# Siber Güvenlik İçin Veri Madenciliği
# Vize Ödev Raporu

Hazırlayanlar: 200504016- Betül Yaşar

200504025- Muzaffer Birinci

Bilgilendirme: Bu rapor, malware tespiti amacıyla kullanılan bir veri seti üzerinde makine öğrenimi modeli geliştirme sürecini ayrıntılı olarak açıklamaktadır. Ana odak, ağ trafiğindeki potansiyel zararlı etkinlikleri etkin bir şekilde sınıflandırabilmek için güvenilir bir model oluşturmaktır.

## Malware Traffic Analysis Knowledge Dataset 2019 (MTA-KDD'19) Analizi

**Malware Traffic Analysis Knowledge Dataset 2019 (MTA-KDD'19)**, makine öğrenimi tabanlı zararlı yazılım trafiği analiz algoritmalarını eğitmek ve değerlendirmek için özel olarak hazırlanmış ve güncellenmiş edilmiş bir veri setidir.

### Kullanılan Kütüphaneler ve Amaçları

Projemizde Python dilinde aşağıdaki kütüphaneler kullanılmıştır:

```
In [35]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.preprocessing import MinMaxScaler
         from sklearn.model_selection import train_test_split
         from sklearn.decomposition import PCA
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score, classification_report
         import seaborn as sns
         from sklearn.metrics import confusion_matrix
```

1. **Pandas**: Veri setini yüklemek, temizlemek ve hazırlamak için kullanılan bir kütüphane. Pandas, veri analizi ve manipülasyonu için zengin işlevselliğe sahiptir. Kodumuzda **pd.read_csv** fonksiyonu ile veri setini yüklemekte ve **fillna** metodu ile eksik verileri doldurmaktayız.

2. **Scikit-learn**: Makine öğrenimi modellerini eğitmek, test etmek ve değerlendirmek için kullanılan geniş çaplı bir kütüphane. Model oluşturma, eğitim, tahmin ve performans değerlendirme işlemleri için gerekli araçları sağlar. Özellikle, **MinMaxScaler** sınıfı verileri normalleştirmek için, **RandomForestClassifier** sınıfı modeli oluşturmak ve eğitmek için, **train_test_split** fonksiyonu veri setini bölümlemek için, ve **accuracy_score**, **confusion_matrix** fonksiyonları modelin performansını değerlendirmek için kullanılmıştır.

3. **Matplotlib ve Seaborn**: Veri görselleştirme için kullanılan kütüphaneler. Modelin sonuçlarını ve performansını görsel olarak anlamamızı sağlar. Özellikle,

**matplotlib.pyplot** ve **seaborn** kütüphaneleri, kafa karışıklığı matrisini görselleştirmek için kullanılmıştır.

## Veri Ön İşleme ve Normalizasyon

Eksik verilerin tamamlanması:

```
In [37]: # Eksik verileri (NaN değerlerini) sütun ortalamalarıyla dolduralım
         df.fillna(df.mean(), inplace=True)

         # Eksik veri kalmış mı kontrol edelim
         missing_values = df.isnull().sum()
         missing_values
```

Veri seti, ağ trafiğine ilişkin çeşitli özellikleri içermektedir. İlk adımda, veri setindeki eksik değerleri **df.mean()** kullanarak sütunların ortalamaları ile doldurduk. Bu, veri setinin bütünlüğünü korumak ve eksik veriler nedeniyle oluşabilecek yanlılıkları önlemek için yapılmıştır.

Normalizasuon işlemi:

```
In [41]: #Normalizasyon işlemi
         scaler = MinMaxScaler()

         # 'label' sütunu hariç tüm sütunları ölçeklendirme
         df_scaled = df.copy()
         df_scaled[df_scaled.columns[:-1]] = scaler.fit_transform(df_scaled[df_scaled.columns[:-1]])
```

Normalizasyon aşamasında, **MinMaxScaler** kullanılarak tüm özellikler 0 ile 1 arasında bir ölçeğe dönüştürülmüştür. Bu adım, farklı ölçeklere sahip özelliklerin model tarafından eşit ağırlıkta değerlendirilmesini sağlar. Normalizasyon, modelin eğitimi sırasında daha hızlı ve etkili öğrenme sürecine katkıda bulunur.

## Model Seçimi ve Eğitimi

```
In [42]: #RandomForestClassifier oluşturma ve eğitme
         clf = RandomForestClassifier(n_estimators=100, random_state=42)
         clf.fit(X_train, y_train)
```

Karar Ağaçları ve Rastgele Ormanlar modeli, veri setimiz için seçilmiş olan makine öğrenimi algoritmasıdır. Rastgele Ormanlar, birden fazla karar ağacının sonuçlarını birleştirerek hem modelin genel doğruluğunu artırır hem de aşırı uyuma (overfitting) karşı dayanıklılık sağlar. Bu özellikler, özellikle yüksek boyutlu ve karmaşık özelliklere sahip veri setlerinde avantajlıdır.

Modelin oluşturulması ve eğitimi için Scikit-learn kütüphanesinin **RandomForestClassifier** sınıfı kullanılmıştır. Bu sınıf, modelin parametrelerinin (örneğin, ağaç sayısı) ve eğitim sürecinin kolayca yönetilmesini sağlar. Model, **fit** metodu ile eğitim veri seti üzerinde eğitilmiştir.

## Model Değerlendirmesi

```
# Modelin test veri seti üzerindeki performansını değerlendirme
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     10305

    accuracy                           1.00     10305
   macro avg       1.00      1.00      1.00     10305
weighted avg       1.00      1.00      1.00     10305
```

Modelin test veri seti üzerindeki performansını değerlendirmek için, modelin tahminleri (**predict** metodu kullanılarak) gerçek değerlerle karşılaştırılmış ve **accuracy_score** fonksiyonu

ile modelin doğruluğu hesaplanmıştır. Ayrıca, **confusion_matrix** fonksiyonu kullanılarak bir kafa karışıklığı matrisi oluşturulmuş ve bu matris **seaborn** kütüphanesinin **heatmap** fonksiyonu ile görselleştirilmiştir. Bu görselleştirme, modelin hangi sınıfları doğru tahmin ettiğini ve hangi sınıflarda hata yaptığını net bir şekilde göstermektedir.