

MAKİNE ÖĞRENMESİ FINAL PROJESİ



MUSTAFA EMRE DİKİCİ
BİLGİSAYAR MÜHENDİSLİĞİ
21040101009

Ödev Linki: https://github.com/dikiciemre/machine_learning_diabet_2

GİRİŞ

Bu proje, makine öğrenmesi yöntemlerini kullanarak diyabet teşhisinde kullanılabilecek modellerin geliştirilmesini ve performanslarının değerlendirilmesini amaçlamaktadır. Çalışmada, diyabet teşhisine yönelik veriler kullanılarak çeşitli makine öğrenmesi algoritmaları ile modeller eğitilmiş ve bu modellerin performansları karşılaştırılmıştır. Proje kapsamında kullanılan veri seti, diyabet teşhisinin gerekli olan çeşitli tıbbi ölçümleri içermektedir. Bu ölçümler arasında glikoz seviyesi, kan basıncı, cilt kalınlığı, insülin seviyesi ve vücut kitle indeksi gibi parametreler bulunmaktadır.

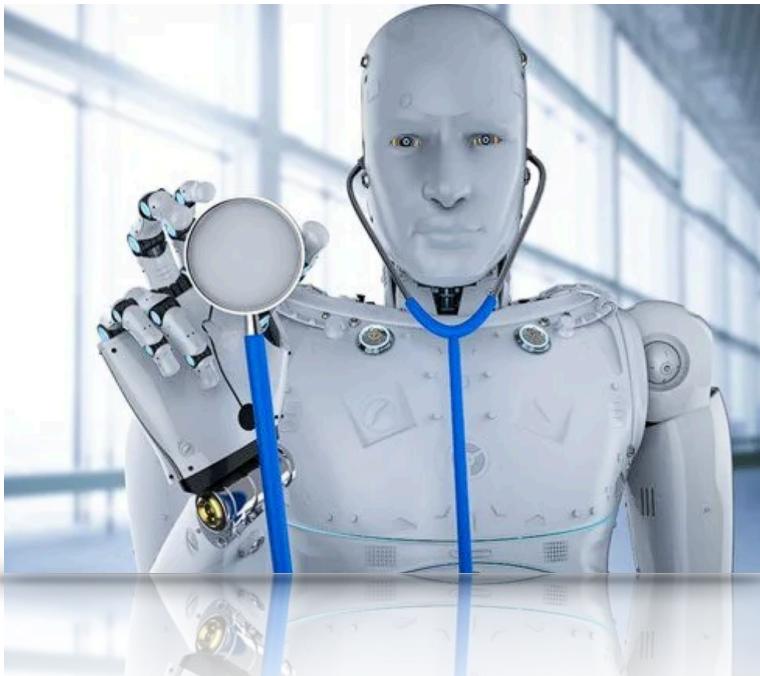
Projenin Amacı

Bu çalışmanın temel amacı, diyabet teşhisini için etkili bir model geliştirmek ve farklı makine öğrenmesi algoritmalarının performansını değerlendirmektir. Projede, Gaussian Naive Bayes, K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP) ve Support Vector Machines (SVM) gibi çeşitli algoritmalar kullanılmıştır. Her bir modelin performansı, doğruluk, kesinlik, duyarlılık, özgürlük, F1 skoru ve ROC AUC gibi metriklerle değerlendirilmiştir.

Kullanılan Yöntemler ve Araçlar

Projede kullanılan yöntemler ve araçlar şunlardır:

- Veri İşleme ve Manipülasyon:** Pandas ve NumPy kütüphaneleri kullanılarak veri seti yüklenmiş, işlenmiş ve analiz için hazırlanmıştır.
- Model Eğitimi ve Değerlendirme:** Scikit-learn kütüphanesi kullanılarak çeşitli makine öğrenmesi modelleri eğitilmiş ve değerlendirilmiştir.
- Görselleştirme:** Matplotlib ve Seaborn kütüphaneleri kullanılarak performans metrikleri ve modellerin çıktıları görselleştirilmiştir.



Veri Seti ve Özellikleri

Veri seti, diyabet teşhisini konulmuş veya konulmamış bireylere ait çeşitli tıbbi ölçümleri içermektedir. Bu veri seti, diyabet teşhisini konmuş (1) veya konmamış (0) bireylerin tıbbi geçmişini ve çeşitli ölçümlerini içeren 'diabetes.csv' dosyasından alınmıştır. Özellikler arasında glikoz seviyesi, kan basıncı, cilt kalınlığı, insülin seviyesi ve vücut kitle indeksi gibi tıbbi parametreler bulunmaktadır. Hedef değişken ise diyabet teşhisini olup olmadığı bilgisini taşımaktadır.

Model Eğitimi ve Test Edilmesi

Veri seti, model eğitimi için %70 eğitim ve %30 test olarak ikiye ayrılmıştır. Model eğitiminde kullanılan yöntemler ve adımlar şu şekildedir:

1. **Gaussian Naive Bayes Modeli:** Bu model, veri setindeki özellikler arasındaki bağımsızlık varsayımasına dayanmaktadır. Model, eğitim veri seti kullanılarak eğitilmiş ve test veri seti üzerinde tahminler yapılmıştır.
2. **K-Nearest Neighbors (KNN) Modeli:** Bu modelde, farklı k değerleri için doğruluk skoru hesaplanmış ve en iyi k değeri ile nihai model eğitilmiştir.
3. **Multi-Layer Perceptron (MLP) Modeli:** Yapay sinir ağları temelli bu model, belirli sayıda iterasyon boyunca eğitilmiş ve test veri seti üzerinde tahminler yapılmıştır.
4. **Support Vector Machines (SVM) Modeli:** Sınıflar arasında en iyi ayırma hiper düzlemini bulmaya çalışan bu model, eğitim veri seti kullanılarak eğitilmiş ve test veri seti üzerinde tahminler yapılmıştır.

Performans Değerlendirme ve Sonuçlar

Her bir modelin performansı, doğruluk, kesinlik, duyarlılık, özgüllük, F1 skoru ve ROC AUC gibi metriklerle değerlendirilmiştir. Bu metrikler, modellerin gerçek veriler üzerindeki tahmin yeteneklerini ölçmek için kullanılmıştır. Ayrıca, konfüzyon matrisleri ve ROC eğrileri görselleştirilerek modellerin performansları karşılaştırılmıştır.

Bu proje, diyabet teşhisinde makine öğrenmesi algoritmalarının etkinliğini göstermeyi amaçlamakta ve farklı modellerin performanslarını karşılaştırarak en uygun modeli belirlemeye çalışmaktadır. Elde edilen sonuçlar, diyabet teşhisi konusundaki çalışmalar için önemli bir referans noktası oluşturabilir.



1. MADDE KODU

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Veri setini yükleyin
data = pd.read_csv('diabetes.csv')

# Özellikler ve hedef değişkeni ayırin
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Kategorik ve sayısal sütunları belirleyin
categorical_cols = X.select_dtypes(include=['object']).columns
numerical_cols = X.select_dtypes(include=['number']).columns

# Ön işleme pipeline'sını oluşturun
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)])

# Model pipeline'sını oluşturun
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier', RandomForestClassifier())])

# Hiperparametre optimizasyonu için Grid Search
param_grid = {
    'classifier__n_estimators': [50, 100, 200],
    'classifier__max_depth': [None, 10, 20, 30],
    'classifier__min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(model, param_grid, cv=5, n_jobs=-1, scoring='accuracy')
grid_search.fit(X, y)

# En iyi modeli seçin
best_model = grid_search.best_estimator_

# Model performansını değerlendirin
cv_results = cross_val_score(best_model, X, y, cv=5, scoring='accuracy')
cv_mean_accuracy = np.mean(cv_results)
print(f'Cross-validated accuracy: {cv_mean_accuracy:.2f}')

# Tahminleri yapın
y_pred = best_model.predict(X)
classification_rep = classification_report(y, y_pred, output_dict=True)

# Siniflandırma raporunu DataFrame'e dönüştürün
classification_df = pd.DataFrame(classification_rep).transpose()

# Çapraz doğrulama sonuçlarını DataFrame'e ekleyin
cv_results_df = pd.DataFrame(cv_results, columns=['Cross-Validation Accuracy'])
cv_results_df.loc['mean'] = cv_mean_accuracy

# Tahminleri ve gerçek değerleri birleştirin
results_df = pd.DataFrame({'Gerçek Değerler': y, 'Tahminler': y_pred})

# Tüm sonuçları bir araya getirin
combined_df = pd.concat([results_df, cv_results_df, classification_df], axis=1)

# Sonuçları bir CSV dosyasına kaydedin
combined_df.to_csv('tum_sonucular.csv', index=False)

print("Tüm sonuçlar tum_sonucular.csv dosyasına kaydedildi.")
```

1. MADDE KODU ÇIKTILARI

tum_sonuclar

Gerçek Değerler	Tahminler	Cross-Validation Accuracy	precision	recall	f1-score	support
1.0	1.0	0.7727272727272727	0.9881422924901185	1.0	0.9940357852882703	500.0
0.0	0.0	0.7142857142857143	1.0	0.9776119402985075	0.9886792452830189	268.0
1.0	1.0	0.7662337662337663	0.9921875	0.9921875	0.9921875	0.9921875
0.0	0.0	0.8366013071895425	0.9940711462450593	0.9888059701492538	0.9913575152856446	768.0
1.0	1.0	0.7516339869281046	0.9922801383399209	0.9921875	0.9921665760156045	768.0
0.0	0.0					
1.0	1.0					
0.0	0.0					
1.0	1.0					
1.0	1.0					
0.0	0.0					
1.0	1.0					
1.0	1.0					
1.0	1.0					
0.0	0.0					
1.0	1.0					
0.0	0.0					
0.0	0.0					
1.0	1.0					

2. MADDE KODU

```
##  
import numpy as np  
import pandas as pd  
from sklearn.model_selection import train_test_split  
from sklearn.naive_bayes import GaussianNB  
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, roc_curve, roc_auc_score  
import matplotlib.pyplot as plt  
  
# Örnek bir veri seti yükleyelim. Veri setinizin adını ve yolunu buraya ekleyin.  
df = pd.read_csv('diabetes.csv')  
  
# Özellikleri ve hedef değişkeni belirleyin  
X = df.drop('Outcome', axis=1)  
y = df['Outcome']  
  
# Veriyi eğitim ve test setlerine bölelim (%70 eğitim, %30 test)  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)  
  
# Gaussian Naive Bayes modelini oluştur ve eğit  
nb_model = GaussianNB()  
nb_model.fit(X_train, y_train)  
  
# Test veri seti üzerinde tahmin yap  
y_pred = nb_model.predict(X_test)  
  
# Konfüzyon matrisi  
cm = confusion_matrix(y_test, y_pred)  
  
# Performans metrikleri  
accuracy = accuracy_score(y_test, y_pred)  
precision = precision_score(y_test, y_pred, average='binary') # İkili sınıflandırma için 'binary'  
recall = recall_score(y_test, y_pred, average='binary') # İkili sınıflandırma için 'binary'  
f1 = f1_score(y_test, y_pred, average='binary') # İkili sınıflandırma için 'binary'  
specificity = cm[0, 0] / (cm[0, 0] + cm[0, 1])  
roc_auc = roc_auc_score(y_test, y_pred)  
  
# Sonuçları bir DataFrame olarak düzenleyin  
results = pd.DataFrame({  
    'Metric': ['Accuracy', 'Precision', 'Recall (Sensitivity)', 'Specificity', 'F1 Score', 'ROC AUC'],  
    'Value': [accuracy, precision, recall, specificity, f1, roc_auc]  
})  
  
# Sonuçları bir CSV dosyasına kaydedin  
results.to_csv('classification_results.csv', index=False)  
  
# ROC eğrisi  
fpr, tpr, thresholds = roc_curve(y_test, nb_model.predict_proba(X_test)[:, 1])  
  
# ROC Eğrisi çizimi ve kaydedilmesi  
plt.figure()  
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)  
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')  
plt.xlim([0.0, 1.0])  
plt.ylim([0.0, 1.05])  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('Receiver Operating Characteristic')  
plt.legend(loc="lower right")  
plt.savefig('roc_curve.png')  
plt.show()  
  
# Konfüzyon matrisini bir CSV dosyasına kaydedin  
cm_df = pd.DataFrame(cm, index=['Actual Negative', 'Actual Positive'], columns=['Predicted Negative', 'Predicted Positive'])  
cm_df.to_csv('confusion_matrix.csv')
```

2. MADDE KODU AÇIKLAMASI

1. Gerekli Kütüphanelerin İçe Aktarılması

- **numpy** ve **pandas** veri işleme için,
- **sklearn** model oluşturma, değerlendirme ve veri bölme için,
- **matplotlib** grafik çizimi için kullanılır.

2. Veri Setinin Yüklenmesi

Veri seti **diabetes.csv** dosyasından yüklenir ve bir DataFrame (**df**) olarak okunur.

3. Özelliklerin ve Hedef Değişkenin Belirlenmesi

- **X** bağımsız değişkenleri (özellikler) içerir.
- **y** bağımlı değişkeni (hedef) içerir, bu örnekte 'Outcome' kolonu.

4. Veri Setinin Eğitim ve Test Setlerine Bölünmesi

Veri seti, %70 eğitim ve %30 test olacak şekilde bölünür. **random_state=42** kullanılarak bölünmenin tekrarlanabilir olması sağlanır.

5. Gaussian Naive Bayes Modelinin Oluşturulması ve Eğitilmesi

- Gaussian Naive Bayes modeli oluşturulur (**nb_model**).
- Model, eğitim veri seti (**X_train**, **y_train**) kullanılarak eğitilir.

6. Test Veri Seti Üzerinde Tahmin Yapılması

Eğitilen model, test veri seti (**X_test**) üzerinde tahminler yapar ve sonuçlar **y_pred** değişkenine kaydedilir.

7. Konfüzyon Matrisinin Hesaplanması

Konfüzyon matrisi, gerçek ve tahmin edilen değerler kullanılarak hesaplanır. Bu matris, doğru ve yanlış sınıflandırmaların sayısını gösterir.

8. Performans Metriklerinin Hesaplanması

- **Doğruluk (Accuracy)**: Doğru tahminlerin oranı.
- **Kesinlik (Precision)**: Doğru pozitif tahminlerin, toplam pozitif tahminlere oranı.
- **Duyarlılık (Recall)**: Doğru pozitif tahminlerin, toplam gerçek pozitiflere oranı.
- **F1 Skoru**: Kesinlik ve duyarlılığın harmonik ortalaması.
- **Özgüllük (Specificity)**: Doğru negatif tahminlerin, toplam gerçek negatiflere oranı.
- **ROC AUC Skoru**: ROC eğrisi altındaki alan.

9. Sonuçların CSV Dosyasına Kaydedilmesi

Bu metrikler bir DataFrame olarak düzenlenir ve **classification_results.csv** dosyasına kaydedilir.

10. ROC Eğrisinin Çizilmesi ve Kaydedilmesi

ROC eğrisi, yanlış pozitif oran (FPR) ve doğru pozitif oran (TPR) kullanılarak çizilir ve grafik olarak **roc_curve.png** dosyasına kaydedilir.

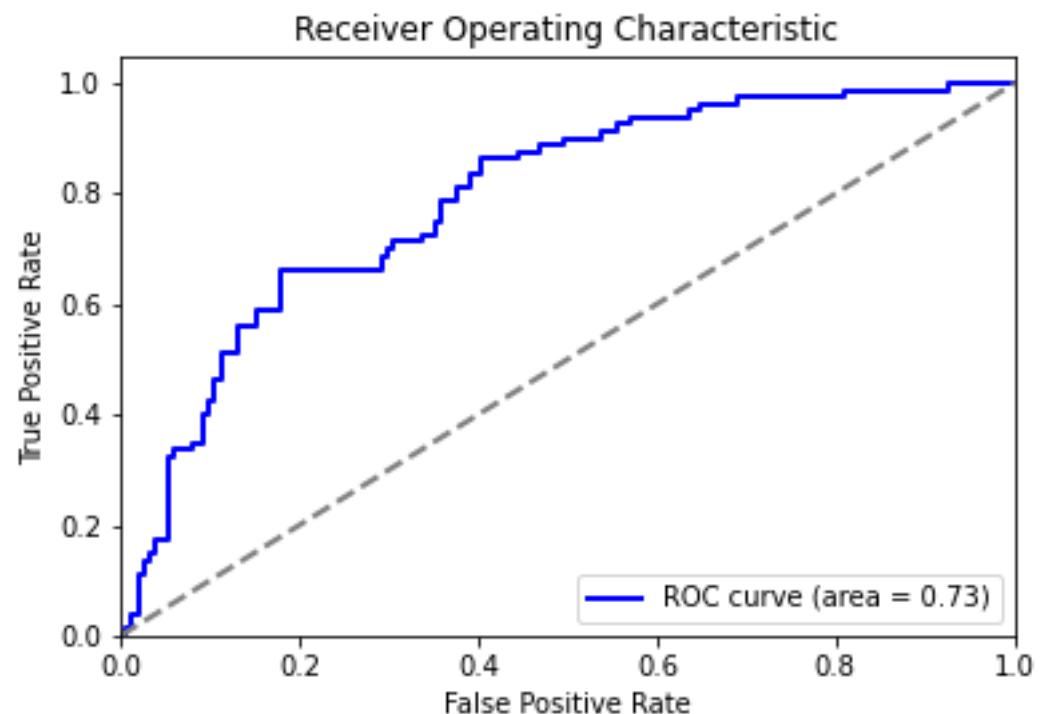
11. Konfüzyon Matrisinin CSV Dosyasına Kaydedilmesi

Konfüzyon matrisi bir DataFrame olarak düzenlenir ve **confusion_matrix.csv** dosyasına kaydedilir.

2. MADDE KODU ÇIKTILARI

classification_results

Metric	Value
Accuracy	0.7445887445887446
Precision	0.6235294117647059
Recall (Sensitivity)	0.6625
Specificity	0.7880794701986755
F1 Score	0.6424242424242423
ROC AUC	0.7252897350993377



confusion_matrix

	Predicted Negative	Predicted Positive
Actual Negative	119	32
Actual Positive	27	53

3. MADDE KODU

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu May 23 22:46:08 2024

@author: emredikici
"""

import ...

# Veri kümесини yükle
file_path = 'diabetes.csv'
data = pd.read_csv(file_path)

# Veri kümесини eğit (70%) ve test (30%) olarak ayır
train_data, test_data = train_test_split(data, test_size=0.3, random_state=42)

# Özellikleri ve etiketlerini ayı
X_train = train_data.drop('Outcome', axis=1)
y_train = train_data['Outcome']
X_test = test_data.drop('Outcome', axis=1)
y_test = test_data['Outcome']

# En iyi k değerini bul
best_k = 1
best_score = 0
scores = []

for k in range(1, 21):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    score = knn.score(X_test, y_test)
    scores.append(score)
    if score > best_score:
        best_score = score
        best_k = k

# En iyi k ile nihai modeli eğit
best_knn = KNeighborsClassifier(n_neighbors=best_k)
best_knn.fit(X_train, y_train)
y_pred = best_knn.predict(X_test)
y_prob = best_knn.predict_proba(X_test)[:, 1]

# Metriği hesapla
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred, output_dict=True)
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

# Sonuçları bir CSV dosyasına kaydet
class_report_df = pd.DataFrame(class_report).transpose()
class_report_df['ROC AUC'] = roc_auc
class_report_df.to_csv('classification_report.csv', index=True)

# Confusion Matrix grafiği
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=[0, 1], yticklabels=[0, 1])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.savefig('confusion_matrix.png')
plt.show()

# ROC Curve grafiği
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.savefig('roc_curve.png')
plt.show()

# En iyi k ve performans metriklerini döndür
best_k, conf_matrix, class_report, roc_auc
```

3. MADDE KODU AÇIKLAMASI

1. Kütüphanelerin İçe Aktarılması

- **pandas:** Veri işleme ve veri kümesi manipülasyonu için kullanılır.
- **sklearn:** Makine öğrenmesi algoritmaları, veri kümesi bölme, metrik hesaplama gibi işlemler için kullanılır.
- **numpy:** Sayısal hesaplamalar ve dizi işlemleri için kullanılır.
- **matplotlib ve seaborn:** Grafik çizimi ve veri görselleştirme için kullanılır.

2. Veri Kümesinin Yüklenmesi

Veri kümesi **diabetes.csv** dosyasından pandas DataFrame olarak yüklenir.

3. Eğitim ve Test Setlerine Bölünmesi

Veri kümesi, %70 eğitim ve %30 test olacak şekilde ikiye ayrılır. Bu, modelin doğruluğunu test etmek için kullanılır. **random_state=42** kullanılarak bölünmenin tekrarlanabilir olması sağlanır.

4. Özelliklerin ve Etiketlerin Ayırılması

Eğitim ve test veri setlerinden bağımsız değişkenler (özellikler) ve bağımlı değişken (hedef) ayrılır. Özellikler bağımsız değişkenler olarak kullanılırken, hedef değişken modelin tahmin etmeye çalıştığı sınıfları içerir.

5. En İyi k Değerinin Belirlenmesi

Bir döngü kullanılarak, 1'den 20'ye kadar farklı **k** değerleri için KNN modeli eğitilir ve test seti üzerinde doğruluk skorları hesaplanır. En yüksek doğruluk skoruna sahip olan **k** değeri en iyi **k** olarak seçilir.

6. Nihai Modelin Eğitilmesi ve Tahmin Yapılması

En iyi **k** değeri ile KNN modeli tekrar eğitilir ve test veri seti üzerinde tahminler yapılır. Modelin her sınıfı ait tahmin olasılıkları da hesaplanır.

7. Performans Metriklerinin Hesaplanması

- **Konfüzyon Matrisi:** Gerçek sınıflar ve modelin tahmin ettiği sınıflar arasındaki ilişkiyi gösterir.
- **Sınıflandırma Raporu:** Doğruluk, kesinlik, duyarlılık ve F1 skoru gibi performans metriklerini içerir.
- **ROC Eğrisi ve AUC:** Modelin ayırt etme yeteneğini gösterir. ROC eğrisi, farklı eşik değerleri için doğru pozitif oranı ve yanlış pozitif oranı grafiğini çizer. AUC (ROC eğrisi altındaki alan), modelin genel performansını değerlendirir.

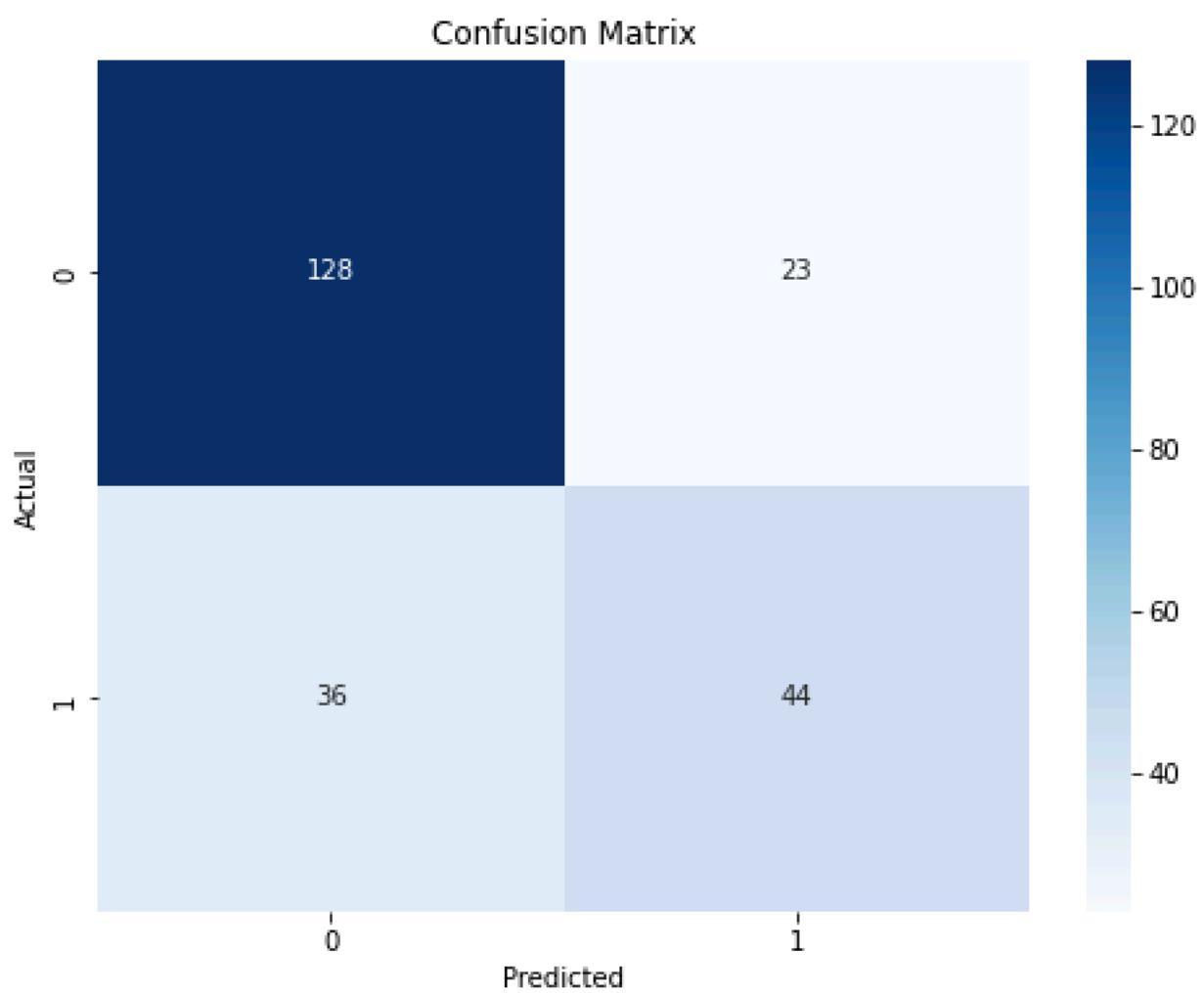
8. Sonuçların Kaydedilmesi

- **Sınıflandırma Raporu:** Performans metrikleri bir CSV dosyasına kaydedilir.
- **Konfüzyon Matrisi:** Grafik olarak çizilir ve bir görüntü dosyasına kaydedilir.
- **ROC Eğrisi:** Grafik olarak çizilir ve bir görüntü dosyasına kaydedilir.

9. En İyi k ve Performans Metriklerinin Döndürülmesi

Kodun sonunda, en iyi **k** değeri, konfüzyon matrisi, sınıflandırma raporu ve ROC AUC skoru döndürülür.

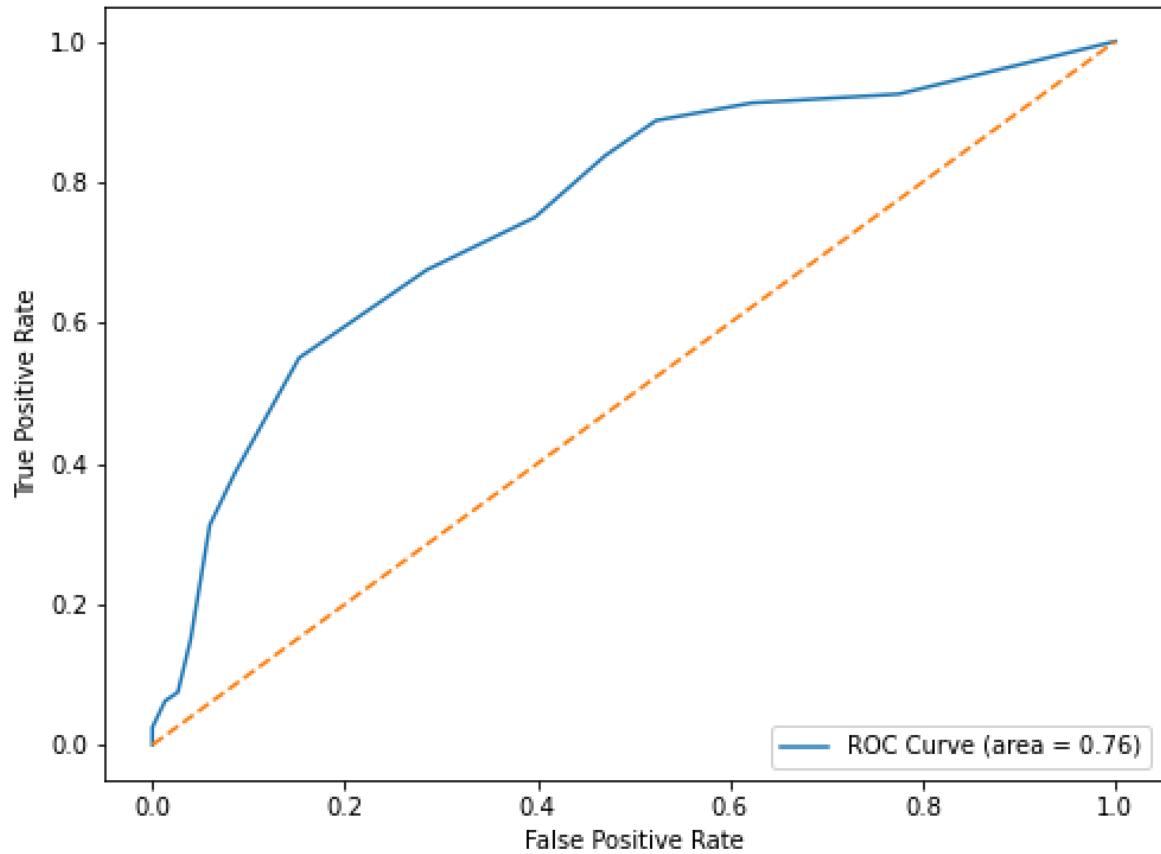
3. MADDE KODU ÇIKTILARI



classification_report

	precision	recall	f1-score	support	ROC AUC
0	0.7804878048780488	0.847682119205298	0.8126984126984126	151.0	0.758816225165563
1	0.6567164179104478	0.55	0.5986394557823129	80.0	0.758816225165563
accuracy	0.7445887445887446	0.7445887445887446	0.7445887445887446	0.7445887445887446	0.758816225165563
macro avg	0.7186021113942482	0.6988410596026491	0.7056689342403628	231.0	0.758816225165563
weighted avg	0.7376232552788796	0.7445887445887446	0.7385654406062568	231.0	0.758816225165563

Receiver Operating Characteristic (ROC) Curve



4. MADDE KODU

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu May 23 22:49:06 2024

@author: emredikici
"""

import ...

# Veri kümnesini yükle
file_path = 'diabetes.csv'
data = pd.read_csv(file_path)

# Veri kümnesini eğitim (%70) ve test (%30) olarak ayıracak şekilde train_data, test_data = train_test_split(data, test_size=0.3, random_state=42)

# Özellikleri ve etiketleri ayrı
X_train = train_data.drop('Outcome', axis=1)
y_train = train_data['Outcome']
X_test = test_data.drop('Outcome', axis=1)
y_test = test_data['Outcome']

# Multi-Layer Perceptron (MLP) sınıflandırıcısını eğit
mlp = MLPClassifier(random_state=42, max_iter=300)
mlp.fit(X_train, y_train)
y_pred_mlp = mlp.predict(X_test)
y_prob_mlp = mlp.predict_proba(X_test)[:, 1]

# Support Vector Machines (SVM) sınıflandırıcısını eğit
svm = SVC(probability=True, random_state=42)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
y_prob_svm = svm.predict_proba(X_test)[:, 1]

# MLP metrikleri hesapla
conf_matrix_mlp = confusion_matrix(y_test, y_pred_mlp)
class_report_mlp = classification_report(y_test, y_pred_mlp, output_dict=True)
fpr_mlp, tpr_mlp, _ = roc_curve(y_test, y_prob_mlp)
roc_auc_mlp = auc(fpr_mlp, tpr_mlp)

# SVM metrikleri hesapla
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
class_report_svm = classification_report(y_test, y_pred_svm, output_dict=True)
fpr_svm, tpr_svm, _ = roc_curve(y_test, y_prob_svm)
roc_auc_svm = auc(fpr_svm, tpr_svm)

# Sonuçları bir CSV dosyasına kaydet
results_mlp = {
    'Confusion Matrix': [conf_matrix_mlp],
    'Classification Report': [class_report_mlp],
    'ROC AUC': [roc_auc_mlp]
}

results_svm = {
    'Confusion Matrix': [conf_matrix_svm],
    'Classification Report': [class_report_svm],
    'ROC AUC': [roc_auc_svm]
}

results_mlp_df = pd.DataFrame(results_mlp)
results_svm_df = pd.DataFrame(results_svm)

results_mlp_df.to_csv('classification_report_mlp.csv', index=False)
results_svm_df.to_csv('classification_report_svm.csv', index=False)

# Confusion Matrix grafikleri
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.heatmap(conf_matrix_mlp, annot=True, fmt="d", cmap="Blues", xticklabels=[0, 1], yticklabels=[0, 1])
plt.title('Confusion Matrix - MLP')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.subplot(1, 2, 2)
sns.heatmap(conf_matrix_svm, annot=True, fmt="d", cmap="Blues", xticklabels=[0, 1], yticklabels=[0, 1])
plt.title('Confusion Matrix - SVM')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.tight_layout()
plt.savefig('confusion_matrices.png')
plt.show()

# ROC Curve grafikleri
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr_mlp, tpr_mlp, label=f'ROC Curve (area = {roc_auc_mlp:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - MLP')
plt.legend(loc='lower right')

plt.subplot(1, 2, 2)
plt.plot(fpr_svm, tpr_svm, label=f'ROC Curve (area = {roc_auc_svm:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - SVM')
plt.legend(loc='lower right')

plt.tight_layout()
plt.savefig('roc_curves.png')
plt.show()

# Sonuçlar ve grafikleri döndür
conf_matrix_mlp, class_report_mlp, roc_auc_mlp, conf_matrix_svm, class_report_svm, roc_auc_svm
```

4. MADDE KODU AÇIKLAMASI

1. Gerekli Kütüphanelerin İçe Aktarılması

Kodun başında, veri işleme, model oluşturma ve değerlendirme, grafik çizme için gerekli kütüphaneler içe aktarılır:

- **pandas**: Veri okuma ve işleme için.
- **sklearn**: Model oluşturma, eğitim ve değerlendirme için.
- **matplotlib** ve **seaborn**: Grafik çizimi ve veri görselleştirme için.

2. Veri Kümesinin Yüklenmesi

Veri kümesi bir CSV dosyasından pandas DataFrame olarak yüklenir. Bu veri kümesi, diyabet teşhisini için kullanılan çeşitli tıbbi ölçümleri içerir.

3. Veri Setinin Eğitim ve Test Olarak Bölünmesi

Veri kümesi, eğitim ve test veri setlerine ayrılır. Eğitim seti modelin öğrenmesi için kullanılırken, test seti modelin performansını değerlendirmek için kullanılır. Bu bölümleme, verinin %70'i eğitim, %30'u test seti olarak belirlenerek yapılır.

4. Özelliklerin ve Etiketlerin Ayrılması

Eğitim ve test veri setlerinden bağımsız değişkenler (özellikler) ve bağımlı değişken (hedef) ayrılır. Özellikler (bağımsız değişkenler) modelin tahmin etmek için kullandığı verileri, hedef (bağımlı değişken) ise modelin tahmin etmeye çalıştığı sınıfları içerir.

5. Multi-Layer Perceptron (MLP) Modelinin Eğitilmesi

MLP sınıflandırıcısı oluşturulur ve eğitim veri seti kullanılarak model eğitilir. MLP, yapay sinir ağları temelli bir modeldir ve belirli sayıda iterasyon (bu kodda 300) boyunca eğitilir. Eğitim sonrası, test seti üzerinde tahminler yapılır ve bu tahminlerin olasılıkları hesaplanır.

6. Support Vector Machines (SVM) Modelinin Eğitilmesi

SVM sınıflandırıcısı oluşturulur ve eğitim veri seti kullanılarak model eğitilir. SVM, sınıflar arasında en iyi ayırma hiper düzlemini bulmaya çalışan bir modeldir. Eğitim sonrası, test seti üzerinde tahminler yapılır ve bu tahminlerin olasılıkları hesaplanır.

7. Performans Metriklerinin Hesaplanması

Her iki model için aşağıdaki performans metrikleri hesaplanır:

- **Konfüzyon Matrisi**: Gerçek sınıflar ve modelin tahmin ettiği sınıflar arasındaki ilişkiye gösterir.
- **Sınıflandırma Raporu**: Doğruluk, kesinlik, duyarlılık ve F1 skoru gibi performans metriklerini içerir.
- ROC Eğrisi ve AUC (ROC Eğrisi Altındaki Alan): Modelin ayırt etme yeteneğini gösterir.

8. Sonuçların Kaydedilmesi

Her iki modelin performans metrikleri, ayrı ayrı CSV dosyalarına kaydedilir. Bu dosyalar, her modelin konfüzyon matrisi, sınıflandırma raporu ve ROC AUC skorunu içerir.

9. Grafiklerin Oluşturulması

- Konfüzyon Matrisi Grafikleri:** Her iki modelin konfüzyon matrisleri görselleştirilir ve tek bir grafik dosyasına kaydedilir.
- ROC Eğrisi Grafikleri:** Her iki modelin ROC eğrileri çizilir ve tek bir grafik dosyasına kaydedilir.

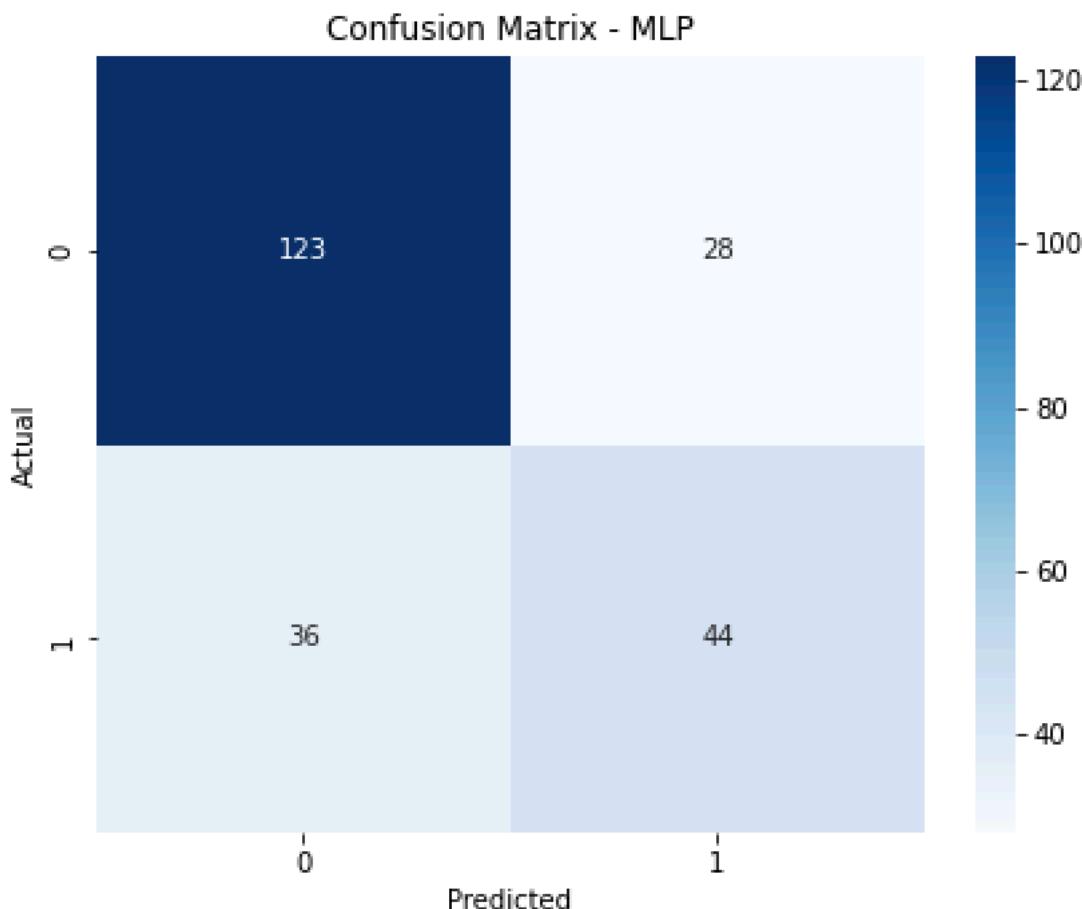
10. Sonuçların Döndürülmesi

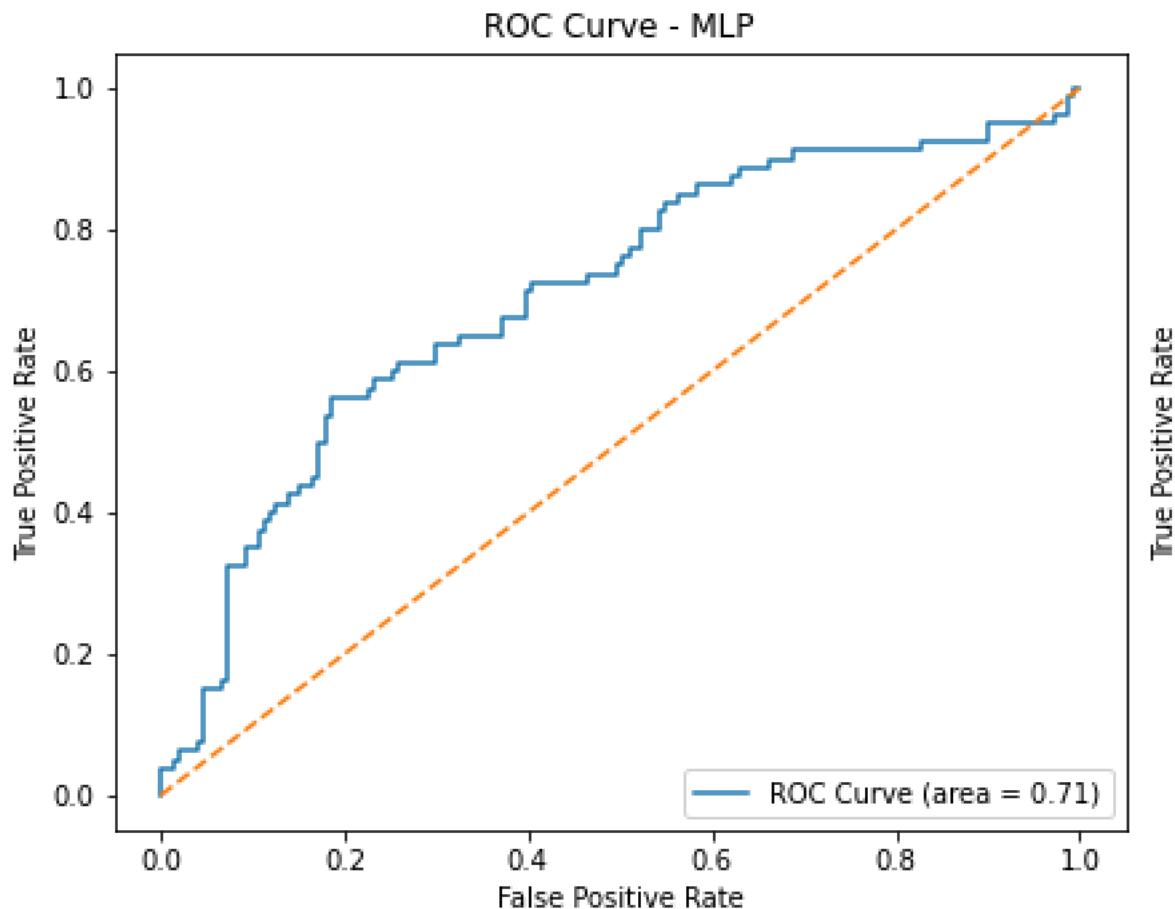
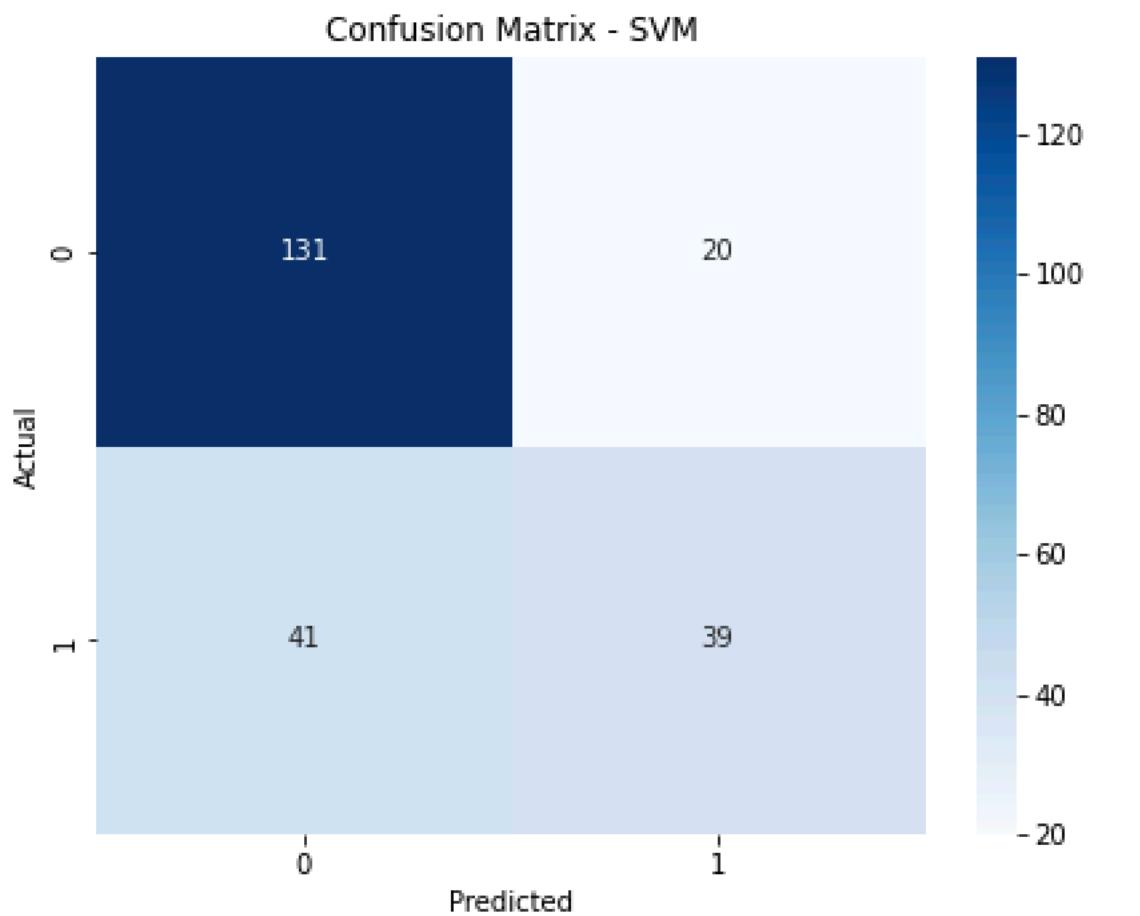
Kodun sonunda, her iki modelin konfüzyon matrisi, sınıflandırma raporu ve ROC AUC skorları döndürülür. Bu, modellerin performansını analiz etmek için kullanılabilir.

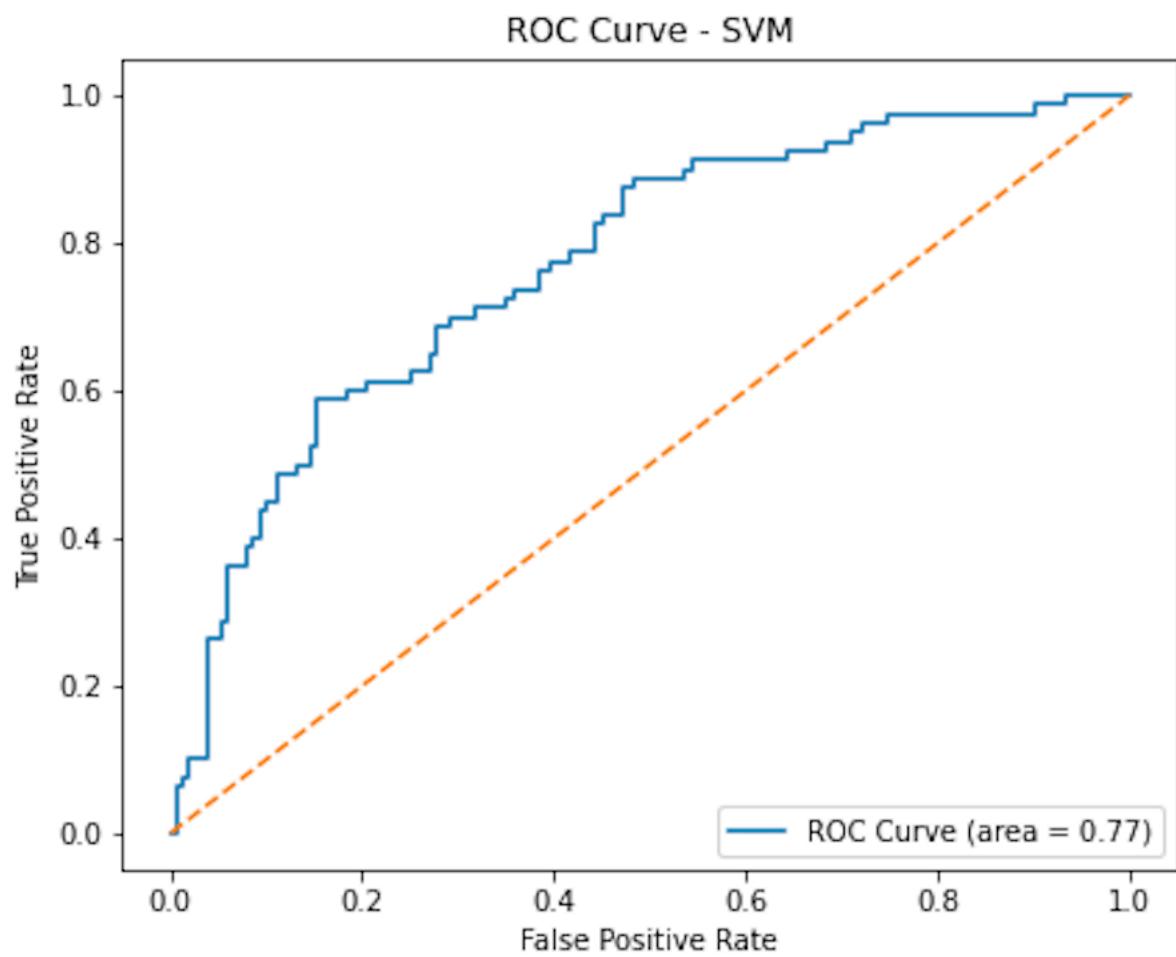
4. MADDE KODU ÇIKTILARI

classification_report_mlp		
Confusion Matrix	Classification Report	ROC AUC
[[123 28] [36 44]]	{'0': {'precision': 0.7735849056603774, 'recall': 0.8145695364238411, 'f1-score': 0.7935483870967743, 'support': 151.0}, '1': {'precision': 0.6111111111111112, 'recall': 0.55, 'f1-score': 0.5789473684210527, 'support': 80.0}}	0.707864238410596

classification_report_svm		
Confusion Matrix	Classification Report	ROC AUC
[[131 20] [41 39]]	{'0': {'precision': 0.7616279069767442, 'recall': 0.8675496688741722, 'f1-score': 0.8111455108359134, 'support': 151.0}, '1': {'precision': 0.6610169491525424, 'recall': 0.4875, 'f1-score': 0.5611510791366906, 'support': 80.0}}	0.7721854304635762







KAYNAKÇA

https://tr.wikipedia.org/wiki/Makine_%C4%9E%C4%9Frenimi

Makine_%C4%9E%C4%9Frenimi#:~:text=Makine%20%C4%9E%C4%9Frenimi%2C%20bilgisayarlar%C4%B1n%20de
neyimlerinden%20%C4%9E%C4%9Frenerek,ve%20tahminlerde%20bulunma%20yetene%C4%9Ene%
20dayan%C4%B1r.

<https://www.oracle.com/tr/artificial-intelligence/machine-learning/what-is-machine-learning/>

<https://www.ibm.com/topics/naive-bayes#:~:text=Na%C3%ADve%20Bayes%20is%20part%20of,important%20to%20differentiate%20between%20classes.>

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://en.wikipedia.org/wiki/Multilayer_perceptron

<https://isikhanelif.medium.com/multi-layer-perceptron-mlp-nedir-4758285a7f15>