

ÜSKÜDAR ÜNİVERSİTESİ

**MAKİNE ÖĞRENMESİ**

**FİNAL PROJE ÖDEVİ**

**GENEL BİLGİLER**

|  |
| --- |
| **Öğrencinin Adı-Soyadı: Mehmet ORTACEBİN** |
| **Anabilim Dalı: Fen Bilimleri Enstitüsü** |
| **Programı: Yapay Zekâ Mühendisliği** |
| **Öğrenci Numarası: 234312019** |
| **Öğretim Üyesi Adı-Soyadı: Dr.Öğr. Üyesi Gökalp TULUM** |

**İÇİNDEKİLER**

[**1.** **Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız.** 1](#_Toc167636366)

[**2.** **Eğitim veri seti için Naive bayes sınıflandırıcısını uygulayınız. Elde ettiğiniz sonuçları raporlayınız.** 2](#_Toc167636367)

[**3.** **Eğitim veri seti için K-en yakın komşuluk sınıflandırıcısını uygulayınız. En iyi k değerini belirleyerek Elde ettiğiniz sonuçları raporlayınız.** 5](#_Toc167636368)

[**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 raporlayıız.** 9](#_Toc167636369)

[**5.** **Size atanan veri setine tüm algoritmalar için optimizasyon uygulayınız.** 16](#_Toc167636370)

[**6.** **Eğitim veri seti için Naive bayes sınıflandırıcısını uygulayınız. Elde ettiğiniz sonucları raporlayınız. Test verisi için ortalama karesel hatayı hesaplayınız.** 16](#_Toc167636371)

[**7.** **Eğitim veri seti için K-en yakın komşuluk sınıflandırıcısını uygulayınız. En iyi k değerini belirleyerek Elde ettiğiniz sonuçları raporlayınız. Test verisi için ortalama karesel hatayı hesaplayınız.** 20](#_Toc167636372)

[**8.** **Eğitim veri seti için Multi-Layer Perceptron (MLP) ve Support Vector Machines (SVM) sınıflandırıcılarını uygulayınız. Test verisi için ortalama karesel hatayı hesaplayınız. Eğitim ve test adımlarında elde ettiğiniz sonucları raporlayınız.** 24](#_Toc167636373)

**MAKİNE ÖĞRENMESİ FİNAL ÖDEVİ**

# Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız.

**KOD:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, roc\_auc\_score, roc\_curve, classification\_report

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

import seaborn as sns  # seaborn kütüphanesini import ediyoruz

# Veri setini yükle

data = pd.read\_csv('/content/drive/MyDrive/ColabNotebooks/MachineLearning/final/dataset/veri-seti.txt', sep="\t", header=None)

data.columns = ["Hamilelik", "Glikoz", "Kan Basinci", "Cilt Kalınlığı", "İnsülin", "BMI", "Diyabet Fonksiyonu", "Yaş", "Sonuc"]

# Veri setinin ilk birkaç satırını göster

print(data.head())

"""#Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız."""

# Veri setini özellikler (X) ve etiketler (y) olarak ayır

X = data.drop("Sonuc", axis=1)

y = data["Sonuc"]

# Veriyi eğitim (%70) ve test (%30) olarak ayır

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**ÇIKTI:**

**metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

# Eğitim veri seti için Naive bayes sınıflandırıcısını uygulayınız. Elde ettiğiniz sonuçları raporlayınız.

**KOD:**

# Gaussian Naive Bayes sınıflandırıcısını başlat

nb\_classifier = GaussianNB()

# Sınıflandırıcıyı eğit

nb\_classifier.fit(X\_train, y\_train)

# Test seti üzerinde tahminlerde bulun

y\_pred = nb\_classifier.predict(X\_test)

# Değerlendirme metriklerini hesapla

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

# Karışıklık matrisi görselleştirme

print("Confusion Matrix:")

print(conf\_matrix)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.xlabel("Tahmin Edilen")

plt.ylabel("Gerçek")

plt.title("Karışıklık Matrisi")

plt.show()

# Sınıflandırma raporunu göster

print("Sınıflandırma Raporu:")

print(class\_report)

# ROC eğrisini hesapla

y\_prob = nb\_classifier.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

# ROC eğrisini çiz

plt.figure()

plt.plot(fpr, tpr, color='blue', lw=2, label='ROC eğrisi (alan = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('Yanlış Pozitif Oranı')

plt.ylabel('Doğru Pozitif Oranı')

plt.title('Alıcı İşletim Karakteristiği (ROC) Eğrisi')

plt.legend(loc="lower right")

plt.show()

#ROC AUC Değeri Hesapla

roc\_auc = roc\_auc\_score(y\_test, y\_pred)

print(f"ROC AUC: {roc\_auc}")

**ÇIKTI:**

**metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM:**

Bu sonuçlar, Naive Bayes sınıflandırıcısının performansını değerlendirmek için oldukça bilgilendirici. Confusion matrix'e baktığımızda, modelin 119 doğru tahmini (true negatives) ve 53 doğru tahmini (true positives) olduğunu görüyoruz. Ancak, 32 false negatives ve 27 false positives olduğunu görmekteyiz. Bu, modelin bazı olumlu örnekleri olumsuz olarak sınıflandırdığını ve bazı olumsuz örnekleri ise olumlu olarak sınıflandırdığını göstermektedir.

Sınıflandırma raporuna göz attığımızda, precision (kesinlik), recall (duyarlılık) ve f1-score (f1-puanı) gibi metrikler üzerinden sınıflandırıcının performansını daha detaylı bir şekilde değerlendirebiliriz. Precision, pozitif olarak tahmin edilen örneklerin ne kadarının gerçekten pozitif olduğunu gösterirken, recall, gerçek pozitiflerin ne kadarının doğru bir şekilde sınıflandırıldığını gösterir. F1-score ise precision ve recall'in harmonik ortalamasıdır, bu nedenle hem precision hem de recall arasındaki dengeyi sağlar.

**metin, çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu**Bu değerlendirmeye göre, sınıflandırıcının sınıf 0 için precision'u 0.82 ve recall'u 0.79 iken, sınıf 1 için precision'u 0.62 ve recall'u 0.66'dır. Bu, modelin sınıf 0'ı sınıflandırmada daha iyi performans gösterdiğini, ancak sınıf 1'i sınıflandırmada daha düşük performans gösterdiğini göstermektedir.

**YORUM:**

ROC AUC (Receiver Operating Characteristic Area Under Curve) değeri, sınıflandırıcının sınıflandırma performansının genel bir ölçüsüdür. 0 ile 1 arasında bir değer alır, 1'e ne kadar yakınsa, modelin performansı o kadar iyidir. Bu durumda, ROC AUC değeri 0.725 olarak hesaplanmıştır, yani modelin oldukça iyi bir performans sergilediğini ancak mükemmel olmadığını göstermektedir.

Sonuç olarak, Naive Bayes sınıflandırıcısı, belirli bir veri kümesi için makul bir performans sergilemektedir, ancak sınıf dengesizliği veya diğer faktörler nedeniyle bazı iyileştirmelere ihtiyaç duyabilir.

# **3.** Eğitim veri seti için K-en yakın komşuluk sınıflandırıcısını uygulayınız. En iyi k değerini belirleyerek Elde ettiğiniz sonuçları raporlayınız.

**KOD:**

k\_range = range(1, 31)

k\_scores = []

for k in k\_range:

    knn = KNeighborsClassifier(n\_neighbors=k)

    scores = cross\_val\_score(knn, X\_train, y\_train, cv=10, scoring='accuracy')

    k\_scores.append(scores.mean())

best\_k = k\_range[np.argmax(k\_scores)]

print(f"En iyi k değeri: {best\_k}")

# En iyi k değeri ile modeli eğit

knn = KNeighborsClassifier(n\_neighbors=best\_k)

knn.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, classification\_report, roc\_auc\_score, roc\_curve

# Test seti üzerinde tahmin yapma

y\_pred = knn.predict(X\_test)

# Konfüzyon matrisi

print("Confusion Matrix:")

print(conf\_matrix)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.title("Konfüzyon Matrisi")

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek Sınıf')

plt.show()

# Sınıflandırma raporu

print("Sınıflandırma Raporu:")

print(classification\_report(y\_test, y\_pred))

# ROC Eğrisi ve AUC

y\_pred\_proba = knn.predict\_proba(X\_test)[:, 1] # Sadece pozitif sınıfın olasılıklarını alır

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_proba)

auc = roc\_auc\_score(y\_test, y\_pred\_proba)

# ROC Eğrisini çizme

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label='ROC Curve (AUC = {:.2f})'.format(auc))

plt.plot([0, 1], [0, 1],  color='red', linestyle='--')  # Rastgele sınıflandırıcı için referans çizgisi

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')

plt.title('ROC Curve')

plt.legend(loc='lower right')

plt.show()

print("ROC AUC:", auc)

**ÇIKTI:**





metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**YORUM:**

1. **Konfuzyon Matrisi (Confusion Matrix):** Karmaşıklık matrisi, modelinizin doğru ve yanlış sınıflandırma sayılarını gösterir. Pozitif sınıf (1) için 53 doğru sınıflandırma ve 27 yanlış sınıflandırma yapılmış. Negatif sınıf (0) için 119 doğru sınıflandırma ve 32 yanlış sınıflandırma yapılmış.
2. **Sınıflandırma Raporu:** Precision, recall ve f1-score gibi metrikler, sınıflandırma modelinin performansını ölçmek için kullanılır. Pozitif sınıf (1) için precision 0.62, recall 0.57 ve f1-score 0.60 olarak hesaplanmış. Negatif sınıf (0) için precision 0.78, recall 0.81 ve f1-score 0.80 olarak hesaplanmış. Weighted avg (ağırlıklı ortalama) doğruluk (accuracy) için 0.73 olarak hesaplanmış.

**SONUÇLARIN GENEL DEĞERLENDİRMESİ:**

Modelin negatif sınıfı (0) pozitif sınıf (1)'dan daha iyi sınıflandırdığını görüyoruz. Bu, modelin negatif sınıfı daha iyi ayırt ettiğini, ancak pozitif sınıfı daha az hassas bir şekilde sınıflandırdığını gösterebilir. Weighted avg doğruluk 0.73 olarak gösterilmiş, bu da modelin genel olarak kabul edilebilir bir performans sergilediğini ancak iyileştirme alanları olduğunu gösterebilir.

metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**YORUM:**

ROC AUC, sınıflandırma modelinin sınıflandırma yeteneğini ölçen bir metriktir. Değer ne kadar yüksekse, modelin performansı o kadar iyidir. 0.756 değeri, modelin genel olarak iyi performans gösterdiğini gösteriyor.

**SONUÇLARIN GENEL DEĞERLENDİRMESİ:**

ROC AUC değeri 0.75'in üzerinde olduğundan, modelin sınıflandırma yeteneği oldukça iyi olarak değerlendirilebilir.

Bu değerlendirme sonuçlarına dayanarak, modelin performansını daha da artırmak için pozitif sınıfı daha iyi sınıflandırmak için geliştirmeler yapabilir. Örneğin, farklı özniteliklerin veya model parametrelerinin kullanılması, modeli daha iyi hale getirebilir.

# 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.

**KOD:**

# Özellikleri ölçeklendir (MLP ve SVM için önemli)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

from sklearn.neural\_network import MLPClassifier

# MLP sınıflandırıcısını oluştur ve eğit

mlp = MLPClassifier(max\_iter=1000, random\_state=42)

mlp.fit(X\_train\_scaled, y\_train)

# Test seti üzerinde tahminlerde bulun

y\_pred\_mlp = mlp.predict(X\_test\_scaled)

# Değerlendirme metriklerini hesapla

conf\_matrix\_mlp = confusion\_matrix(y\_test, y\_pred\_mlp)

class\_report\_mlp = classification\_report(y\_test, y\_pred\_mlp)

# MLP Karışıklık matrisini göster

print("MLP Karışıklık Matrisi:")

print(conf\_matrix\_mlp)

# MLP karışıklık matrisi görselleştirme

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix\_mlp, annot=True, fmt="d", cmap="Blues", xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.xlabel("Tahmin Edilen")

plt.ylabel("Gerçek")

plt.title("MLP Karışıklık Matrisi")

plt.show()

#MLP Sınıflandırma raporunu göster

print("MLP Sınıflandırma Raporu:")

print(class\_report\_mlp)

import matplotlib.pyplot as plt

from sklearn.metrics import roc\_curve, roc\_auc\_score

# MLP ROC eğrisini hesapla ve çiz

y\_prob\_mlp = mlp.predict\_proba(X\_test\_scaled)[:, 1]

fpr\_mlp, tpr\_mlp, \_ = roc\_curve(y\_test, y\_prob\_mlp)

roc\_auc\_mlp = roc\_auc\_score(y\_test, y\_prob\_mlp)

plt.figure()

plt.plot(fpr\_mlp, tpr\_mlp, color='blue', lw=2, label='MLP ROC eğrisi (alan = %0.2f)' % roc\_auc\_mlp)

plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('Yanlış Pozitif Oranı')

plt.ylabel('Doğru Pozitif Oranı')

plt.title('MLP ROC Eğrisi')

plt.legend(loc="lower right")

plt.show()

print(f"MLP ROC AUC: {roc\_auc\_mlp}")

from sklearn.svm import SVC

# SVM sınıflandırıcısını oluştur ve eğit

svm = SVC(probability=True, random\_state=42)

svm.fit(X\_train\_scaled, y\_train)

# Test seti üzerinde tahminlerde bulun

y\_pred\_svm = svm.predict(X\_test\_scaled)

# Değerlendirme metriklerini hesapla

conf\_matrix\_svm = confusion\_matrix(y\_test, y\_pred\_svm)

class\_report\_svm = classification\_report(y\_test, y\_pred\_svm)

print("SVM Karışıklık Matrisi:\n", conf\_matrix\_svm)

# SVM karışıklık matrisi görselleştirme

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix\_svm, annot=True, fmt="d", cmap="Blues", xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.xlabel("Tahmin Edilen")

plt.ylabel("Gerçek")

plt.title("SVM Karışıklık Matrisi")

plt.show()

#SVM Diğer değerlendirme metriklerini göster

print("SVM Sınıflandırma Raporu:")

print(class\_report\_svm)

# SVM ROC eğrisini hesapla ve çiz

y\_prob\_svm = svm.predict\_proba(X\_test\_scaled)[:, 1]

fpr\_svm, tpr\_svm, \_ = roc\_curve(y\_test, y\_prob\_svm)

plt.figure()

plt.plot(fpr\_svm, tpr\_svm, color='blue', lw=2, label='SVM ROC eğrisi (alan = %0.2f)' % roc\_auc\_svm)

plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('Yanlış Pozitif Oranı')

plt.ylabel('Doğru Pozitif Oranı')

plt.title('SVM ROC Eğrisi')

plt.legend(loc="lower right")

plt.show()

#SVM ROC AUC DEğeri Hesaplama

print(f"SVM ROC AUC: {roc\_auc\_svm}")

**ÇIKTI:**

**metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM:**

Bu sonuçlar, eğitim veri setiniz üzerindeki MLP sınıflandırıcının performansını gösteriyor. Karışıklık matrisi ve sınıflandırma raporu, modelin sınıflandırma yeteneğini ayrıntılı bir şekilde analiz etmemize olanak tanır.

1. **Karışıklık Matrisi:**

* Gerçek sınıf 0 olan 151 örneğin 120'si doğru bir şekilde sınıflandırılmıştır, ancak 31'i yanlış sınıflandırılmıştır.
* Gerçek sınıf 1 olan 80 örneğin 42'si doğru bir şekilde sınıflandırılmıştır, ancak 38'i yanlış sınıflandırılmıştır.

1. **Sınıflandırma Raporu:**

* Precision (kesinlik): Modelin pozitif olarak tahmin ettiği örneklerin ne kadarının gerçekten pozitif olduğunu gösterir. Sınıf 0 için 0.76, sınıf 1 için 0.58'dir. Bu, sınıf 0'ı daha iyi tanımladığını ancak sınıf 1 için daha düşük bir kesinlik olduğunu gösterir.
* Recall (duyarlılık): Gerçek pozitif örneklerin ne kadarının doğru bir şekilde tanımlandığını gösterir. Sınıf 0 için 0.79, sınıf 1 için 0.53'tür. Sınıf 0'ın daha yüksek bir duyarlılığa sahip olduğunu, ancak sınıf 1 için düşük bir duyarlılık olduğunu gösterir.
* F1-score: Precision ve recall'ın harmonik ortalamasıdır. Sınıf 0 için 0.78, sınıf 1 için 0.55'tir.
* Accuracy (doğruluk): Toplam doğru tahminlerin oranını gösterir. Bu durumda, doğruluk 0.70'tir.

Genel olarak, modelin sınıf 0'ı daha iyi tanımlamaktadır ancak sınıf 1 için performansı daha zayıftır. Daha dengeli bir sınıflandırma için modeli iyileştirmek için özellik mühendisliği, hiperparametre ayarı veya model mimarisinde değişiklikler gibi yöntemler düşünebilir.

metin, çizgi, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

**YORUM:**

MLP (Multi-Layer Perceptron) sınıflandırıcısı ile elde ettiğiniz ROC AUC skoru 0.7597682119205298, modelinizin sınıflandırma performansı hakkında çeşitli ipuçları verir. ROC AUC (Receiver Operating Characteristic - Area Under Curve), bir modelin sınıflandırma başarısını değerlendiren önemli bir metriktir. 1'e yakın bir AUC skoru, modelin pozitif ve negatif sınıfları ayırt etmede çok başarılı olduğunu gösterirken, 0.5'e yakın bir skor, modelin rastgele tahmin ettiğini gösterir.

Sonuç olarak, MLP sınıflandırıcınızın elde ettiği ROC AUC skoru 0.7597682119205298, modelinizin makul bir performansa sahip olduğunu ve bazı iyileştirmelerle daha da iyi hale getirilebileceğini göstermektedir.

metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**YORUM:**

**Karışıklık matrisi:** Karışıklık matrisi, sınıflandırıcı tarafından doğru ve yanlış tahmin edilen örneklerin bir özetini sunar. 125 doğru negatif tahmin (true negatives) ve 47 doğru pozitif tahmin (true positives) var. 26 yanlış negatif tahmin (false negatives) ve 33 yanlış pozitif tahmin (false positives) var. Sınıflandırıcının sınıf 0'ı (negatif sınıf) daha iyi tahmin ettiği görülüyor, ancak sınıf 1'i (pozitif sınıf) daha zayıf tahmin ettiği anlaşılıyor.

**Sınıflandırma Raporu:** Precision, recall ve f1-score gibi metrikler, sınıflandırıcının performansını farklı açılardan değerlendirir. Precision, sınıflandırıcının pozitif olarak tahmin ettiği örneklerin ne kadarının gerçekten pozitif olduğunu gösterir. Sınıf 0 için 0.79 ve sınıf 1 için 0.64 olarak belirlenmiş. Recall, gerçek pozitif örneklerin ne kadarının doğru bir şekilde tanımlandığını gösterir. Sınıf 0 için 0.83 ve sınıf 1 için 0.59 olarak belirlenmiş. F1-score, precision ve recall metriklerinin harmonik ortalamasıdır ve bu nedenle her iki metriği de dengeli bir şekilde değerlendirir. Sınıf 0 için 0.81 ve sınıf 1 için 0.61 olarak belirlenmiş. Accuracy, doğru tahminlerin toplam örnek sayısına oranıdır ve bu durumda 0.74 olarak belirlenmiş.

**GENEL DEĞERLENDİRME:**

SVM sınıflandırıcısı, sınıf 0'ı (negatif sınıf) daha iyi tahmin ederken sınıf 1'i (pozitif sınıf) daha az başarıyla tahmin etmektedir. F1-score değerleri, sınıf 1 için daha düşük olduğundan, modelin pozitif sınıfı sınıflandırmakta daha zorlandığını gösteriyor. Bu sonuçlar, modelin sınıflandırma performansını iyileştirmek için daha fazla ayarlama veya daha fazla veri kullanma gerekliliğini gösterebilir. Ayrıca, dengesiz sınıf dağılımları varsa, örneğin, sınıf 1 daha az temsil ediliyorsa, bu da modelin performansını etkileyebilir.

metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**YORUM:**

Bu ROC AUC değeri, SVM sınıflandırıcının performansını değerlendirmek için kullanılan bir ölçüdür. 0 ile 1 arasında bir değer alır, 1'e ne kadar yakınsa, modelin o kadar iyi performans gösterdiği anlamına gelir. Bu durumda, 0.70 civarı bir ROC AUC değeri elde ettim, yani modelin genel olarak kabul edilebilir bir performans sergilediğini söyleyebiliriz.

Ancak, ROC AUC değerinin tek başına yorumlanması bazen yetersiz olabilir. Diğer metriklerle birlikte değerlendirilmesi daha sağlıklı bir sonuç verebilir. Ayrıca, problem domainine ve veri setine bağlı olarak, bu değerin tatmin edici olup olmadığı değişebilir. Örneğin, tıbbi bir uygulama için bu değer yeterli olmayabilirken, daha basit bir sınıflandırma probleminde kabul edilebilir olabilir.

# Size atanan veri setine tüm algoritmalar için optimizasyon uygulayınız.

GridSearchCV Hiperparametre Optimizasyonu kullanılmıştır. GridSearchCV araması, belirli bir modelde en iyi performansı veren değerleri bulmak için hiperparametre ayarlamasında kullanılan bir tekniktir. Bu teknikte kullanıcı tarafından verilen parametrelerin tüm kombinasyonları ile model eğitilir ve bulunan en iyi parametreler tüm modelin performansını etkilediği için önemli bir işlemdir. Makine öğrenmesi modellerinde, veri setine uygun en iyi parametreleri bulmak çok zorlu olabilmektedir.

# Eğitim veri seti için Naive bayes sınıflandırıcısını uygulayınız. Elde ettiğiniz sonucları raporlayınız. Test verisi için ortalama karesel hatayı hesaplayınız.

**KOD:**

from sklearn.naive\_bayes import GaussianNB

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import confusion\_matrix, classification\_report, mean\_squared\_error

nb\_classifier = GaussianNB()

param\_grid = {

    'var\_smoothing': [1e-09, 1e-08, 1e-07, 1e-06, 1e-05, 1e-04, 1e-03, 1e-02, 1e-01, 1.0]

}

grid\_search = GridSearchCV(estimator=nb\_classifier, param\_grid=param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

print("Bulunan en iyi parametreler: ", grid\_search.best\_params\_)

print("En iyi doğruluk: ", grid\_search.best\_score\_)

best\_nb\_classifier = grid\_search.best\_estimator\_

test\_accuracy = best\_nb\_classifier.score(X\_test, y\_test)

print("Test doğruluğu: ", test\_accuracy)

# Test verisi için ortalama karesel hata hesaplanması

y\_pred = best\_nb\_classifier.predict(X\_test)

# Confusion matrix'in hesaplanması

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Confusion matrix'in görselleştirilmesi

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",  xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])

plt.title("Konfuzyon Matrisi")

plt.xlabel("Tahmin Edilen")

plt.ylabel("Gerçek")

plt.show()

# Sınıflandırma raporunun hesaplanması ve yazdırılması

report = classification\_report(y\_test, y\_pred)

print("Sınıflandırma Raporu:\n", report)

mse = mean\_squared\_error(y\_test, y\_pred)

print("Test verisi için Ortalama Karesel Hata (MSE): ", mse)

# ROC eğrisi için false positive oranları (FPR) ve true positive oranları (TPR) hesaplanması

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

# ROC eğrisinin altında kalan alanın (AUC) hesaplanması

roc\_auc = roc\_auc\_score(y\_test, y\_pred)

# ROC eğrisinin çizdirilmesi

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='red', 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('ROC Curve')

plt.legend(loc="lower right")

plt.show()

print("ROC AUC Değeri:", roc\_auc)

**ÇIKTI:**

**metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

****

****

**metin, ekran görüntüsü, sayı, numara, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM:**

**1. Doğruluk ve Diğer Metrikler:**

* Hiperparametre optimizasyonu sonrası elde edilen test doğruluğu (%71.86), optimizasyon öncesine (%74.03) göre bir miktar düşmüş. Ancak GridSearchCV ile elde edilen en iyi doğruluk %76.16, bu da modelin potansiyel olarak daha yüksek performansa sahip olabileceğini gösteriyor.
* Precision, recall ve F1-score metriklerinde küçük değişiklikler gözlenmiş, ancak bu değişiklikler belirgin bir iyileşme veya kötüleşme göstermiyor.

**2. Konfuzyon Matrisi (Confusion Matrix)**

* True Positive (TP) ve True Negative (TN) sayılarında küçük değişiklikler olmuş, ancak genel olarak modelin sınıflandırma davranışında ciddi bir değişiklik gözlenmiyor.
* False Positive (FP) ve False Negative (FN) sayılarındaki değişiklikler de yine benzer düzeyde kalmış.

**3. Test verisi için Ortalama Karesel Hata (MSE):**

* Test verisi için Ortalama Karesel Hata (MSE) değeri 0.2814, bu da hatanın düşük seviyede olduğunu ve modelin hata oranının kabul edilebilir düzeyde olduğunu gösteriyor

***metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu***

**YORUM**

ROC AUC değeri de optimizasyon öncesinde 0.725 iken sonrasında 0.691’e düşmüş. Bu, modelin sınıflandırma yeteneğinin optimizasyon sonrası biraz azaldığını gösteriyor.

**SONUÇ VE DEĞERLENDİRME**

Hiperparametre optimizasyonu sonrasında, doğruluk ve ROC AUC gibi metriklerde küçük düşüşler yaşanmış. Ancak, bu durum modelin genel performansında büyük bir değişiklik olmadığını, sadece küçük varyasyonlar olduğunu gösteriyor. Ayrıca, GridSearchCV ile elde edilen en iyi doğruluk (%76.16) modelin potansiyelini gösteriyor ve bu da farklı veri bölünmeleriyle daha iyi sonuçlar alınabileceğini işaret ediyor.

Bu sonuçlara dayanarak, modelin hiperparametre optimizasyonunun genel performansa ciddi bir etkisi olmadığını söyleyebiliriz. Ancak, modelin performansını iyileştirmek için farklı hiperparametre aralıkları veya daha fazla veri ön işleme teknikleri denenebilir. Ayrıca, modelin performansını artırmak için başka makine öğrenimi algoritmaları veya daha karmaşık modeller de değerlendirilebilir.

# Eğitim veri seti için K-en yakın komşuluk sınıflandırıcısını uygulayınız. En iyi k değerini belirleyerek Elde ettiğiniz sonuçları raporlayınız. Test verisi için ortalama karesel hatayı hesaplayınız.

**KOD:**

from sklearn.model\_selection import GridSearchCV

from sklearn.neighbors import KNeighborsClassifier

param\_grid = {'n\_neighbors': range(1, 31)}

knn = KNeighborsClassifier()

grid\_search = GridSearchCV(knn, param\_grid, cv=10, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

# En iyi parametreleri ve en iyi skoru alın

best\_k = grid\_search.best\_params\_['n\_neighbors']

best\_score = grid\_search.best\_score\_

print(f"En iyi k değeri: {best\_k}")

print(f"Bu k değeriyle elde edilen en iyi doğruluk skoru: {best\_score}")

best\_knn = KNeighborsClassifier(n\_neighbors=best\_k)

best\_knn.fit(X\_train, y\_train)

from sklearn.metrics import mean\_squared\_error, confusion\_matrix, classification\_report

import seaborn as sns

import matplotlib.pyplot as plt

# Test verisi üzerinde tahmin yapma

y\_pred = best\_knn.predict(X\_test)

# Confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

# Sınıflandırma raporu

report = classification\_report(y\_test, y\_pred)

# Heatmap ile confusion matrix görselleştirme

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek')

plt.title('Konfuzyon Matrisi')

plt.show()

print("Sınıflandırma Raporu:")

print(report)

# Ortalama kare hata hesaplama

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Test verisi için ortalama karesel hata: {mse}")

from sklearn.metrics import roc\_curve, auc

# Tahmin olasılıklarını al

y\_pred\_proba = best\_knn.predict\_proba(X\_test)[:, 1]

# ROC eğrisi ve AUC değeri hesapla

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_proba)

roc\_auc = auc(fpr, tpr)

# ROC eğrisini çiz

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='red', 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('ROC Curve')

plt.legend(loc="lower right")

plt.show()

print("ROC AUC: ", roc\_auc)

**ÇIKTI:**

**metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

****

****

**metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM**

**1. Tutarlılık ve İyileşme:**

* Eğitim verisi ve hiperparametre optimizasyonu sonrası elde edilen sonuçlar oldukça tutarlıdır. Özellikle k değerinin 11 olarak belirlenmesi her iki durumda da aynıdır, bu da modelin stabil olduğunu gösterir.
* Hiperparametre optimizasyonu sonrası en iyi doğruluk skoru 0.7486 olarak bulunmuştur, bu skor eğitim verisindeki %73'lük doğruluk oranı ile uyumludur ve küçük bir iyileşme göstermektedir.

**2. Model Performansı:**

* Konfuzyon Matrixi (Confusion Matrix): Her iki durumda da modelin çoğunluk sınıfı (sınıf 0) daha iyi tanımladığı, azınlık sınıfında (sınıf 1) ise daha fazla hata yaptığı görülüyor. Bu durum sınıflar arasındaki dengesizlikten kaynaklanabilir.
* Precision, Recall, F1-score: Değerler arasında belirgin bir fark yoktur. Sınıf 0 için precision, recall ve f1-score değerleri oldukça yüksektir, ancak sınıf 1 için bu değerler daha düşüktür. Bu, modelin sınıf 1'i tanımakta zorlandığını gösterir.

**3. Test verisi için Ortama Karesel Hata (MSE):**

* Test verisi için ortalama karesel hata 0.2684 olarak hesaplanmıştır. MSE değeri, sınıflandırma problemlerinde doğrudan performans göstergesi olmasa da modelin hata oranı hakkında genel bir fikir verir. Bu değer oldukça düşük olup modelin iyi bir performans sergilediğini göstermektedir.

**çizgi, metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM**

**ROC AUC:** Hiperparametre optimizasyonu öncesi ve sonrası ROC AUC değeri aynıdır (0.7566), bu da modelin genel olarak aynı performansı koruduğunu gösterir.

**Genel Değerlendirme:**

Hiperparametre optimizasyonu sonrası model performansında anlamlı bir değişiklik olmamıştır. En iyi k değeri olan 11 sabit kalmıştır ve diğer metrikler de büyük ölçüde aynı kalmıştır. Bu, modelin zaten optimum performansa yakın olduğunu ve hiperparametre optimizasyonunun çok büyük bir iyileşme sağlamadığını göstermektedir.

Model performansını daha da iyileştirmek için, veri setinde sınıf dengesizliklerini ele almak (örneğin, SMOTE gibi yöntemlerle) veya model kompleksitesini artırarak (örneğin, daha karmaşık modeller kullanarak) farklı yaklaşımlar denenebilir.

# Eğitim veri seti için Multi-Layer Perceptron (MLP) ve Support Vector Machines (SVM) sınıflandırıcılarını uygulayınız. Test verisi için ortalama karesel hatayı hesaplayınız. Eğitim ve test adımlarında elde ettiğiniz sonucları raporlayınız.

**KOD:**

from sklearn.neural\_network import MLPClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.preprocessing import StandardScaler

# Özellikleri ölçeklendir (MLP ve SVM için önemli)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Çoklu Katmanlı Algılayıcı (MLP) için parametre aralıkları

param\_grid\_mlp = {

    'hidden\_layer\_sizes': [(50,), (100,), (50, 50), (100, 50)],

    'activation': ['logistic', 'relu'],

    'solver': ['sgd', 'adam'],

    'alpha': [0.0001, 0.001, 0.01],

}

# MLP sınıflandırıcısı oluştur

mlp = MLPClassifier(max\_iter=2000) # Örnek olarak 2000 iterasyon

model = MLPClassifier(hidden\_layer\_sizes=(100,)) # Örnek olarak 100 nöronlu tek bir gizli katman

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

model.fit(X\_train\_scaled, y\_train)

# GridSearchCV ile parametre optimizasyonunu yap

grid\_search\_mlp = GridSearchCV(mlp, param\_grid\_mlp, cv=5)

grid\_search\_mlp.fit(X\_train\_scaled, y\_train)

# En iyi parametreleri ve en iyi skoru göster

print("En iyi parametreler:", grid\_search\_mlp.best\_params\_)

print("En iyi skor:", grid\_search\_mlp.best\_score\_)

from sklearn.metrics import mean\_squared\_error, confusion\_matrix, classification\_report

# Test verisi tahmin edilir

y\_pred = grid\_search\_mlp.best\_estimator\_.predict(X\_test\_scaled)

# Confusion matrix çizilir

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("MLP Confusion Matrix:")

print(conf\_matrix)

# Confusion matrix grafiği çizilir

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.title('MLP Konfuzyon Matrisi')

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek')

plt.show()

# Sınıflandırma raporu oluşturulur

class\_report = classification\_report(y\_test, y\_pred)

print("MLP Sınıflandırma Raporu:")

print(class\_report)

# Ortalama karesel hata hesaplanır

mse = mean\_squared\_error(y\_test, y\_pred)

print("MLP Test verisi için Ortalama Kareli Hata:", mse)

from sklearn.metrics import roc\_curve, roc\_auc\_score

import matplotlib.pyplot as plt

# Test verisi üzerinde tahminler yapılır

y\_scores = grid\_search\_mlp.best\_estimator\_.predict\_proba(X\_test\_scaled)[:, 1]

# ROC eğrisi oluşturulur

fpr, tpr, thresholds = roc\_curve(y\_test, y\_scores)

# AUC (Alan Altındaki Kareler) hesaplanır

auc = roc\_auc\_score(y\_test, y\_scores)

# ROC eğrisi çizilir

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.2f})')

plt.plot([0, 1], [0, 1], color='red', label='Random Guessing')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('MLP ROC Curve')

plt.legend()

plt.grid(True)

plt.show()

print("MLP ROC AUC: ", auc)

from sklearn.svm import SVC

from sklearn.model\_selection import GridSearchCV

# Destek Vektör Makineleri (SVM) için parametre aralıkları

param\_grid\_svm = {

    'C': [0.1, 1, 10, 100],

    'gamma': [0.001, 0.01, 0.1, 1],

    'kernel': ['linear', 'rbf', 'poly'],

}

# SVM sınıflandırıcısı oluştur

svm = SVC()

# GridSearchCV ile parametre optimizasyonunu yap

grid\_search\_svm = GridSearchCV(svm, param\_grid\_svm, cv=5)

grid\_search\_svm.fit(X\_train\_scaled, y\_train)

# En iyi parametreleri ve en iyi skoru göster

print("En iyi parametreler:", grid\_search\_svm.best\_params\_)

print("En iyi skor:", grid\_search\_svm.best\_score\_)

from sklearn.metrics import mean\_squared\_error, confusion\_matrix, classification\_report

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

import pandas as pd

# GridSearchCV sonuçlarını kullanarak en iyi modeli alalım

best\_svm\_model = grid\_search\_svm.best\_estimator\_

# Test verisi üzerinde tahminler yapalım

y\_pred = best\_svm\_model.predict(X\_test\_scaled)

# Confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("SVM Confusion Matrix:")

print(conf\_matrix)

# Confusion matrix grafiği çizilir

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=["Negatif", "Pozitif"], yticklabels=["Negatif", "Pozitif"])

plt.title('SVM Konfuzyon Matrisi')

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek')

plt.show()

# Sınıflandırma raporu oluşturulur

class\_report = classification\_report(y\_test, y\_pred)

print("SVM Sınıflandırma Raporu:")

print(class\_report)

# Ortalama karesel hata hesaplanır

mse = mean\_squared\_error(y\_test, y\_pred)

print("SVM Test verisi için Ortalama Kareli Hata:", mse)

from sklearn.metrics import roc\_curve, roc\_auc\_score

import matplotlib.pyplot as plt

# Test verisi üzerinde tahminler yapalım

y\_scores = grid\_search\_svm.best\_estimator\_.decision\_function(X\_test\_scaled)

# ROC eğrisi oluşturulur

fpr, tpr, thresholds = roc\_curve(y\_test, y\_scores)

# AUC (Alan Altındaki Kareler) hesaplanır

auc = roc\_auc\_score(y\_test, y\_scores)

# ROC eğrisi çizilir

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.2f})')

plt.plot([0, 1], [0, 1], color='red', label='Random Guessing')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('SVM ROC Curve')

plt.legend()

plt.grid(True)

plt.show()

print("SVM ROC AUC: ", auc)

**ÇIKTI:**

****

****

metin, ekran görüntüsü, sayı, numara, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**YORUM:**

Hiperparametre optimizasyonu uyguladıktan sonra elde ettiğiniz sonuçlar, modelin performansında belirgin bir iyileşme olduğunu gösteriyor. Bu iyileşmeyi daha ayrıntılı olarak inceleyelim:

**1. Karışıklık Matrisi:**

**Öncesi:**

[[120 31]

[ 38 42]]

**Sonrası:**

[[119 32]

[ 31 49]]

* Pozitif sınıfın (1) doğru tespitlerinde (True Positive) artış var: 42'den 49'a çıktı.
* Pozitif sınıfın yanlış negatiflerinde (False Negative) azalma var: 38'den 31'e düştü.
* Negatif sınıfın (0) doğru tespitlerinde (True Negative) hafif bir azalma: 120'den 119'a düştü.
* Negatif sınıfın yanlış pozitiflerinde (False Positive) hafif bir artış: 31'den 32'ye çıktı.

**2. Sınıflandırma Raporu:** Sınıflandırma raporuna bakıldığında, model performansında belirgin iyileşmeler olduğu görülmektedir. Önceki değerlere göre, sınıf 0 için precision 0.76'dan 0.79'a, recall 0.79'dan aynı kalarak, f1-score ise 0.78'den 0.79'a yükselmiştir. Sınıf 1 için ise precision 0.58'den 0.60'a, recall 0.53'ten 0.61'e ve f1-score 0.55'ten 0.61'e çıkmıştır. Genel doğruluk (accuracy) değeri 0.70'den 0.73'e yükselmiştir. Ayrıca, macro ve weighted average f1-score değerlerinde de sırasıyla 0.66'dan 0.70'e ve 0.70'den 0.73'e artış gözlemlenmiştir. Bu artışlar, modelin genel performansında ve her iki sınıf için de daha dengeli bir iyileşme sağlandığını göstermektedir.

**3. Test verisi için Ortalama Kareli Hata (MSE):**

MSE: 0.2727272727272727 Bu değer, modelin tahmin hatalarının büyüklüğünü gösterir. Düşük bir değer, modelin tahminlerinde daha az hata yaptığını gösterir.

**Genel Değerlendirme:**

Hiperparametre optimizasyonu sonrası modelin performansında belirgin bir iyileşme gözlemlenmiştir:

* Doğruluk (accuracy), ROC AUC, ve f1-score değerlerinde artış var.
* Pozitif sınıfın doğru tespit oranı arttı ve yanlış negatifler azaldı.
* Negatif sınıfın doğru tespit oranında hafif bir azalma olsa da, genel olarak modelin performansı iyileşmiş.
* Ortalama Kareli Hata (MSE) düşük, bu da modelin tahminlerinin doğruluğunu gösterir.

Bu sonuçlar, modelinizin hiperparametre optimizasyonu sayesinde daha iyi genelleştirme yeteneğine sahip olduğunu ve veriyi daha iyi sınıflandırdığını gösteriyor. Optimizasyon işlemi, modelinizin genel performansını artırmak için oldukça etkili olmuştur.

**metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM**

**ROC AUC:**

**Öncesi:** 0.7597682119205298

**Sonrası:** 0.8000827814569537

Eğitim veri seti üzerinde uygulanan Multi-Layer Perceptron (MLP) sınıflandırıcısı başlangıçta 0.7597'lik bir ROC AUC skoruna sahipti. Hiperparametre optimizasyonu sonrası bu skor 0.8001'e yükseldi. ROC AUC'deki bu artış, modelin sınıflandırma performansının iyileştiğini ve daha doğru tahminler yapabildiğini gösteriyor. Hiperparametre optimizasyonu, modelin öğrenme oranı, katman sayısı ve nöron sayısı gibi parametrelerini ayarlayarak daha verimli ve doğru bir model elde edilmesini sağlamıştır.

Bu sonuç, modelin gerçek dünyada daha güvenilir ve doğru tahminler yapmasına olanak tanır. İleriye dönük olarak, daha geniş bir hiperparametre aralığı ile optimizasyon yapılabilir veya farklı optimizasyon teknikleri kullanılabilir.

**ÇIKTI:**

****

**metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM**

Hiperparametre optimizasyonu uyguladıktan sonra elde edilen sonuçları değerlendirirken, model performansının çeşitli metrikler üzerinde nasıl değiştiğine dikkat etmemiz gerekiyor. İşte bu sonuçların bir analizi:

**1. Karışıklık Matrisi**

**Öncesi:**

[[125 26]

[ 33 47]]

**Sonrası:**

[[123 28]

[ 30 50]]

Optimizasyon sonrası doğru sınıflandırılan pozitif örneklerin (true positives) sayısında bir artış gözlemleniyor (47'den 50'ye). Yanlış pozitiflerin (false positives) sayısında ise hafif bir artış var (26'dan 28'e). Ayrıca yanlış negatiflerin (false negatives) sayısında azalma (33'ten 30'a) olduğu görülüyor. Bu da modelin pozitif sınıfı tanımada daha iyi hale geldiğini gösteriyor.

**2. Sınıflandırma Raporu**

Optimizasyon sonrası sınıflandırma raporunda dikkat çeken noktalar şunlardır: Accuracy değerinde %1'lik bir artış gözlemlenmiştir (0.74'ten 0.75'e). Özellikle 1. sınıf için recall değerinde bir iyileşme söz konusudur; bu değer 0.59'dan 0.62'ye yükselmiş ve modelin pozitif sınıfı daha iyi tanıdığını göstermektedir. Her iki sınıf için de F1-Score değerlerinde iyileşme görülmektedir, özellikle 1. sınıfta bu değer 0.61'den 0.63'e yükselmiştir. Ayrıca, weighted ve macro average metriklerinde de genel bir iyileşme mevcuttur. Bu durum, modelin genel performansının arttığını göstermektedir.

**3. Test Verisi için Ortalama Kareli Hata (MSE)**

MSE: 0.2510822510822511

MSE değeri 0.251 olarak raporlanmış, bu da modelin tahminlerinde ortalama karesel hata olarak düşük bir hata oranına sahip olduğunu gösterir. Ancak, sınıflandırma problemlerinde MSE yaygın olarak kullanılmaz, daha çok regresyon problemleri için uygundur. Yine de, bu değerin düşük olması optimizasyonun model tahminlerinde tutarlılığı artırdığını gösterebilir.

**Genel Değerlendirme**

Hiperparametre optimizasyonu sonrası:

Modelin doğruluk oranı (%1) artmıştır.

* Modelin doğruluk oranı (%1) artmıştır.
* Pozitif sınıfı tanımadaki başarı (recall) ve genel performans (ROC AUC) anlamlı şekilde iyileşmiştir.
* F1-score ve diğer ortalama metrikler (macro ve weighted avg) de genel iyileşmeyi desteklemektedir.

Sonuç olarak, hiperparametre optimizasyonu, modelin pozitif sınıfı daha iyi tanımasını ve genel olarak daha dengeli bir performans sergilemesini sağlamıştır. Bu da eğitim veri seti üzerinde yapılan optimizasyonun başarılı olduğunu gösterir.

**metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YORUM**

**ROC AUC**

**Öncesi:** 0.7076572847682119

**Sonrası:** 0.8030629139072848

ROC AUC değerinde oldukça belirgin bir artış var (0.71'den 0.80'e). Bu, modelin pozitif sınıfı tanımadaki genel doğruluğunun önemli ölçüde arttığını gösterir.