Bandırma Onyedi Eylül Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Yazılım Mühendisliği Bölümü 2023-2024 Bahar Dönemi, Veri Madenciliği Ödevi

Merve Bağışlar[1]

[1] *2111505050*

Özet— Siber saldırı tespiti, geçmiş bilgilerden esinlenerek gelecekteki olası saldırıları belirlemeye yönelik işlemlerdir. Bu çalışmada, ağ trafiği ve kullanıcı davranışları gibi 45 farklı özellik üzerine odaklanılarak siber saldırı tespiti yapılmaktadır. Çalışmada siber saldırı davranışları öğrenilerek tahminlerde bulunulmuştur. Modelin etkinliği ve dayanıklılığını değerlendirmek için çeşitli ağlardan alınan saldırı verilerinin farklı parametreleri denenmiştir. Bu çalışmadaki siber saldırı parametreleri; KNN, Decision Tree, Random Forest modelleri aracılığıyla sürdürülebilir sonuçlara ulaşmak için toplanmış, eğitilmiş ve test edilmiştir. Eğitim ve testler sonucunda elde edilen siber saldırı tespitleri, modelin doğruluğunu saptamak için gerçek saldırı verileriyle karşılaştırılmıştır. Bu çalışmanın sonuçları, modelin belirli parametrelerle siber saldırı verilerini tahmin etmede başarılı olduğunu ortaya koymuştur. Çalışmanın sonucunda elde edilen veriler, gerçek veriler ile tahmin edilen sonuç verilerini karşılaştırarak minimum hata ile maksimum doğruluk sağlanmaya çalışılmıştır.

Anahtar Kelime—tahmin; siber saldırı parametreleri; eğitim; test; siber saldırı tespiti.

I. Giriş

Siber güvenlik, modern dijital dünyada en kritik konular arasında yer almaktadır. Günümüzde, dijital sistemler ve ağlar üzerinde gerçekleşen siber saldırıların sayısı ve karmaşıklığı hızla artmaktadır. Bu saldırıların tespiti ve önlenmesi, hem bireysel kullanıcılar hem de kurumlar için hayati önem taşımaktadır. Geleneksel siber güvenlik yöntemleri, saldırıların sürekli değişen doğası ve yüksek hacimli veri akışları nedeniyle yetersiz kalabilmektedir. Bu nedenle, makine öğrenimi gibi ileri teknolojiler, siber saldırıların daha etkin ve verimli bir şekilde tespit edilmesi için kullanılmaktadır. Makine öğrenimi, çeşitli değişkenler arasındaki ilişkileri tanımlayarak potansiyel saldırıları belirlemek için ağ verilerini analiz eder. Bu yaklaşımla, siber saldırı tespitinin doğruluğu ve verimliliği artırılabilir. Makine öğrenimi algoritmalarının siber güvenlik alanında uygulanması, dijital ağların ve sistemlerin güvenliğini büyük ölçüde artırabilir.

Bu çalışmada, Yeni Güney Galler Üniversitesi'nin (Avustralya) Cyber Range Lab'sinde yakalanan ağ paketleri kullanılarak siber saldırı tespiti üzerine odaklanılmaktadır. Veri seti, dokuz farklı türde siber saldırıyı içermekte olup, eğitim ve test verisi olmak üzere iki set halinde sunulmuştur. Çalışmada kullanılan saldırı türleri şunlardır: Fuzzers, Analysis, Backdoors, DoS (Denial of Service), Exploits, Generic, Reconnaissance, Shellcode ve Worms. Bu geniş saldırı yelpazesi, potansiyel güvenlik tehditlerine karşı kapsamlı bir koruma sağlamak için önemlidir. Araştırmada, karar ağacı (Decision Tree), rastgele orman (Random Forest) ve k-en yakın komşu (K-Nearest Neighbors, KNN) gibi makine öğrenimi algoritmaları kullanılarak siber saldırıların tespiti hedeflenmiştir. Bu algoritmalar, farklı veri işleme teknikleri ve parametrelerle eğitilerek, en yüksek doğruluğa ulaşmak amacıyla optimize edilmiştir. Çalışmanın amacı, siber saldırı davranışlarını öğrenerek gelecekteki olası saldırıları tahmin etmek ve bu sayede dijital sistemlerin güvenliğini sağlamaktır.

Bu çalışmada eğitim ve test veri setleri birleştirilerek, önişleme aşamaları uygulanmış ve kategorik veriler etiketlenmiştir. Model performanslarını değerlendirmek için doğruluk, hatırlama (recall) ve kesinlik (precision) metrikleri kullanılmıştır.

Çalışmanın sonuçları, siber saldırı tespitinde bu algoritmaların ne kadar etkili olduğunu ortaya koymaktadır. Ayrıca, farklı saldırı türlerinin tespitindeki başarı oranları ve en iyi performans gösteren modellerin belirlenmesi hedeflenmiştir. Bu şekilde, siber güvenlik alanında daha güçlü ve güvenilir çözümler geliştirilmesi amaçlanmaktadır..

İletişim maili:
Merve,Bağışlar, mervebagislar@ogr.bandirma.edu.tr

II. Veri setinin temini

Bu çalışmada, siber saldırı çeşitlerine bağlı olarak 45 değişkenden training veri setinde 82,332 satır ,testing veri setinde ise 175,341 satır oluşan bir veri seti kullanılmıştır. Çalışmada kullanılan veri seti; [3] kaggle adresinden alınmıştır. Kullanılan veri seti, siber saldırıları etkileyebilecek id, proto, state gibi 45 adet nitelik bulunmaktadır. Veri setinin %70'i eğitim, %30'u ise test verisi olarak ayrılmıştır.

.

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate
0	0.000011	udp	-	INT	2	0	496	0	90909.090200
1	0.000008	udp	-	INT	2	0	1762	0	125000.000300
2	0.000005	udp	-	INT	2	0	1068	0	200000.005100
3	0.000006	udp	-	INT	2	0	900	0	166666.660800
4	0.000010	udp	-	INT	2	0	2126	0	100000.002500
5	0.000003	udp	-	INT	2	0	784	0	333333.321500
6	0.000006	udp	-	INT	2	0	1960	0	166666.660800
7	0.000028	udp	-	INT	2	0	1384	0	35714.285220
8	0.000000	arp	-	INT	1	0	46	0	0.000000
9	0.000000	arp	-	INT	1	0	46	0	0.000000

Şekil. 1: Veri Seti

III. Kullanılan Yöntemlerin Teorik Bilgisi

a. K-En Yakın Komşu ((K-Nearest Neighbour-KNN))

K-En Yakın Komşu (KNN) algoritması, denetimli öğrenme yöntemlerinden biridir ve hem sınıflandırma hem de regresyon problemlerinde kullanılabilir. KNN algoritması, bir veri noktasının sınıfını veya değerini tahmin etmek için en yakın K adet komşusunun etiketlerine veya değerlerine bakar.
Genellikle, en yakın gözlemler, göz önünde bulundurulan veri noktasına en küçük Öklid mesafesine sahip olanlar olarak tanımlanır. Öklid mesafesinin matematiksel ifadesi denklem 1'de verilmiştir.

$$d = \sqrt{\sum_{i=1}^{k}(X_i - Y_i)^2} \quad (1)$$

İşleyişi basit olmakla birlikte, genellikle şu adımları içerir:

1. Veri Kümesinin Hazırlanması: Veri kümesi, eğitim ve test setlerine ayrılır. Eğitim seti modelin öğrenmesi, test seti ise performans değerlendirmesi için kullanılır.

2. Mesafe Hesaplama: Yeni bir veri noktasının, eğitim kümesindeki tüm noktalarla olan mesafesi hesaplanır. Genellikle Öklidyen mesafesi kullanılır.

3. Komşuların Belirlenmesi: Mesafeler sıralanır ve en küçük mesafeye sahip K komşu seçilir.

4. Sınıflandırma veya Regresyon: K komşunun sınıf etiketlerine bakılarak yeni veri noktasının sınıfı belirlenir (sınıflandırma) veya komşuların değerlerinin ortalaması alınarak yeni veri noktasının değeri tahmin edilir (regresyon).

b. Rastgele Orman Algoritması (Random Forest -RF)

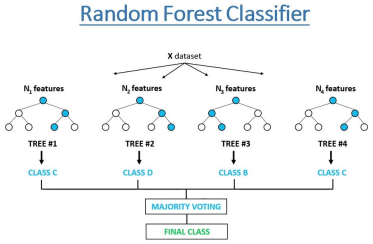
Rastgele Orman (Random Forest - RF) algoritması, güçlü ve esnek bir denetimli öğrenme yöntemidir. Sınıflandırma ve regresyon problemlerinde yaygın olarak kullanılır. Birçok karar ağacının oluşturulup birleştirilmesiyle çalışır ve her bir ağacın tahminlerinin çoğunluğuna veya ortalamasına bakarak nihai sonucu belirler.Rastgele Orman algoritmasının temel çalışma prensibi aşağıdaki adımlarla özetlenebilir:

1. Veri Setinin Rastgele Alt Kümesi ve Özelliklerin Rastgele Alt Kümesi:Eğitim verisinden, yerleştirmeli örnekleme yöntemiyle birçok farklı alt küme oluşturulur. Her bir alt küme, belirli bir sayıda özelliğin rastgele seçilmesiyle oluşturulur.

2. Karar Ağaçlarının Oluşturulması: Oluşturulan her bir alt küme, bir karar ağacı modeli eğitmek için kullanılır.Karar ağacının her bir düğümünde, belirli bir sayıda rastgele seçilmiş özellik arasından en iyi bölme bulunur.

3. Tahminlerin Birleştirilmesi: Sınıflandırma problemlerinde, her bir karar ağacının oyuyla sınıf tahmini yapılır ve çoğunluk oyu nihai sınıfı belirler.Regresyon problemlerinde ise, tüm ağaçların tahminlerinin ortalaması alınarak son tahmin yapılır.

Şekil 2 'de Rastgele orman algoritmasını anlatan bir görsel verilmiştir.



Şekil. 2: Rastgele Orman

Random Forest, diğer algoritmalara göre daha az eğitim süresi gerektirir, büyük veri kümelerinde çıktıyı yüksek doğrulukla tahmin eder ve verilerin büyük bir kısmı eksik olduğunda da doğruluğunu korur. Bu sebeplerden tercih edilebilmektedir.

c. Decision Tree

Karar ağacı (Decision Tree), makine öğrenimi ve veri madenciliğinde kullanılan, karar verme süreçlerini modelleyen bir sınıflandırma ve regresyon tekniğidir. Karar ağacı, veriyi belirli kurallara göre bölerek dallanma yapısı oluşturur ve bu dallanmalar sonucunda tahminlerde bulunur. Ağaç yapısının her düğümü bir özellik (attribute) ya da karar noktasını, her dalı ise bu karar sonucunda olası sonuçları temsil eder. Karar ağaçları, hem açıklanabilirlik hem de yorumlanabilirlik açısından avantajlıdır.
Karar ağacı algoritması, veriyi en iyi şekilde nasıl böleceğine karar vermek için belirli kriterler kullanır. Bu kriterler arasında şunlar bulunur:

- Gini İndeksi:Gini indeksi, veri setindeki saflığı ölçmek için kullanılan bir metriktir.Düşük Gini indeksi, düğümdeki örneklerin daha homojen olduğunu gösterir.

$$Gini = 1 - \sum_{i=1}^{C}(p_i)^2$$

Şekil. 3: Gini Formülü

- Bilgi Kazancı (Information Gain):Bilgi kazancı, entropi kavramına dayanır ve bir bölmenin veri setinin düzensizliğini ne kadar azalttığını ölçer.Yüksek bilgi kazancı, iyi bir bölmenin göstergesidir.

- Kikare (Chi-Square):Kikare testi, kategorik verilerde beklenen ve gözlenen frekanslar arasındaki farkı ölçmek için kullanılır.Bu fark, veri setini bölerken kullanılır.

d. Performans Ölçütleri

1. Karmaşıklık Matrisi

Karışıklık matrisi, test verisi için sınıflandırma modellerinin performansını belirlemede kullanılır. Sadece test verilerinin gerçek değerleri biliniyorsa belirlenebilir. Bu matrisin dört parametresi vardır. Bu parametreler, Doğru Pozitif(TP),Doğru Negatif (TN),Yanlış Pozitif(FP) ve Yanlış Negatiftir(FN).

- **Accuary(Doğruluk)**: Doğru tahminlerin tüm tahminlere oranıdır. $(TN+TP)/(TP+FP+TN+FN)$

- **Precision (Kesinlik)**: Pozitif tahmin edilen durumdaki başarıyı gösterir. $TP/(TP+FP)$

- **Recall(Hassasiyet)**: Pozitif durumların ne kadar başarılı tahmin edildiğidir. $TP/(TP+FN)$

- **Specificity(Özgüllük)**:Negatif durumların ne kadar başarılı tahmin edildiğidir. $TN/(TN+FP)$

2. Roc Eğrisi

AUC-ROC eğrisi, farklı eşik değerlerinde bir sınıflandırma modelinin performans ölçüm metriğidir. TPR ve FPR parametreleri ile elde edilir.

IV. Önerilen Yöntem ve Deneysel Sonuçlar

Projede kullanılacak yöntemler seçilip araştırıldıktan sonra Google COLAB ortamında kod yazılmaya başlandı. Öncelikle veri seti COLAB ortamında açıldı. Kütüphanelerin içe aktarılması, veri setinin yüklenmesi için ve veri setinin görselleştirilmesi için

```
!pip install nbformat
!pip install graphviz
!pip install dtreeviz
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import dtreeviz
```

komutları kullanıldı. Aşağıdaki kod satırında sırası ile ilk önce veri seti okunur, okunan veri setinin kaç satır kaç sütundan oluştuğu öğrenildi.olarak Veri setlerinin boyutlarını ve sütun isimlerinin aynı olup olmadığı kontrol edildi.

```
training = pd.read_csv("/content/
    UNSW_NB15_training-set.csv.zip")
testing = pd.read_csv("/content/UNSW_NB15_testing-
    set.csv.zip")
print("training ",training.shape)
print("testing ",testing.shape)
all(training.columns == testing.columns)
```

Eğitim ve test veri setlerini birleştirerek tek bir veri çerçevesi oluşturuyoruz. id sütununu çıkararak indeksleri sıfırdan başlatıyoruz.Veri setinin ilk 5 satırını Şekil 4'de görüldüğü gibi görüntülüyoruz.

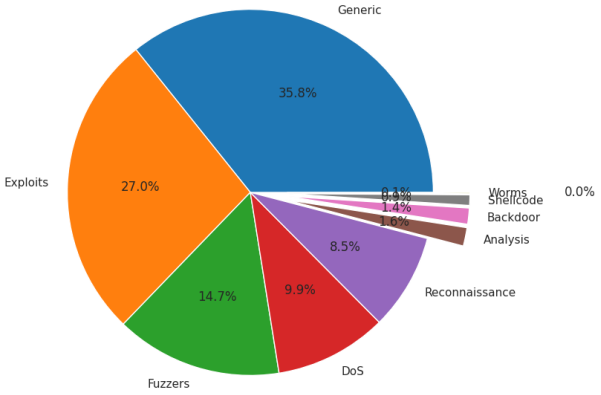
	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cm
0	0.000011	udp	-	INT	2	0	496	0	90909.0902	254	...	1	2	0	
1	0.000008	udp	-	INT	2	0	1762	0	125000.0003	254	...	1	2	0	
2	0.000005	udp	-	INT	2	0	1068	0	200000.0051	254	...	1	3	0	
3	0.000006	udp	-	INT	2	0	900	0	166666.6608	254	...	1	3	0	
4	0.000010	udp	-	INT	2	0	2126	0	100000.0025	254	...	1	3	0	

Şekil. 4: Veri Setinin İlk 5 Satırı

Proto, service, ve state sütunlarını kategorik veri türüne dönüştürüp label encoding uyguluyoruz.Saldırı kategorilerinin dağılımını görselleştiriyoruz.

```
for col in ['proto', 'service', 'state']:
df[col] = df[col].astype('category').cat.codes
df['attack_cat'] = df['attack_cat'].astype('
    category')
validAttacks = df[df['label']==1]['attack_cat'].
    value_counts()
print(validAttacks)
plt.figure(figsize = (15,8))
plt.pie(validAttacks,labels = validAttacks.index,
    autopct = '%1.1f%%',explode =
    [0,0,0,0,0,0.2,0.2,0.2,0.2,1.2])
plt.show()
```

Görselleştirdiğimiz saldırı kategorilerinin dağılımını Şekil 5'deki gibi görüntülüyoruz



Şekil. 5: Saldırı Kategorilerinin Dağılımı

Veri setini eğitim ve test alt kümelerine ayırıyoruz. Veriyi %70 ve %30 olarak bölüyoruz.

```
from sklearn.model_selection import
    train_test_split
X = df.drop(columns = ['attack_cat', 'label'])
y = df['label'].values
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.3,
    random_state=11)
feature_names = list(X.columns)
print("X_train shape: ", X_train.shape)
print("y_train shape: ", y_train.shape)
print("X_test shape: ", X_test.shape)
print("y_test shape: ", y_test.shape)
```

Sonuç olarak Xtrain shape: (180371, 42) ytrain shape: (180371,) Xtest shape: (77302, 42) ytest shape: (77302,) bu çıktıyı alıyoruz

a. Deney 1:Decision Tree algoritması ile sınıflandırma

Hazırlanan veri setinin sınıflandırılması için ilk önce aşağıda verilen kod parçası kullanılmıştır.

```
from sklearn.tree import DecisionTreeClassifier
```

Scikit-learn kütüphanesindeki GridSearchCV kullanılarak DecisionTreeClassifier modelinin hiperparametreleri ayarlanmıştır. GridSearchCV, model, hiperparametre ızgarası, çapraz doğrulama sayısı (cv=5) ve skorlama yöntemi (scoring='recall') ile yapılandırılmıştır. En iyi hiperparametreler ve en iyi duyarlılık skoru yazdırılmıştır, bu da en iyi performansı sağlayan ayarları ve bu performansın geri çağırma (recall) skorunu gösterir.

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [2, 4],
    'min_samples_split': [2, 4],
    'min_samples_leaf': [1, 2]}
dt = DecisionTreeClassifier()
grid_search = GridSearchCV(dt, param_grid, cv=5,
    scoring='recall')
grid_search.fit(X_train, y_train)
print("Best parameters:", grid_search.best_params_
    )
print("Best recall score:", grid_search.
    best_score_)
```

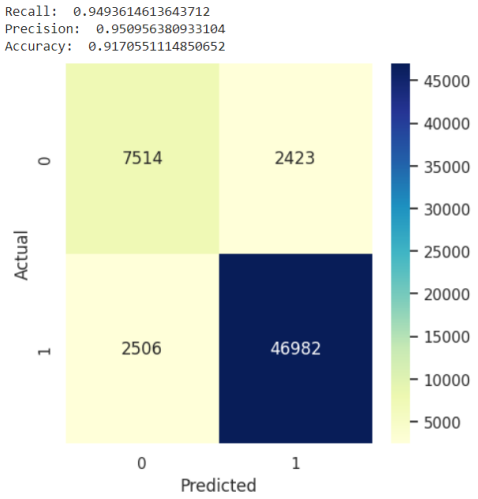
En iyi modeli eğitim veri kümesi üzerinde eğitip test veri kümesinde performansını değerlendiriyoruz.

```
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
clf=grid_search.best_estimator_
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
recall = recall_score(y_test, y_pred)
print("Recall: ", recall)
```

Potansiyel saldırıları tespit etmek için veri filtreleme yapıyoruz

```
X_test = X_test.reset_index(drop=True)
rules= "(sttl <= 61.00 & sinpkt<= 0.00) | (sttl >
     61.00 )"
ind = X_test.query(rules).index
X_test_2 = X_test.loc[ind,:]
y_test_2 = y_test[ind]
print(X_test.shape)
print(X_test_2.shape)
print("filtered data" , (1- np.round(X_test_2.
    shape[0] / X_test.shape[0],2))*100, "%")
```

Aşağıda sınıflandırma modelinin performansını değerlendirilir.Ardından, modeli test veri kümesi üzerinde değerlendirmek için modelevaluation adlı bir fonksiyon kullanır. Bu fonksiyon, modelin tahminlerini gerçek etiketlerle karşılaştırır ve doğruluk, duyarlılık ve kesinlik gibi performans metriklerini hesaplar. Son olarak, tahminlerin gerçek etiketlerle karşılaştırıldığı bir ısı haritası çizer. Burada *Doğruluk* parametresi, 0.94, *Kesinlik* parametresi 0.95, *Duyarlılık* parametresi 0.94 ve olarak hesaplanmıştır.

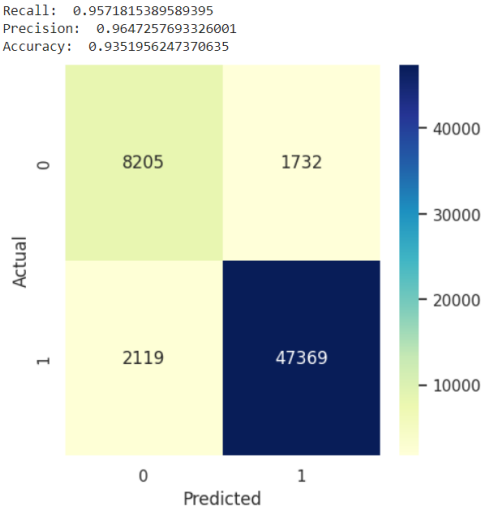


Şekil. 6: Decision Tree Karmaşıklık Matrisi

b. Deney 2: Rastgele Orman Sınıflandırması

Bu kod parçası, RandomForestClassifier modelini random-state=11 ile oluşturur ve eğitir, böylece tekrarlanabilir sonuçlar elde edilir. Model, modelevaluation fonksiyonu kullanılarak değerlendirilir ve doğruluk, kesinlik, geri çağırma gibi performans metrikleri results sözlüğünde 'Random Forest Model' anahtarı altında saklanır.

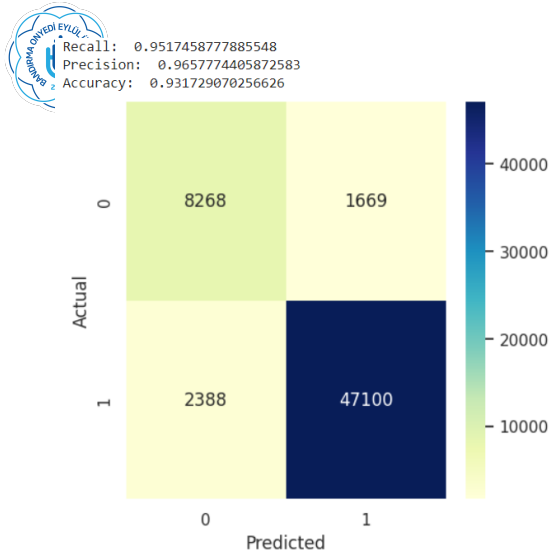
```
from sklearn.ensemble import
    RandomForestClassifier
rf = RandomForestClassifier(random_state=11)
results['Random Forest Model'] = model_evaluation(
    rf)
```



Şekil. 7: Random Forest Karmaşıklık Matrisi

Bu çalışmada XGBoost sınıflandırıcısı (XGBClassifier) kullanılarak bir model oluşturulmuş ve modelevaluation fonksiyonuyla değerlendirilmiştir. Modelin performans metrikleri şunlardır: Hassasiyet (recall) 0.95, kesinlik (precision) 0.96 ve doğruluk (accuracy) 0.93. Bu yüksek değerler, modelin hem genel doğruluğunun hem de pozitif sınıfları tespit etme ve pozitif tahminlerinde isabet oranının yüksek olduğunu göstermektedir.

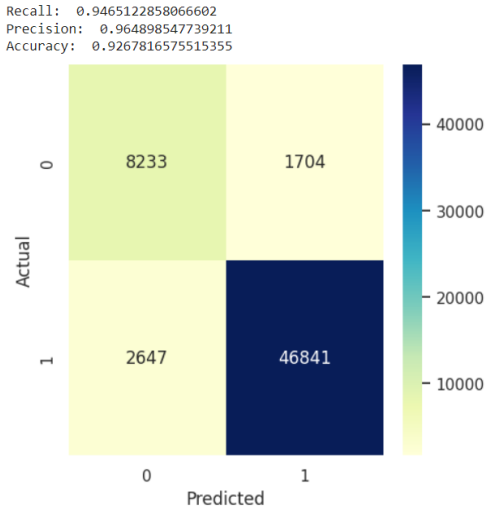
```
from xgboost import XGBClassifier
xgbc = XGBClassifier()
results['XGBoost Classifier'] = model_evaluation(
    xgbc)
```



Şekil. 8: XGBoost Karmaşıklık Matrisi

LightGBM algoritması kullanılarak oluşturulan sınıflandırma modeli, eğitim ve test verileri üzerinde değerlendirildi. Modelin performans metrikleri şunlardır: Hassasiyet (recall) 0.94, kesinlik (precision) 0.96 ve doğruluk (accuracy) 0.92. Bu sonuçlar, LightGBM'in büyük ve karmaşık veri setlerinde hızlı ve yüksek performanslı sonuçlar sağladığını göstermektedir.

```
from lightgbm import LGBMClassifier
lgbc = LGBMClassifier()
results['Light GBM Classifier'] = model_evaluation
    (lgbc)
```



Şekil. 9: LightGBM Karmaşıklık Matrisi

Şekil 10'da kullandığımız Random Forest, LightGBM,XGBoost modellerinin performanslarını görsel olarak karşılaştırmak ve farklı modeller arasında hangisinin daha iyi performans gösterdiğini belirliyoruz.

```
comparision = pd.DataFrame(results)
comparision
```

	Recall	Random Forest Model	XGBoost Classifier	Light GBM Classifier
Recall	0.949119	0.957182	0.951746	0.946512

Şekil. 10: Random Forest, LightGBM,XGBoost Modellerinin Performansları

Wilcoxon sıralama toplamı testi kullanılarak yapılan analizde, üç modelin geri çağırma puanları arasında varyasyonu ölçen bir Z istatistiği ve p-değeri hesaplanmıştır. Null hipotez, üç modelin geri çağırma performanslarında fark olmadığıdır. p-değeri 0.05'ten küçükse, null hipotez reddedilir ve modeller arasında anlamlı fark olduğu kabul edilir. Test sonucunda, p-değeri 0.05'ten büyük bulunmuş ve üç modelin geri çağırma performansları arasında istatistiksel olarak anlamlı bir fark olmadığı sonucuna varılmıştır.

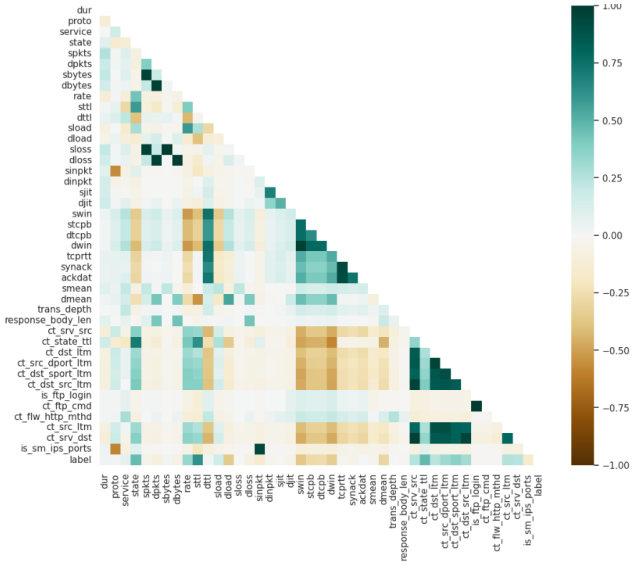
```
from scipy.stats import wilcoxon
z_statistic, p_value = wilcoxon([comparision.iloc
    [0][0], comparision.iloc[0][1], comparision.
    iloc[0[2]] )
print('Z-statistic:', z_statistic)
print('p-value:', p_value)
if p_value < 0.05:
print('The difference in the recall of the three
    models is statistically significant.')
else:print('The difference in the recall of the
    three models is not statistically significant.
    ')
```

```
Z-statistic: 0.0
p-value: 0.25
The difference in the recall of the three models is not statistically significant.
```

Şekil. 11: Üç Model arasında Wilcoxon Hipotez Testi

Korelasyon diyagramları, farklı değişkenlerin birbirleriyle ve siber saldırılarla nasıl ilişkili olduğunu görselleştirmekte yardımcı olabilir. Ayrıca, rastgele orman modelleri, farklı özelliklerin hedef değişkeni (siber saldırılar) tahmin etmedeki önemini belirlemeye yardımcı olabilir. Her bir hücre, iki değişken arasındaki korelasyon katsayısını temsil eder. Değerler genellikle -1 ile 1 arasındadır. Pozitif değerler pozitif bir ilişkiyi, negatif değerler ise negatif bir ilişkiyi gösterir. Değer ne kadar yakınsa, ilişki o kadar güçlüdür.

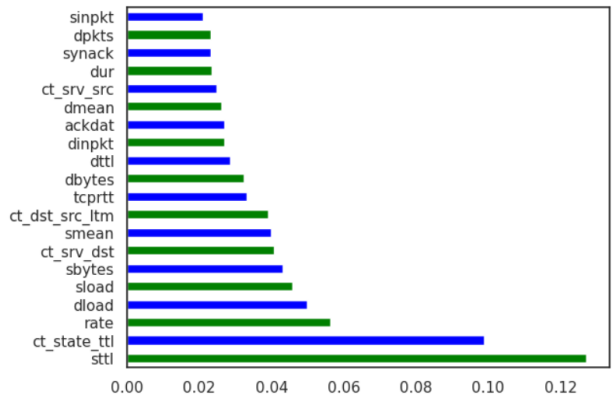
```
numeric_df = df.select_dtypes(include=[np.number])
corr_matrix = numeric_df.corr()
mask = np.triu(np.ones_like(corr_matrix, dtype=
    bool))
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, vmin=-1, vmax=1, cmap='
    BrBG', mask=mask)
plt.show()
```



Şekil. 12: Korelasyon Matrisi

En önemli 20 özelliği görselleştirerek hangi özelliklerin model tarafından en kritik olarak değerlendirildiğini kolayca anlamamıza olanak tanır.

```
feat_importances = pd.Series(rf.
    feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh',
    color=['g','b']*5)
plt.show()
```



Şekil. 13: En Önemli 20 Özellik

En iyi 10 özelliği seçelim ve bu özelliklerin siber saldırı türüyle olan ilişkilerini bulalım ve bu özellikleri kullanarak bir Random Forest modeli oluşturur, eğitir ve test eder.Bu durumda accuracy değeri 0.95 olarak çıkar.

```
top10= feature_imp.Name[:10].tolist()
top10
X = df[top10]
y = df['label'].values
rf_top10=RandomForestClassifier(random_state=11)
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.3,
    random_state=11)
rf_top10.fit(X_train, y_train)
y_pred = rf_top10.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Accuracy: ", acc)
```

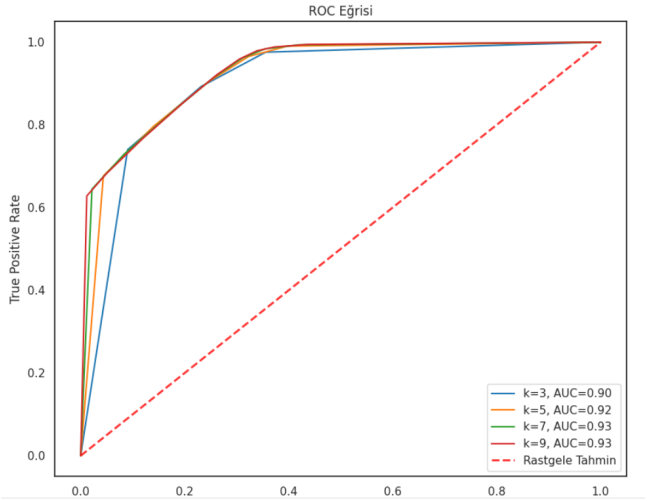
c. Deney 3: KNN sınıflandırması

Kod örneğinde, veri seti eğitim ve test setlerine ayrılır ve belirli bir komşu sayısı kullanılarak (bu örnekte 5) KNN modeli oluşturulur, eğitilir ve test edilir. Modelin performansı doğruluk (accuracy), Kesinlik (precision) ve Hassasiyet (recall) gibi metriklerle değerlendirilir. Bu metrikler, modelin ne kadar doğru tahminler yaptığını ve sınıf dengesizliklerini gösterir.Accuracy 0.85,Precision 0.86,Recall 0.90 değerini almaktadır.

```
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.3,
    random_state=11)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

Her bir K değeri için, KNN modeli oluşturulur ve test seti üzerindeki sınıflandırma olasılıkları elde edilir. Ardından, bu olasılıklar kullanılarak ROC eğrisi çizilir ve alan altındaki eğri (AUC) hesaplanır.

```
k_values = [3, 5, 7, 9]
plt.figure(figsize=(10, 8))
for k in k_values:
knn = KNeighborsClassifier(n_neighbors=k) knn.fit(
    X_train, y_train)
y_scores = knn.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f'k={k}, AUC={roc_auc:.2f
    }')
plt.plot([0, 1], [0, 1], linestyle='--', lw=2,
    color='r', label='Rastgele Tahmin', alpha=.8)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show())
print("Recall:", recall)
```



Şekil. 14: KNN Farklı K Değerleri İçin Roc Eğrisi

d. Deney 4:Veri seti bölümleme oranının değiştirilmesi

BAŞARIM	%20 Test %80 Eğitim	%30 Test %70 Eğitim	%40 Test %60 Eğitim
Decision Tree	0.9089	0.9168	0.9164
Random Forest	0.9442	0.9571	0.9554
KNN	0.8478	0.8505	0.7667

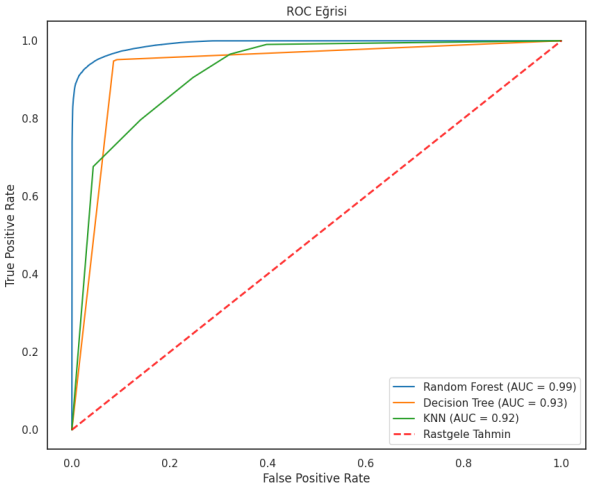
Şekil. 15: Veri Setinin Farklı Oranlarda Bölünmesi

Şekil 15 gösteriyor ki en yüksek başarım oranı Random Forest algoritmasında alınmış ve burada veri setinin bölünme oranı %30 test, %70 eğitim olarak belirlenmiştir.En iyi performans %30 test, %70 eğitim olarak gözlenmiştir.

V. Projenin Performansının Karşılaştırılması

ROC eğrisi grafiğinde, üç farklı makine öğrenme modeli olan Random Forest, Decision Tree ve KNN'nin performansları karşılaştırılmıştır. Random Forest modeli, 0.99 AUC değeri ile en yüksek performansa sahip olup, doğru pozitif oranı yüksek ve yanlış pozitif oranı düşüktür. Decision Tree modeli 0.93 AUC değeri ile ikinci sırada yer alırken, KNN modeli

0.92 AUC değeri ile üçüncü sıradadır. Tüm modeller, rastgele tahmin (kırmızı kesikli çizgi) ile kıyaslandığında çok daha iyi performans sergilemektedir. Özetle, Random Forest en iyi performansı gösterirken, diğer iki model de başarılı sonuçlar elde etmiştir.



Şekil. 16: Modellerin Karşılaştırılması

VI. Sonuçlar ve Çıkarım

Bu çalışmada KNN, Decision Tree, Random Forest makine öğrenme algoritmaları kullanılarak siber saldırı çeşitlerine bağlı olarak 45 değişkenden training veri setinde 82,332 satır ,testing veri setinde ise 175,341 satırdan oluşan bir veri seti kullanılarak siber saldırı tahmini yapılmaktadır. Veri setinde 9 adet etiket bilgisi bulunmaktadır. Bunlar saldırının Fuzzers ,Analysis ,Backdoors, DoS, Exploits Generic ,Reconnaissance ,Shellcode, Worms olmasıdır. Veri yöntemi kısmında, çalışmada kullanılacak modeller ve projede kullanılacak performans ölçütleri araştırılmıştır. Projenin uygulanmasına geçilerek, veri önişleme ,veri kategorilendirme,filtreleme,görselleştirme,hipotez testi gerçekleştirilmiştir. Veri, eğitim ve test verisi olarak belirli bir oranda ayrılmıştır. Bu ayrılma sonucunda Data Test ve Data Train verileri oluşmuştur. Daha sonra veriler etiket ve özellik bilgileri olarak ayrılmış bu ayrılmanın sonucunda da xtrain, xtest, ytrain ve ytest verileri elde edilmiştir. Modellerin uygulamasına geçilerek KNN, Random Forest, Decision Tree modelleri uygulanmış, her model için ROC eğrisi ve Confusion Matris çizdirilmiş; modelin başarım, hassasiyet, kesinlik ve özgüllük değerleri hesaplanmıştır. Modellerin başarım değerleri kıyaslanarak en başarılı ve en başarısız modeller belirlenmiştir. Buna göre en başarılı model 0.9571 ile Random Forest olurken, en başarısız model 0.8505 ile Decision Tree olmuştur. Modellerin hassasiyet değerleri kıyaslanarak hassasiyeti en yüksek ve en düşük modeller belirlenmiştir. Buna göre en yüksek hassasiyet değeri 0.96 ile Random Forest olurken, en düşük hassasiyet değeri 0.86 ile KNN olmuştur. Decision Tree 0.94 oranında hassasiyet değeri almıştır. Ayrıca, farklı saldırı türlerinin tespitindeki başarı oranları ve en iyi performans gösteren modellerin belirlenmesi gösterilmiştir. Bu şekilde, siber güvenlik alanında daha güçlü ve güvenilir çözümler geliştirilmesi hedeflenmiştir.Literatür taraması yapılarak makine öğrenme türleri araştırılmıştır. Makine öğreniminin temeli olarak nitelendirilen 3 algoritma öğrenilmiştir. Veri seti bulunarak incelenmiş, özelliklere göre veri seti ayrılması gerçekleştirilmiştir. Çalışmada kullanılacak modeller olan KNN, Random Forest, Decision Tree incelenmiş, kullanım alanları ve yapıları hakkında bilgi sahibi olunmuştur. Performans ölçütü olarak kullanılacak Confusion Matris ve Roc eğrisi araştırılmıştır. 3 farklı makine öğrenme algoritması kullanılarak siber saldırı tespiti yapılmıştır. Bu değerler için ROC eğrisi ve Confusion matrisi çizdirilmiş, başarımlar ölçülmüştür. Deneysel sonuçlarda komşuluk sayısı ve veri seti bölümlemesi değiştirilerek sonuçlar alınmış eldeki sonuçlarla kıyaslanmıştır. Siber saldırı tespitini en iyi yapan modelin 0.95 başarımla Random Forest olduğu görülmüştür.

References

[1] Andreas C. Müller, Sarah Guido, "Introduction to Machine Learning with Python", O'Reilly Media, 2016.

[2] Christopher M. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006.

[3] Akriti Upadhyay, "Cyber Attack Detection with Random Forest", Kaggle, 2021. Available:https://www.kaggle.com/code/akritiupadhyayks/cyber-attack-detection-with-random-forest-Random-Forest-With-Attack-Category-As-the-Prediction-Labels.

[4] Jake VanderPlas, "Python Data Science Handbook", O'Reilly Media, 2016.

[5] Géron Aurélien, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", O'Reilly Media, 2019.