/wiki/Receiver_operating_characteristic., Erişim Tarihi: 01 Kasım 2020.

Witten, I. H., Frank, E. ve Hall, M. A. (2011) "Ensemble Learning", İçinde: *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, ss. 351–373. doi: 10.1016/B978-0-12-374856-0.00008-0.

Wolpert, D. H. (1992) "Stacked generalization", *Neural Networks*, 5(2), ss. 241–259. doi: 10.1016/S0893-6080(05)80023-1.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z. Steinbach, M., Hand, D. J. ve Steinberg, D. (2008) "Top 10 algorithms in data mining", *Knowledge and Information Systems*, 14(1), ss. 1–37. doi: 10.1007/s10115-007-0114-2.

Yang, Y. (2017) "Ensemble Learning", İçinde: *Temporal Data Mining Via Unsupervised Ensemble Learning*. Elsevier, ss. 35–56. doi: 10.1016/B978-0-12-811654-8.00004-X.

Yılmaz, A. (2019) "Öğrenme", İçinde: Soylu, İ. ve Aksan, G. (ed.) *Yapay Zeka*. 6. Baskı. KODLAB Yayınevi, ss. 39–57.

EKLER

Ek 1: Python Kodları

```
'''
Created on Anıl KURT 2020
'''
import os
import time
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
from catboost import CatBoostClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification
_report, recall_score, precision_score, f1_score, roc_curve, auc
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import LabelEncoder, MinMaxScaler


def VeriAl_Train_Test_Birlikte_attack_normal():
    data = pd.read_csv("drive/Colab-Machine-
Learning/NSL_KDD99/KDDTrain_Test_Birlikte.txt",
                       encoding='utf-8')

    X, y = data.drop(columns=["class", "class_2"]), data["class"]

    SaldiriTurleri = ['back', 'land', 'neptune', 'pod', 'smurf', 'teardrop',
                      'apache2', 'mailbomb', 'processtable', 'udpstorm',
                      'worm', 'ipsweep', 'nmap', 'portsweep', 'satan',
                      'mscan', 'saint', 'buffer_overflow', 'loadmodule',
                      'perl', 'rootkit', 'ps', 'xterm', 'sqlattack',
                      'ftp_write', 'guess_passwd', 'imap', 'multihop', 'phf',
                      'spy', 'warezclient', 'warezmaster', 'httptunnel',
                      'named', 'sendmail', 'xlock', 'xsnoop', 'snmpgetattack',
                       'snmpguess']

    for i in SaldiriTurleri:
        y[y==i] = 'attack'

    return X, y
```

Ek 2: Python Kodları (Devam)

```
X_Train_Test_Birlikte_attack_normal, y_Train_Test_Birlikte_attack_normal = V
eriAl_Train_Test_Birlikte_attack_normal()

# Eğitim_Test_Birlikte veri kümesi içerisindeki kategorik sütunları buluyoru
z.
KategorikSutunlar_Train_Test_Birlikte_attack_normal = X_Train_Test_Birlikte_
attack_normal.loc[:, X_Train_Test_Birlikte_attack_normal.dtypes == np.object
].dropna()

def EtiketKodlama(X, KategorikSutun):
    '''
        Etiket Kodlama şemasını kullanarak kategorik verilerin dönüşümünü ge
rçekleştirdik.
        For döngüsü yapılabilir bütün kategorik veriler için
    :param X:
    :param KategorikSutunlar:
    :return:
    '''
    EtiketKodlamaSema = LabelEncoder()
    KategorikDegerlerinEtiketleri={}
    for i in KategorikSutun.columns:
        EtiketKodlamaSema.fit(X[i])
        KategorikDegerlerinEtiketleri[i] = list(EtiketKodlamaSema.classes_)
# sütun içindeki kategorik değerlerin 0'dan başlayıp devam eden kategorik sı
rasını alıyoruz.
        KategorikLabels = pd.Series(EtiketKodlamaSema.transform(X[i]))
        #KategorikLabelsList = {index: label for index, label in enumerate(E
tiketKodlamaSemasi.classes_)}
        X[i + "_LabelEncoder"] = KategorikLabels
        X = X.drop(columns=[i])
    return X, KategorikDegerlerinEtiketleri

# Eğitim veri kümesi X sütunu kategorik verileri EtiketKodlama haline dönüşt
ürdük.
X_Train_Test_Birlikte_attack_normal_Label_Hali, Train_Test_Birlikte_Kategori
ler = EtiketKodlama(X_Train_Test_Birlikte_attack_normal.copy(), KategorikSut
unlar_Train_Test_Birlikte_attack_normal)

EtiketKodlamaSema = LabelEncoder()
# Eğitim _Test_Birlikte veri kümesi Y sütununu 0-1 şekline getiriyoruz.
y_Train_Test_Birlikte_attack_normal_Label_Hali = EtiketKodlamaSema.fit_trans
form(y_Train_Test_Birlikte_attack_normal)

# Eğitim kümesini Etiket kodlama yapınca sütunların yeri değişmişti sırasısı
nı tekrar düzeltiyoruz.
Sutunlar = ['duration', 'protocol_type_LabelEncoder', 'service_LabelEncoder'
            , 'flag_LabelEncoder', 'src_bytes', 'dst_bytes', 'land',
            'wrong_fragment', 'urgent', 'hot', 'num_failed_logins',
            'logged_in', 'num_compromised', 'root_shell', 'su_attempted',
             'num_root', 'num_file_creations', 'num_shells',
            'num_access_files', 'num_outbound_cmds', 'is_host_login',
```

Ek 3: Python Kodları (Devam)

```
                'is_guest_login', 'count', 'srv_count', 'serror_rate',
                'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate',
                'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate',
                'dst_host_count', 'dst_host_srv_count',
                'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
                'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
                 'dst_host_serror_rate', 'dst_host_srv_serror_rate',
                'dst_host_rerror_rate', 'dst_host_srv_rerror_rate']

X_Train_Test_Birlikte_attack_normal_Label_Hali = X_Train_Test_Birlikte_attac
k_normal_Label_Hali[Sutunlar]

# Eğitim veri kümesinden 'num_outbound_cmds' sütununu çıkarttık.
X_Train_Test_Birlikte_attack_normal_Label_Hali = X_Train_Test_Birlikte_attac
k_normal_Label_Hali.drop(columns=['num_outbound_cmds'])

# Min-Max Ölçeklendirme uyguluyoruz.
def MinMaxOlceklendirme(X):
    '''
    MinMax ölçeklendirme uyguluyoruz.

    Açıklama:
     MinMaxScaler, veri kümesini, tüm öznitelik değerlerini [0, 1] aralığınd
a olacak şekilde yeniden ölçeklendirir.
    :param X:
    :return:
    '''
    MinMaxOlcek = MinMaxScaler()
    MinMaxCikti = pd.DataFrame(MinMaxOlcek.fit_transform(X), columns=X.colum
ns)
    return MinMaxCikti

X_Train_Test_Birlikte_attack_normal_Label_Hali_MinMax = MinMaxOlceklendirme(
X_Train_Test_Birlikte_attack_normal_Label_Hali.copy())

# Veri Kümesini X_train, X_test, y_train, y_test şeklinde ayırıyoruz.
X_train = X_Train_Test_Birlikte_attack_normal_Label_Hali_MinMax.iloc[:125973
]
X_test = X_Train_Test_Birlikte_attack_normal_Label_Hali_MinMax.iloc[125973:]
X_test.reset_index(drop=True, inplace=True)
y_train = y_Train_Test_Birlikte_attack_normal_Label_Hali[:125973]
y_test = y_Train_Test_Birlikte_attack_normal_Label_Hali[125973:]

# 40 öznitelik kullanarak sınıflandırma algoritmalarını eğitip performans
metriklerini hesaplıyoruz.
def ModelFonksiyonu(CatBoostSinif, RandomForestSinif, AdaBoostSinif, X_train
, y_train, X_test, y_test):
    Algoritmalar = {"CatBoost": CatBoostSinif, "RandomForest": RandomForestS
inif, "AdaBoost": AdaBoostSinif}
    n = 0
```

Ek 4: Python Kodları (Devam)

```
    colors = ['r', 'y', 'c']
    lw = 2
    Result = {}
    plt.figure()
    for i in Algoritmalar:
        # Test Veri Kümesi
        StartTime = time.clock()
        Algoritmalar[i].fit(X_train, y_train)
        EndTime = time.clock()
        EgitimSuresi = EndTime - StartTime
        StartTime = time.clock()
        y_pred = Algoritmalar[i].predict(X_test)
        EndTime = time.clock()
        TestSuresi = EndTime - StartTime
        y_pred_ = Algoritmalar[i].predict_proba(X_test)[:,1]
        Test_BasariPuani = accuracy_score(y_test, y_pred)
        Test_F1Skoru = f1_score(y_test, y_pred)
        Test_RecallSkoru = recall_score(y_test, y_pred)
        Test_PrecisionSkoru = precision_score(y_test, y_pred)
        Test_ClassificationRaporu = classification_report(y_test, y_pred, ta
rget_names=['attack', 'normal'])
        Test_KarisiklikMatrisi = confusion_matrix(y_test, y_pred)

        # Eğitim Veri Kümesi
        Egitim_y_pred = Algoritmalar[i].predict(X_train)
        Egitim_y_pred_ = Algoritmalar[i].predict_proba(X_train)[:,1]
        Egitim_BasariPuani = accuracy_score(y_train, Egitim_y_pred)
        Egitim_F1Skoru = f1_score(y_train, Egitim_y_pred)
        Egitim_RecallSkoru = recall_score(y_train, Egitim_y_pred)
        Egitim_PrecisionSkoru = precision_score(y_train, Egitim_y_pred)
        Egitim_ClassificationRaporu = classification_report(y_train, Egitim_
y_pred, target_names=['attack', 'normal'])
        Egitim_KarisiklikMatrisi = confusion_matrix(y_train, Egitim_y_pred)

        # Test Veri Kümesi ROC Grafiği

        fpr, tpr, _ = roc_curve(y_test, y_pred_)

        plt.plot(fpr, tpr, color=colors[n], lw=lw, label='ROC curve for {0},
 area = {1:.4f}'.format(i, auc(fpr, tpr), alpha=0.5))
        plt.fill_between(fpr, tpr, y2=0, color=colors[n], alpha=0.1)

        n = n + 1

        ResultsDict = {"Test_Basari_Puani(Accuracy)_"+str(i): Test_BasariPua
ni,
                       "Test_Classification_Raporu_"+str(i): Test_Classific
ationRaporu,
                       "Test_Precision_Skoru_"+str(i): Test_PrecisionSkoru,
                       "Test_Recall_Skoru_"+str(i): Test_RecallSkoru,
                       "Test_F1_Skoru_"+str(i): Test_F1Skoru,
```

Ek 5: Python Kodları (Devam)

```
                        "Test_Confusion_Matrisi_"+str(i): Test_KarisiklikMat
risi,
                        "TestEgitim_Suresi_"+str(i): EgitimSuresi,
                        "Test_Test_Suresi_"+str(i): TestSuresi,
                        "Oznitelik_Sayisi_"+str(i): len(X_train.columns),
                        "Egitim_Basari_Puanı_"+str(i): Egitim_BasariPuani,
                        "Egitim_ClassificationRaporu_"+str(i): Egitim_Classi
ficationRaporu,
                        "Egitim_Precision_skoru_"+str(i): Egitim_PrecisionSk
oru,
                        "Egitim_Recall_skoru_"+str(i): Egitim_RecallSkoru,
                        "Egitim_F1_skoru_"+str(i): Egitim_F1Skoru,
                        "Egitim_KarisiklikMatrisi_"+str(i): Egitim_Karisikli
kMatrisi
                        }
        Result.update(ResultsDict)

    FilenameRocCurve = 'drive/Colab-Machine-
Learning/NSL_KDD99/40_Oznitelik_Roc_Curve_Egrisi.png'
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([-0.02, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('İşlem Karakteristik Eğrisi')
    plt.legend(loc="lower right")
    plt.savefig(FilenameRocCurve, dpi=300)
    return Result

def Yazdir(TumSonuclar):
    TumSonuclarDF = pd.DataFrame(TumSonuclar)
    DatasetResultsName = "Oznitelik_Secimsiz_Normal_Attack_Sonuc.csv"
    Path = "drive/Colab-Machine-Learning/NSL_KDD99/"
    ResultsPath = os.path.join(Path, DatasetResultsName)
    TumSonuclarDF.to_csv(ResultsPath, index=False)

TumSonuclar = []

AdaBoostS= AdaBoostClassifier(random_state=0, n_estimators=100, learning_rat
e=1.0)
RandomforestS = RandomForestClassifier(random_state=0, n_estimators=100)
CatBoostS = CatBoostClassifier(loss_function = "Logloss", random_seed = 0, n
_estimators = 100, learning_rate = 0.03, l2_leaf_reg = 3)

Sonuc = ModelFonksiyonu(CatBoostS, RandomforestS, AdaBoostS, X_train, y_trai
n, X_test, y_test)
TumSonuclar.append(Sonuc)
Yazdir(TumSonuclar)
```

Ek 6: Python Kodları (Devam)

```
# Bilgi kazancı yöntemi uygulayarak öznitelik seçimi yapıyoruz.
BilgiKazancı = mutual_info_classif(X_train, y_train, random_state=0)
BilgiKazancıNumpy = np.split(BilgiKazancı, indices_or_sections=40, axis=0)

BilgiKazancıDF = pd.Series(data=BilgiKazancıNumpy, index=X_Train_Test_Birlik
te_attack_normal_Label_Hali.columns)
DatasetResultsName = "Bilgi_Kazancı_Degeri.xlsx"
Path = "drive/Colab-Machine-Learning/NSL_KDD99/"
ResultsPath = os.path.join(Path, DatasetResultsName)
BilgiKazancıDF.sort_values(ascending=False).to_excel(ResultsPath, index=True
)
CikacakOznitelikIsimleri = BilgiKazancıDF[BilgiKazancıDF < 0.001].index

# Bilgi kazancı değeri 0,001'den az olan öznitelikleri kaldırıyoruz.
X_train.drop(columns=CikacakOznitelikIsimleri, inplace=True)
X_test.drop(columns=CikacakOznitelikIsimleri, inplace=True)

# 31 öznitelik içeren veri kümemiz üzerinde sınıflandırma algoritmalarını
eğitip performans metriklerini hesaplıyoruz.
def ModelFonksiyonu(CatBoostSinif, RandomForestSinif, AdaBoostSinif, X_train
, y_train, X_test, y_test):
    Algoritmalar = {"CatBoost": CatBoostSinif, "RandomForest": RandomForestS
inif, "AdaBoost": AdaBoostSinif}
    n = 0
    colors = ['r', 'y', 'c']
    lw = 2
    Result = {}
    plt.figure()
    for i in Algoritmalar:
        # Test Veri Kümesi
        StartTime = time.clock()
        Algoritmalar[i].fit(X_train, y_train)
        EndTime = time.clock()
        EgitimSuresi = EndTime - StartTime
        StartTime = time.clock()
        y_pred = Algoritmalar[i].predict(X_test)
        EndTime = time.clock()
        TestSuresi = EndTime - StartTime
        y_pred_ = Algoritmalar[i].predict_proba(X_test)[:,1]
        Test_BasariPuani = accuracy_score(y_test, y_pred)
        Test_F1Skoru = f1_score(y_test, y_pred)
        Test_RecallSkoru = recall_score(y_test, y_pred)
        Test_PrecisionSkoru = precision_score(y_test, y_pred)
        Test_ClassificationRaporu = classification_report(y_test, y_pred, ta
rget_names=['attack', 'normal'])
        Test_KarisiklikMatrisi = confusion_matrix(y_test, y_pred)

        # Eğitim Veri Kümesi
        Egitim_y_pred = Algoritmalar[i].predict(X_train)
        Egitim_y_pred_ = Algoritmalar[i].predict_proba(X_train)[:,1]
```

Ek 7: Python Kodları (Devam)

```
        Egitim_BasariPuani = accuracy_score(y_train, Egitim_y_pred)
        Egitim_F1Skoru = f1_score(y_train, Egitim_y_pred)
        Egitim_RecallSkoru = recall_score(y_train, Egitim_y_pred)
        Egitim_PrecisionSkoru = precision_score(y_train, Egitim_y_pred)
        Egitim_ClassificationRaporu = classification_report(y_train, Egitim_
y_pred, target_names=['attack', 'normal'])
        Egitim_KarisiklikMatrisi = confusion_matrix(y_train, Egitim_y_pred)

        # Test Veri Kümesi ROC Grafiği

        fpr, tpr, _ = roc_curve(y_test, y_pred_)

        plt.plot(fpr, tpr, color=colors[n], lw=lw, label='ROC curve for {0},
 area = {1:.4f}'.format(i, auc(fpr, tpr), alpha=0.5))
        plt.fill_between(fpr, tpr, y2=0, color=colors[n], alpha=0.1)

        n = n + 1

        ResultsDict = {"Test_Basari_Puani(Accuracy)_"+str(i): Test_BasariPua
ni,
                       "Test_Classification_Raporu_"+str(i): Test_Classific
ationRaporu,
                       "Test_Precision_Skoru_"+str(i): Test_PrecisionSkoru,
                       "Test_Recall_Skoru_"+str(i): Test_RecallSkoru,
                       "Test_F1_Skoru_"+str(i): Test_F1Skoru,
                       "Test_Confusion_Matrisi_"+str(i): Test_KarisiklikMat
risi,
                       "TestEgitim_Suresi_"+str(i): EgitimSuresi,
                       "Test_Test_Suresi_"+str(i): TestSuresi,
                       "Oznitelik_Sayisi_"+str(i): len(X_train.columns),
                       "Egitim_Basari_Puanı_"+str(i): Egitim_BasariPuani,
                       "Egitim_ClassificationRaporu_"+str(i): Egitim_Classi
ficationRaporu,
                       "Egitim_Precision_skoru_"+str(i): Egitim_PrecisionSk
oru,
                       "Egitim_Recall_skoru_"+str(i): Egitim_RecallSkoru,
                       "Egitim_F1_skoru_"+str(i): Egitim_F1Skoru,
                       "Egitim_KarisiklikMatrisi_"+str(i): Egitim_Karisikli
kMatrisi
                       }
        Result.update(ResultsDict)

    FilenameRocCurve = 'drive/Colab-Machine-
Learning/NSL_KDD99/31_Oznitelik_Roc_Curve_Egrisi.png'
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([-0.02, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('İşlem Karakteristik Eğrisi')
```

Ek 8: Python Kodları (Devam)

```
    plt.legend(loc="lower right")
    plt.savefig(FilenameRocCurve, dpi=300)
    return Result

def Yazdir(TumSonuclar):
    TumSonuclarDF = pd.DataFrame(TumSonuclar)
    DatasetResultsName = "Oznitelik_Secimli_Normal_Attack_Sonuc.csv"
    Path = "drive/Colab-Machine-Learning/NSL_KDD99/"
    ResultsPath = os.path.join(Path, DatasetResultsName)
    TumSonuclarDF.to_csv(ResultsPath, index=False)

TumSonuclar = []

AdaBoostS= AdaBoostClassifier(random_state=0, n_estimators=100, learning_rat
e=1.0)
RandomforestS = RandomForestClassifier(random_state=0, n_estimators=100)
CatBoostS = CatBoostClassifier(loss_function = "Logloss", random_seed = 0, n
_estimators = 100, learning_rate = 0.03, l2_leaf_reg = 3)

Sonuc = ModelFonksiyonu(CatBoostS, RandomforestS, AdaBoostS, X_train, y_trai
n, X_test, y_test)
TumSonuclar.append(Sonuc)
Yazdir(TumSonuclar)
```

ÖZGEÇMİŞ

Anıl Kurt, 1994 yılında Bilecik'te doğmuştur. İlk, orta ve lise eğitimini Bilecik'te tamamladı. 2012 yılında Ertuğrul Gazi Lisesinden mezun oldu. 2012 yılında Bartın Üniversitesi Yönetim Bilişim Sistemleri Bölümü'ne başladı. 2014 yılında Sakarya Üniversitesi Yönetim Bilişim Sistemleri Bölümü'ne yatay geçiş yaptı ve 2017 yılında bitirdi. 2017 yılında başladığı Sakarya Üniversitesi Bilişim Sistemleri Mühendisliği Bölümü yüksek lisans eğitimine devam etmektedir.