Yusuf Pilcioğlu

23OSGP048

YAPAY ZEKA UYGULAMALARI DERSİ ÖDEV

Spam E-Postalarının Sınıflandırılması: Makine Öğrenmesi Yaklaşımı

Giriş

Giriş Bu çalışmada, spam (istenmeyen) e-postaların sayısal öznitelikler kullanılarak tespit edilmesi amaçlanmıştır. UCI Spambase veri seti kullanılarak, e-postaların spam olup olmadığını tahmin edebilecek güvenilir bir makine öğrenmesi modeli geliştirilmiştir. Model performansı, doğruluk, ROC eğrisi, AUC skoru ve 5-Fold çapraz doğrulama metrikleriyle değerlendirilmiştir.

Bu testin temel amacı, e-posta filtreleme sistemlerinde otomatik spam algılamayı sağlayacak bir sınıflandırıcı oluşturmaktır. Bu sayede, kullanıcıların gelen kutularının gereksiz mesajlardan arındırılması ve bilgi güvenliğinin artırılması hedeflenmektedir.

Veri Kümesi Bilgileri

- **Veri Seti:** kaggle.com – spambase_csv.csv
- **Hedef Değişken:** spam (1=spam, 0=spam değil)
- **Bağımsız Değişenler:** E-posta içeriğindeki kelime ve karakter frekanslarına dayalı 57 sayısal öznitelik
- **Eksik Veriler:** En sık tekrar eden değer ile doldurulmuştur
- **Veri Bölünmesi:** %75 eğitim, %25 test veri seti
- **Öznitelik Ölçekleme:** StandardScaler ile uygulanmıştır

Kullanılan Makine Öğrenmesi Modelleri

Bu çalışmada 10 farklı denetimli öğrenme algoritması kullanılmıştır:

1. **CatBoost Classifier :** Kategorik verilerle çok iyi çalışan, hız ve doğruluk açısından güçlü bir boosting algoritması.
2. **LightGBM Classifier :** Büyük veri setlerinde hızlı çalışan, hafıza kullanımı düşük, yüksek performanslı bir gradient boosting yöntemi.
3. **Random Forest Classifier :** Çok sayıda karar ağacının oy çokluğu ile karar verdiği, aşırı öğrenmeye karşı dayanıklı bir topluluk (ensemble) yöntemi.
4. **XGBoost Classifier :** Hataları aşamalı düzelten ve yüksek doğruluk sağlayan, popüler bir gradient boosting algoritması.
5. **Gradient Boosting Classifier :** Zayıf modelleri ardışık olarak geliştirip hataları minimize eden güçlü bir topluluk yöntemidir.
6. **Linear Support Vector Machine (SVM) :** Verileri doğrusal bir sınırla ayırmaya çalışan, büyük boyutlu veri için uygun güçlü sınıflandırıcı.
7. **Logistic Regression :** İkili sınıflandırma problemlerinde olasılık tahmini yapan, basit ve yorumlanabilir doğrusal model.
8. **Decision Tree Classifier :** Veriyi dallara ayırarak kararlar veren, kolay anlaşılır ama aşırı öğrenmeye yatkın model.
9. **K-Nearest Neighbors (KNN) :** Sınıflandırmayı en yakın komşuların etiketlerine göre yapan, basit ancak hesaplama maliyeti yüksek yöntem.
10. **Naive Bayes (GaussianNB) :** Özelliklerin bağımsız olduğunu varsayan, hızlı çalışan olasılıksal sınıflandırıcıdır; sayısal veriler için uygundur.

Bu algoritmalar hem test seti üzerindeki başarılarına göre hem de 5-katlı çapraz doğrulama (5-Fold Cross Validation) skorlarına göre değerlendirilmiştir

Kullanılan Model Karşılaştırmaları

Model	Test Doğruluk (%)	5-Fold CV Skoru (%)
CatBoost	96.09	93.94
LightGBM	96.09	93.17
Random Forest	95.83	92.87
XGBoost	95.66	92.94
Gradient Boosting	94.96	93.00
Linear SVM	92.62	90.20
Logistic Regression	92.27	91.13
Decision Tree	91.49	88.24
K-Nearest Neighbors	90.01	88.42
Naive Bayes	82.71	82.57

En Başarılı Model CatBoost modeli, test doğruluğu %96.09 ve 5-katı çapraz doğrulama skoru %93.94 ile en iyi performansı sergilemiştir. LightGBM ve XGBoost modelleri de benzer şekilde yüksek başarım göstermiştir.

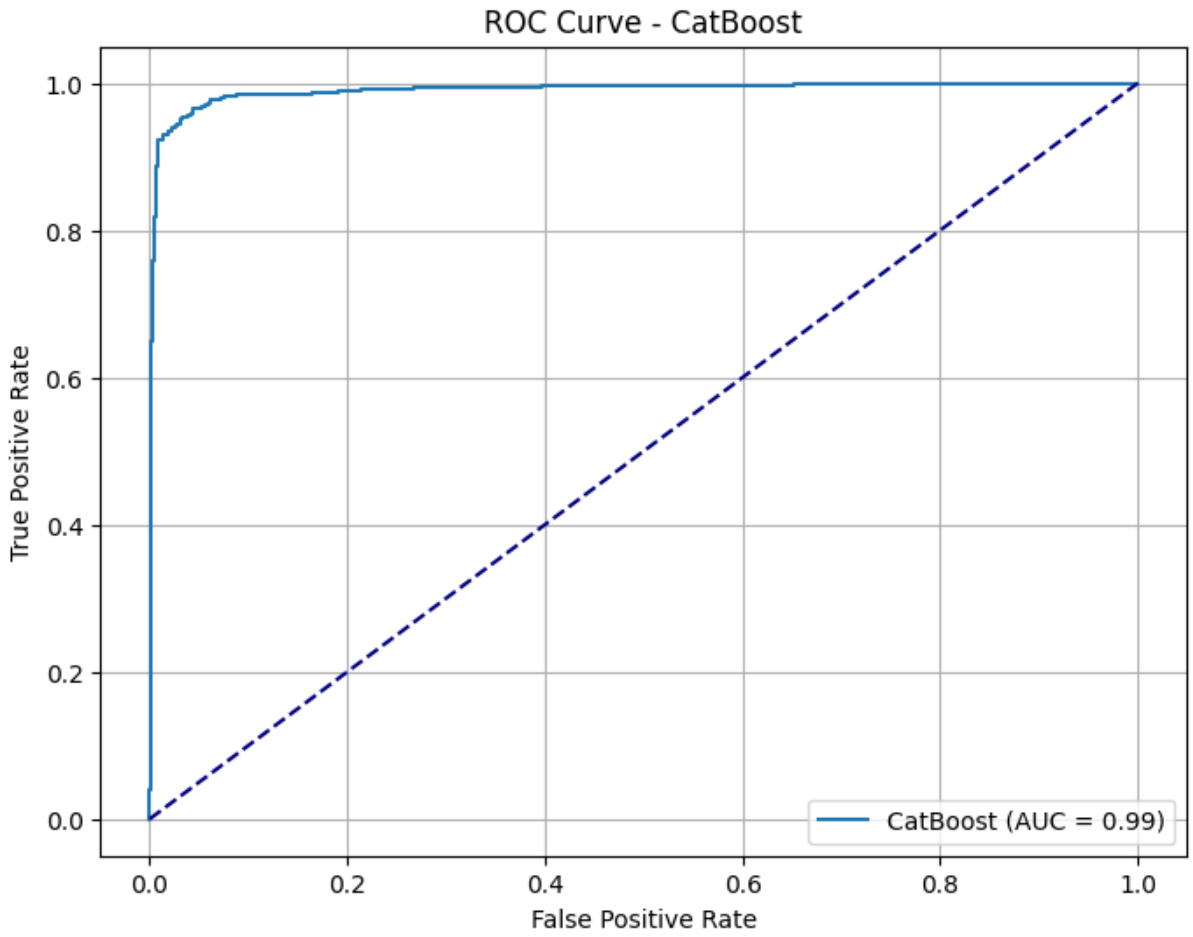
CatBoost Model Performansı

- **Test Seti Doğruluğu :** %96.09
- **Çapraz Doğrulama Skoru (5-Fold CV) :** %93.94
- **ROC AUC Skoru :** 0.99

Model, hem test verisinde hem de genel genelleme başarısında oldukça yüksek bir performans göstermiştir.

CatBoost ROC Eğrisi (Receiver Operating Characteristic)

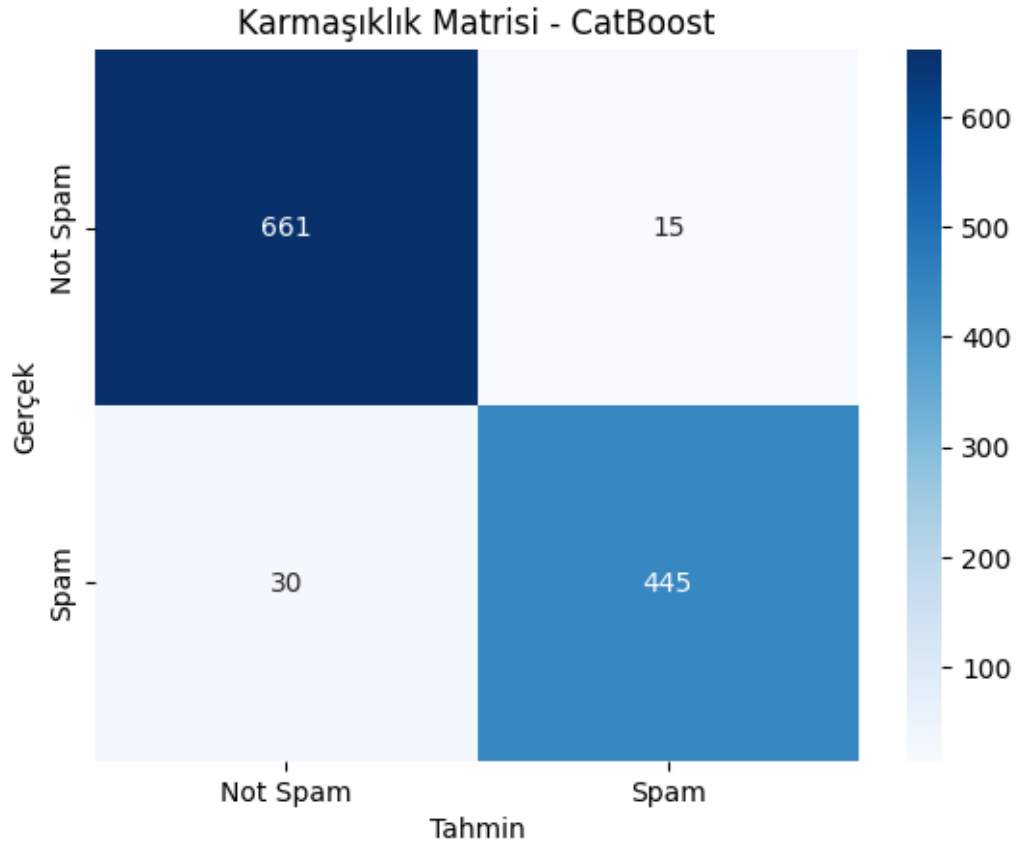
ROC eğrisi (Receiver Operating Characteristic Curve), modelin pozitif sınıfı (spam e-posta) doğru şekilde ayırt etme başarısını görselleştirir. Eğri altında kalan alan (AUC = 0.99) ise modelin hem spam hem de spam olmayan e-postaları ayırt etme gücünün çok yüksek olduğunu gösterir. AUC değerinin 1'e yakın olması, modelin yanlış pozitif ve yanlış negatif oranlarının çok düşük olduğunu ve neredeyse tüm spam e-postaları doğru sınıflandırdığını kanıtlar. Bu da, modelin pratikte spam tespitinde oldukça etkili ve güvenilir olduğunu ifade eder.



CatBoost Karmaşıklık Matrisi

	Tahmin: Not Spam	**Tahmin: Spam**
Gerçek: Not Spam	661	15
Gerçek: Spam	30	445

Model, toplam 475 spam e-postadan 445'ini doğru sınıflandırmış, 30'unu yanlış olarak "Not Spam" diye tahmin etmiştir. Ayrıca, 676 spam olmayan e-postadan 661'ini doğru şekilde "Not Spam" olarak tanımış, 15'ini ise yanlışlıkla "Spam" olarak sınıflandırmıştır.



CatBoost Sınıflandırma Raporu

Model: CatBoost

Test Seti Doğruluğu: 0.9609

5-Fold Çapraz Doğrulama Skoru (Ort.): 0.9394

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	676
1	0.97	0.94	0.95	475
accuracy			0.96	1151
macro avg	0.96	0.96	0.96	1151
weighted avg	0.96	0.96	0.96	1151

Sonuç ve Yorum

CatBoost modeli, spam e-postaları yüksek doğrulukla sınıflandırmada başarılı olmuştur. Özellikle spam olan e-postaları doğru tespit etme başarısı, kullanıcıların gereksiz ve zararlı mesajlardan korunmasında kritik bir rol oynayabilir. Modelin yüksek test doğruluğu ve 5-Fold çapraz doğrulama skorları, genellenebilir ve güvenilir bir performans sergilediğini göstermektedir. Ayrıca, sınıflandırma raporundaki yüksek precision ve recall değerleri, modelin hem yanlış pozitif hem de yanlış negatif oranlarının düşük olduğunu kanıtlamaktadır.

Bu güçlü performans, spam filtreleme sistemlerinde modelin etkin şekilde kullanılabileceğini ve e-posta güvenliğini artıracağını göstermektedir.

Python Kodları

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
roc_curve, roc_auc_score

from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.svm import LinearSVC

from sklearn.neighbors import KNeighborsClassifier

from xgboost import XGBClassifier

from lightgbm import LGBMClassifier

from catboost import CatBoostClassifier


# Veri setini oku

df = pd.read_csv("/content/spambase_csv.csv")



# Özellik ve hedef

X = df.iloc[:, :-1]

y = df.iloc[:, -1]
```

```
# Eğitim/test ayrımı
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Özellik ölçekleme
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Modeller
models = {
    "Logistic Regression": LogisticRegression(),
    "Naive Bayes": GaussianNB(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "Linear SVM": LinearSVC(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss'),
    "LightGBM": LGBMClassifier(),
    "CatBoost": CatBoostClassifier(verbose=0)
}

# Sonuçları saklamak için
results = []
```

```
# Tüm modeller için döngü
for name, model in models.items():
    print(f"\n Model: {name}")

    if name == "Naive Bayes":
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        cv_scores = cross_val_score(model, X, y, cv=5)
    else:
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
        cv_scores = cross_val_score(model, scaler.transform(X), y, cv=5)

    # Test doğruluğu
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Test Seti Doğruluğu: {accuracy:.4f}")
    print(f"5-Fold Çapraz Doğrulama Skoru (Ort.): {cv_scores.mean():.4f}")

    # Classification Report
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Not Spam",
"Spam"], yticklabels=["Not Spam", "Spam"])
    plt.title(f"Karmaşıklık Matrisi - {name}")
    plt.xlabel("Tahmin")
```

```
    plt.ylabel("Gerçek")
    plt.show()

    # ROC
    if hasattr(model, "predict_proba"):
        y_probs = model.predict_proba(X_test_scaled if name != "Naive Bayes" else X_test)[:, 1]
    elif hasattr(model, "decision_function"):
        y_probs = model.decision_function(X_test_scaled)
        y_probs = (y_probs - y_probs.min()) / (y_probs.max() - y_probs.min())
    else:
        y_probs = None

    if y_probs is not None:
        fpr, tpr, _ = roc_curve(y_test, y_probs)
        auc_score = roc_auc_score(y_test, y_probs)
        plt.figure(figsize=(8, 6))
        plt.plot(fpr, tpr, label=f"{name} (AUC = {auc_score:.2f})")
        plt.plot([0, 1], [0, 1], linestyle='--', color='navy')
        plt.xlabel("False Positive Rate")
        plt.ylabel("True Positive Rate")
        plt.title(f"ROC Curve - {name}")
        plt.legend()
        plt.grid(True)
        plt.show()

    # Sonuçlara ekle
    results.append({
```

```
        "Model": name,
        "Test Doğruluğu": round(accuracy, 4),
        "CV Ort. Doğruluk (5-Fold)": round(cv_scores.mean(), 4)
    })

# Tüm sonuçları göster
summary_df = pd.DataFrame(results)
print("\n MODELLERİN KARŞILAŞTIRMA TABLOSU:\n")
print(summary_df.sort_values(by="Test Doğruluğu",
ascending=False).to_string(index=False))
```