FET340 – MAKİNE ÖĞRENMESİ
FİNAL ÖDEVİ

21040301034 SİNAN ENGİN YILDIZ

İstanbul Topkapı Üniversitesi Yazılım
Mühendisliği Bölümü
FET340 Dersi Final Ödevi

2024

ÖDEV SORULARI

***FİNAL ÖDEVİ TOPKAPI ALMSCLOUD A YUKLENMELIDIR.**

***ODEVIN ÇIKTISI ALINARAK ELDEN TESLIM EDILMELI VE ODEV YOKLAMA TUTANAĞI İMZALANCAKTIR. TESLIM EDILMEYEN VE YOKLAMA IMZALANMAYAN ÖDEVLER GECERSIZ SAYILACAKTIR.**

***ÖDEV KODLARI VE PERFORMANS ÖLCÜTLERİ İÇİN HESAPLANACAK PERFORMANS METRİKLERİNİN (ROC EĞRİLERİ CONFUSION MATRIX VE SENSITIVITY, SPECIFICITY, RECALL VS) ÇIKTILARI GITHUB LINKI HALINDE VERILMELIDIR**

1. **Size atanan veri setine tüm algoritmalar için optimizasyon uygulayınız.**

2. **Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Eğitim veri seti için Naive bayes sınıflandırıcısını uygulayınız. Elde ettiğiniz sonucları raporlayınız.**

3. **Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Eğitim veri seti için K-en yakın komşuluk sınıflandırıcısını uygulayınız. En iyi k değerini belirleyerek Elde ettiğiniz sonuçları raporlayınız.**

4. **Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Eğitim veri seti için Multi-Layer Perceptron (MLP) ve Support Vector Machines (SVM) sınıflandırıcılarını uygulayınız. Eğitim ve test adımlarında elde ettiğiniz sonucları raporlayınız.**

****Maddelerde Sonuçları raporlama olarak adlandırılan kavram sonuçların konfuzyon matrisinin verilmesi, sensitivity/specificity/accuracy/f1-score veya ROC curve gibi parametreleri ile açıklanması anlamına gelmektedir.**

Dataset

Pima Indians Diabetes Dataset

The Pima Indians Diabetes Dataset involves predicting the onset of diabetes within 5 years in Pima Indians given medical details.

It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 768 observations with 8 input variables and 1 output variable. Missing values are believed to be encoded with zero values. The variable names are as follows:

Number of times pregnant.
Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
Diastolic blood pressure (mm Hg).
Triceps skinfold thickness (mm).
2-Hour serum insulin (mu U/ml).
Body mass index (weight in kg/(height in m)^2).
Diabetes pedigree function.
Age (years).
Class variable (0 or 1)

CEVAPLAR

Açıklama:

Ödevin Jupyter Notebook dosyası içerisindeki tüm kodlarla beraber ekte verilmiştir. Jupyter Notebook uygulaması üzerinden yazılmış kodların üst kısımlarında genel bir açıklama yer almaktadır.

Soru 1

```python
# Kütüphaneler
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# Veri setinin yüklenmesi ve sütun isimlerinin verilmesi
data = pd.read_csv('veri-seti.txt', sep='\t')
data.columns = ['Np', 'Pg', 'Dbp', 'Tst', '2Si', 'Bmi', 'Dpf', 'Age', 'Outcome']

# Eksik verilerin kontrol edilmesi
print(data.isnull().sum())

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

# Eğitim ve test verisinin ayrılması
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Verilerin standartlaştırılması
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model ve hiperparametrelerin tanımlanması
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC()
}

param_grids = {
    'Logistic Regression': {'C': [0.1, 1, 10, 100]},
    'Decision Tree': {'max_depth': [3, 5, 7, 10]},
    'Random Forest': {'n_estimators': [50, 100, 200], 'max_depth': [3, 5, 7, 10]},
    'SVM': {'C': [0.1, 1, 10, 100], 'kernel': ['linear', 'rbf']}
}

# Modellerin eğitilmesi ve optimizasyonu
best_models = {}
for model_name, model in models.items():
    print(f"Training {model_name}...")
    grid_search = GridSearchCV(model, param_grids[model_name], cv=5, scoring='accuracy')
    grid_search.fit(X_train, y_train)
    best_models[model_name] = grid_search.best_estimator_
    print(f"Best params for {model_name}: {grid_search.best_params_}")
    print(f"Best cross-validation accuracy for {model_name}: {grid_search.best_score_}")

# Modellerin test verisi üzerinde değerlendirilmesi
for model_name, model in best_models.items():
    y_pred = model.predict(X_test)
    print(f"Results for {model_name}:")
    print(classification_report(y_test, y_pred))
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

Kütüphanelerin entegre edilmesinin ardından, model test ve train olarak ayrılmıştır. Ardından Logistic Regression, Decision Tree, Random Forest ve SVM kullanılarak optimize edilmiştir. Alınan sonuçlar alt kısımda yer almaktadır.

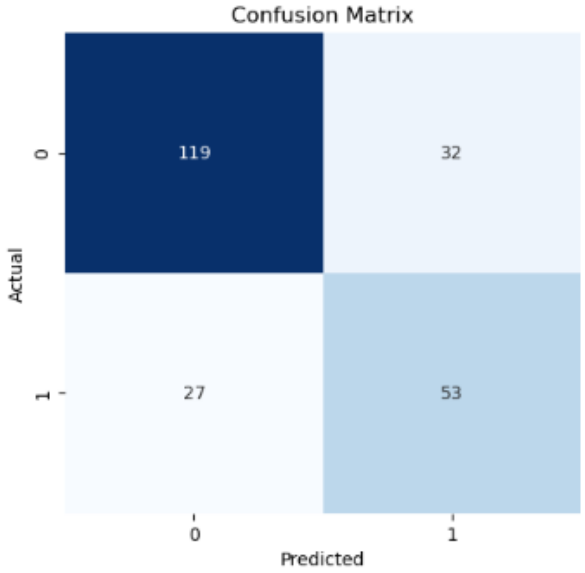
```
Np         0
Pg         0
Dbp        0
Tst        0
2Si        0
Bmi        0
Dpf        0
Age        0
Outcome    0
dtype: int64
Training Logistic Regression...
Best params for Logistic Regression: {'C': 10}
Best cross-validation accuracy for Logistic Regression: 0.7655071304811408
Training Decision Tree...
Best params for Decision Tree: {'max_depth': 3}
Best cross-validation accuracy for Decision Tree: 0.7573903771824604
Training Random Forest...
Best params for Random Forest: {'max_depth': 7, 'n_estimators': 50}
Best cross-validation accuracy for Random Forest: 0.7866853258696522
Training SVM...
Best params for SVM: {'C': 1, 'kernel': 'rbf'}
Best cross-validation accuracy for SVM: 0.7687458349993335
Results for Logistic Regression:
              precision    recall  f1-score   support

           0       0.81      0.80      0.81        99
           1       0.65      0.67      0.66        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154

Accuracy: 0.7532467532467533
Results for Decision Tree:
              precision    recall  f1-score   support

           0       0.80      0.84      0.82        99
           1       0.68      0.62      0.65        55

    accuracy                           0.76       154
   macro avg       0.74      0.73      0.73       154
weighted avg       0.76      0.76      0.76       154

Accuracy: 0.7597402597402597
Results for Random Forest:
              precision    recall  f1-score   support

           0       0.79      0.78      0.79        99
           1       0.61      0.64      0.62        55

    accuracy                           0.73       154
   macro avg       0.70      0.71      0.71       154
weighted avg       0.73      0.73      0.73       154

Accuracy: 0.7272727272727273
Results for SVM:
              precision    recall  f1-score   support

           0       0.77      0.83      0.80        99
           1       0.65      0.56      0.60        55

    accuracy                           0.73       154
   macro avg       0.71      0.70      0.70       154
weighted avg       0.73      0.73      0.73       154

Accuracy: 0.7337662337662337
```

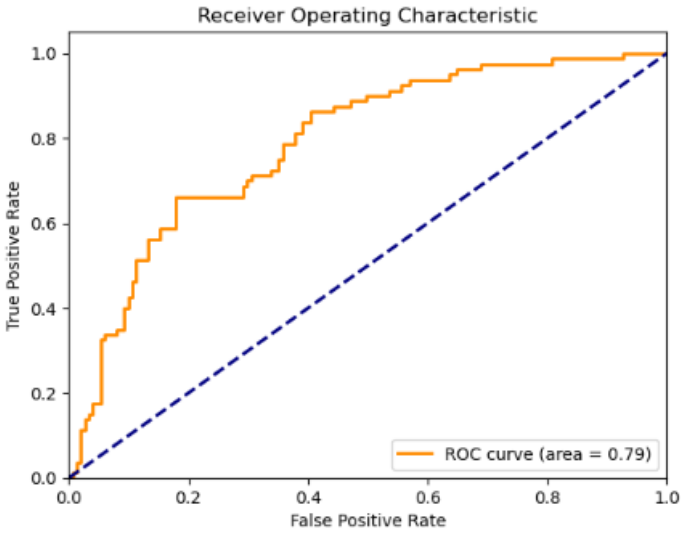
Tüm tekniklerden alınan sonuçlar doğrultusunda en başarılı sonucu küçük bir farkla Logistic Regression tekniği vermiştir.

Soru 2

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns

# Veri setinin yüklenmesi
data = pd.read_csv('veri-seti.txt', sep='\t')
data.columns = ['Np', 'Pg', 'Dbp', 'Tst', '2Si', 'Bmi', 'Dpf', 'Age', 'Outcome']

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

# Eğitim ve test verisinin ayrılması (%70 eğitim, %30 test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Verilerin standartlaştırılması
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Naive Bayes sınıflandırıcısının oluşturulması ve eğitilmesi
nb = GaussianNB()
nb.fit(X_train, y_train)

# Test verisi üzerinde tahmin yapılması
y_pred = nb.predict(X_test)
# Sonuçların raporlanması
# Konfüzyon matrisi
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Sınıflandırma raporu
print(classification_report(y_test, y_pred))

# Sensitivity, Specificity, Accuracy ve F1-Score hesaplanması
tn, fp, fn, tp = cm.ravel()
sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)
accuracy = (tp + tn) / (tp + tn + fp + fn)
f1 = 2 * tp / (2 * tp + fp + fn)

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"Accuracy: {accuracy}")
print(f"F1-Score: {f1}")

# ROC eğrisi ve AUC
y_prob = nb.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = roc_auc_score(y_test, y_prob)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', 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.show()
print(f"ROC AUC: {roc_auc}")
```

Bu soru için yine önceki soru ile benzer adımlar uygulanmıştır. Ardından Naive Bayes sınıflandırıcısı oluşturulmuş ve eğitilmiştir.





ROC AUC: 0.7904801324503311

	precision	recall	f1-score	support
0	0.82	0.79	0.80	151
1	0.62	0.66	0.64	80
accuracy			0.74	231
macro avg	0.72	0.73	0.72	231
weighted avg	0.75	0.74	0.75	231

Sensitivity: 0.6625
Specificity: 0.7880794701986755
Accuracy: 0.7445887445887446
F1-Score: 0.6424242424242425

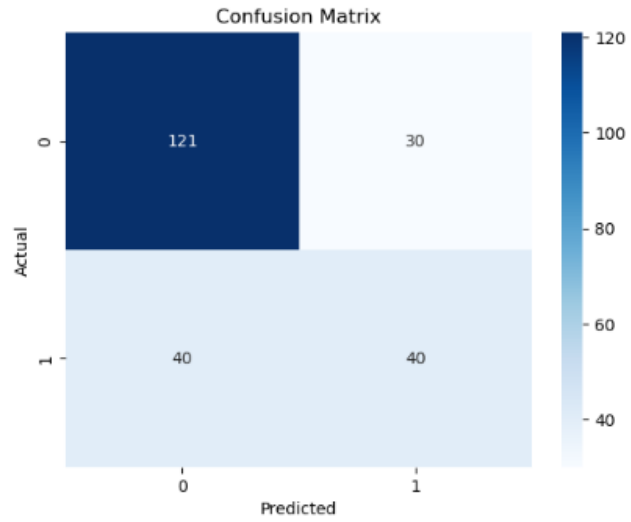
Aldığımız sonuçlardan üretilmiş Confusion Matrix ve ROC eğrisi yukarıdaki görsellerde verilmiştir. Aynı zamanda Confusion Matrix'in alt kısmında başarının ölçülmesi için kullanılan metriklerde gözlemlenebilmektedir.

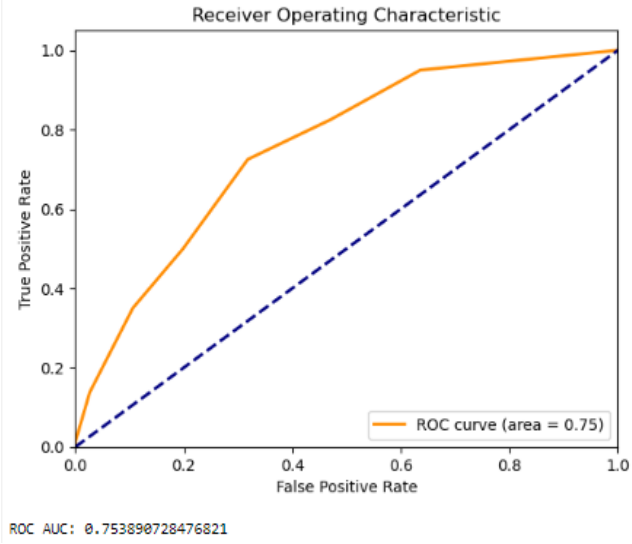
Soru 3

```python
# Gerekli kütüphanelerin yüklenmesi
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns

# Veri setinin yüklenmesi
data = pd.read_csv('veri-seti.txt', sep='\t')

# Sütun isimlerinin verilmesi
data.columns = ['Np', 'Pg', 'Dbp', 'Tst', '2Si', 'Bmi', 'Dpf', 'Age', 'Outcome']

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

# Eğitim ve test verisinin ayrılması (%70 eğitim, %30 test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Verilerin standartlaştırılması
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# KNN sınıflandırıcısı ve GridSearchCV ile en iyi k değerinin bulunması
knn = KNeighborsClassifier()
param_grid = {'n_neighbors': np.arange(1, 31)}
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# En iyi k değerinin alınması
best_k = grid_search.best_params_['n_neighbors']
print(f"Best k value: {best_k}")

# En iyi KNN modeli ile eğitim
best_knn = KNeighborsClassifier(n_neighbors=best_k)
best_knn.fit(X_train, y_train)

# Test verisi üzerinde tahmin yapılması
y_pred = best_knn.predict(X_test)
```

```python
# Sonuçların raporlanması
# Konfüzyon matrisi
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Sınıflandırma raporu
print(classification_report(y_test, y_pred))

# Sensitivity, Specificity, Accuracy ve F1-Score hesaplanması
tn, fp, fn, tp = cm.ravel()
sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)
accuracy = (tp + tn) / (tp + tn + fp + fn)
f1 = 2 * tp / (2 * tp + fp + fn)

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"Accuracy: {accuracy}")
print(f"F1-Score: {f1}")

# ROC eğrisi ve AUC
y_prob = best_knn.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = roc_auc_score(y_test, y_prob)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', 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.show()

print(f"ROC AUC: {roc_auc}")
```

Bu soruda ise veri seti uygun şekle getirildikten sonra K - en yakın komşuluk sınıflandırıcısı kullanılarak model üretilmişitr.





ROC AUC: 0.753890728476821

	precision	recall	f1-score	support
0	0.75	0.80	0.78	151
1	0.57	0.50	0.53	80
accuracy			0.70	231
macro avg	0.66	0.65	0.65	231
weighted avg	0.69	0.70	0.69	231

Sensitivity: 0.5
Specificity: 0.8013245033112583
Accuracy: 0.696969696969697
F1-Score: 0.5333333333333333

K – en yakın komşuluk sınıflandırıcısı kullanılarak elde edilen modelin başarı değerleri, Confusion Matrix ve ROC eğrisi yukarıda verilmişitr.

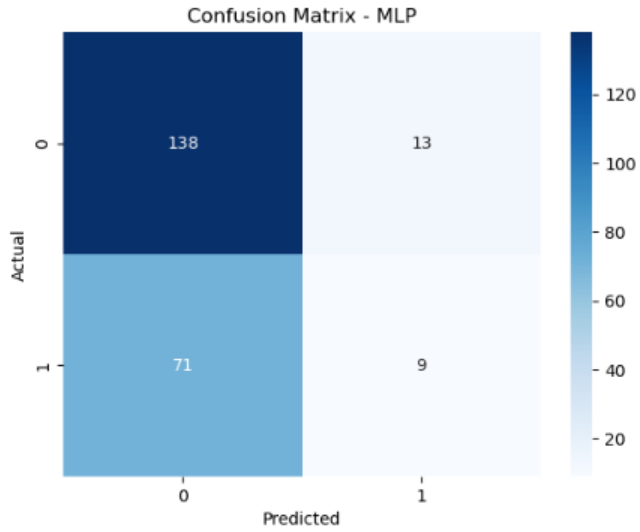
Soru 4

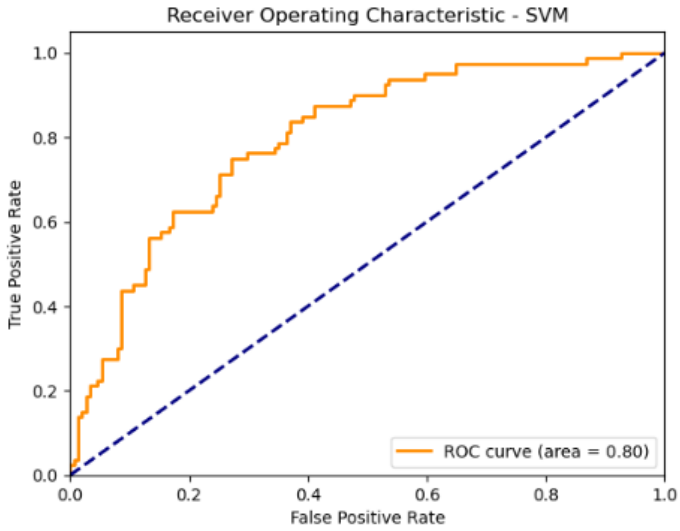
```python
# Gerekli kütüphanelerin yüklenmesi
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns

# Veri setinin yüklenmesi
data = pd.read_csv('veri-seti.txt', sep='\t')

# Sütun isimlerinin verilmesi
data.columns = ['Np', 'Pg', 'Dbp', 'Tst', '2Si', 'Bmi', 'Dpf', 'Age', 'Outcome']

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

# Eğitim ve test verisinin ayrılması (%70 eğitim, %30 test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Verilerin standartlaştırılması
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Multi-Layer Perceptron (MLP) sınıflandırıcısının oluşturulması ve eğitilmesi
mlp = MLPClassifier(random_state=42, max_iter=3000, learning_rate_init=0.0001, early_stopping=True, n_iter_no_change=10)
mlp.fit(X_train, y_train)

# Test verisi üzerinde tahmin yapılması (MLP)
y_pred_mlp = mlp.predict(X_test)

# Support Vector Machines (SVM) sınıflandırıcısının oluşturulması ve eğitilmesi
svm = SVC(probability=True, random_state=42)
svm.fit(X_train, y_train)

# Test verisi üzerinde tahmin yapılması (SVM)
y_pred_svm = svm.predict(X_test)
```

```python
# Sonuçların raporlanması
def report_results(y_test, y_pred, y_prob, model_name):
    print(f"Results for {model_name}:")

    # Konfüzyon matrisi
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix - {model_name}')
    plt.show()

    # Sınıflandırma raporu
    print(classification_report(y_test, y_pred))

    # Sensitivity, Specificity, Accuracy ve F1-Score hesaplanması
    tn, fp, fn, tp = cm.ravel()
    sensitivity = tp / (tp + fn)
    specificity = tn / (tn + fp)
    accuracy = (tp + tn) / (tp + tn + fp + fn)
    f1 = 2 * tp / (2 * tp + fp + fn)

    print(f"Sensitivity: {sensitivity}")
    print(f"Specificity: {specificity}")
    print(f"Accuracy: {accuracy}")
    print(f"F1-Score: {f1}")

    # ROC eğrisi ve AUC
    fpr, tpr, thresholds = roc_curve(y_test, y_prob)
    roc_auc = roc_auc_score(y_test, y_prob)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', 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(f'Receiver Operating Characteristic - {model_name}')
    plt.legend(loc="lower right")
    plt.show()

    print(f"ROC AUC: {roc_auc}")

# MLP sonuçlarının raporlanması
report_results(y_test, y_pred_mlp, mlp.predict_proba(X_test)[:, 1], "MLP")

# SVM sonuçlarının raporlanması
report_results(y_test, y_pred_svm, svm.predict_proba(X_test)[:, 1], "SVM")
```

. Bu soru için yine veri setinin test ve train olarak ayrılmasının ardından. Multi-Layer Perceptron (MLP) ve Support Vector Machines (SVM) sınıflandırıcıları kullanılmıştır ve sonuçlar raporlandırılmıştır.

```
Results for MLP:
```

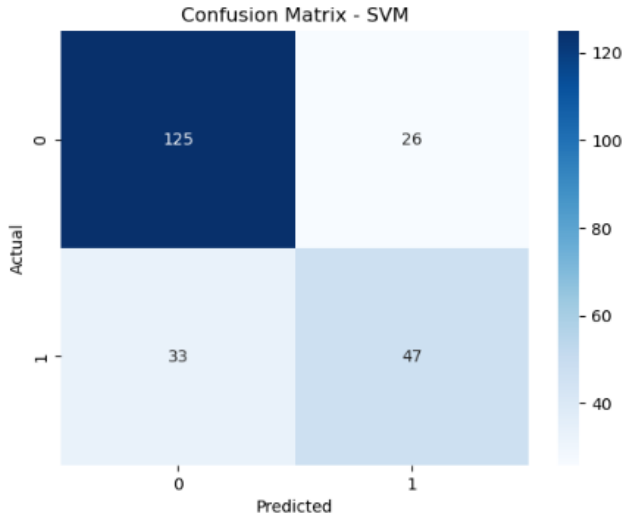


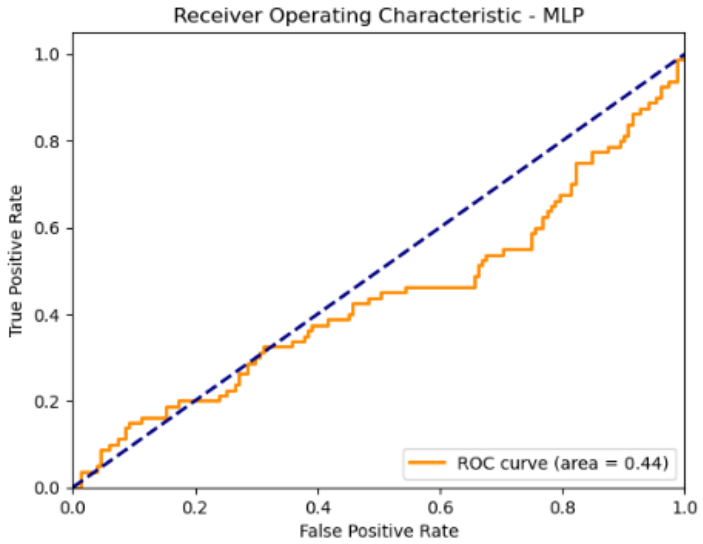



```
              precision    recall  f1-score   support

           0       0.66      0.91      0.77       151
           1       0.41      0.11      0.18        80

    accuracy                           0.64       231
   macro avg       0.53      0.51      0.47       231
weighted avg       0.57      0.64      0.56       231

Sensitivity: 0.1125
Specificity: 0.9139072847682119
Accuracy: 0.6363636363636364
F1-Score: 0.17647058823529413
```

```
ROC AUC: 0.7973509933774834
```

```
Results for SVM:
```






```
              precision    recall  f1-score   support

           0       0.79      0.83      0.81       151
           1       0.64      0.59      0.61        80

    accuracy                           0.74       231
   macro avg       0.72      0.71      0.71       231
weighted avg       0.74      0.74      0.74       231

Sensitivity: 0.5875
Specificity: 0.8278145695364238
Accuracy: 0.7445887445887446
F1-Score: 0.6143790849673203
```

```
ROC AUC: 0.4406456953642384
```

MLP ve SVM sınıflandırıcıları ile elde edilmiş modelin, başarı metrikleri, Confusion Matrix ve ROC eğrileri yukarıda sırasıyla verilmiştir.