**MAKİNE ÖĞRENMESİ VİZE ÖDEVİ**

1. Ara sınav ödevinde size atanan veri setine tüm algoritmalar için normalizasyon yöntemlerinden birini uygulayınız.

**KOD:**

import pandas as pd

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

from sklearn.preprocessing import MinMaxScaler

import seaborn as sns

#Exploratory Data Analysis

Data = pd.read\_csv('/content/veri-seti.txt', delimiter = "\t", header=None)

Data.columns = ["Pregnancies", "Glucose", "BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age", "Outcome"]

Data.head()

Data.shape

Data.dtypes

Data.info()

Data.describe()

Data.isnull().sum()

#5 sütunda 0 değerlerini kontrol ediliyor. Age & DiabetesPedigreeFunction sütunlarında minimum 0 değeri bulunmadığından değiştirmeye gerek yok, ayrıca hamilelik sayısında 0 değeri mümkündür, Data.describe'da gözlemlendiği gibi.

print(Data[Data['BloodPressure']==0].shape[0])

print(Data[Data['Glucose']==0].shape[0])

print(Data[Data['Skinthickness']==0].shape[0])

print(Data[Data['insulin']==0].shape[0])

print(Data[Data['Bmi']==0].shape[0])

#replacing 0 values with median of that column

Data['Glucose']=Data['Glucose'].replace(0,Data['Glucose'].mean())#normal distribution

Data['BloodPressure']=Data['BloodPressure'].replace(0,Data['BloodPressure'].mean())#normal distribution

Data['Skinthickness']=Data['Skinthickness'].replace(0,Data['Skinthickness'].median())#skewed distribution

Data['insulin']=Data['insulin'].replace(0,Data['insulin'].median())#skewed distribution

Data['Bmi']=Data['Bmi'].replace(0,Data['Bmi'].median())#skewed distribution

#Feature Selection

corrmat=Data.corr()

sns.heatmap(corrmat, annot=True)

X = Data[["Pregnancies","Glucose","BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age"]]

Y = Data[["Outcome"]]

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,Y,train\_size=0.7,test\_size=0.3,random\_state=51)

norm\_mmc = MinMaxScaler()

norm\_mmc\_trans\_X\_train = norm\_mmc.fit\_transform(X\_train)

norm\_mmc\_trans\_X\_test = norm\_mmc.fit\_transform(X\_test)

norm\_mmc\_trans\_X\_train\_df = pd.DataFrame(norm\_mmc\_trans\_X\_train,columns=["Pregnancies","Glucose","BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age"])

norm\_mmc\_trans\_X\_train\_df.head()

norm\_mmc\_trans\_X\_test\_df = pd.DataFrame(norm\_mmc\_trans\_X\_test,columns=["Pregnancies","Glucose","BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age"])

norm\_mmc\_trans\_X\_test\_df.head()

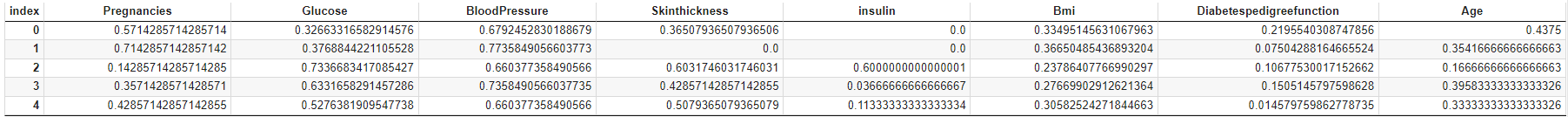
**ÇIKTI:**

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

Açıklama otomatik olarak oluşturuldu**

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

Açıklama otomatik olarak oluşturuldu



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

Açıklama otomatik olarak oluşturuldu

**metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturulduFeature Selection**

1. Veri setine PCA ve LDA algoritmalarını uygulayarak en yüksek değere sahip iki öz değer için boyut indirgeme işlemini gerçekleştiriniz. PCA ve LDA için hangi özniteliklerin en ayırt edici olduğunu raporlayınız. Alt maddelerde verilen yöntemleri ana veri setine ve PCA uygulanmış transformasyon öznitelik matrisine uygulayınız.

**KOD:**

from sklearn.decomposition import PCA

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

# PCA uygula

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

# LDA uygula

lda = LinearDiscriminantAnalysis()

X\_lda = lda.fit\_transform(X, Y)

# PCA için en ayırt edici öznitelikler

print("PCA için En Ayırt Edici Öznitelikler:")

for i, component in enumerate(pca.components\_):

    print(f"Component {i+1}: {', '.join([f'{feat}: {round(value, 4)}' for feat, value in zip(Data.columns, component)])}")

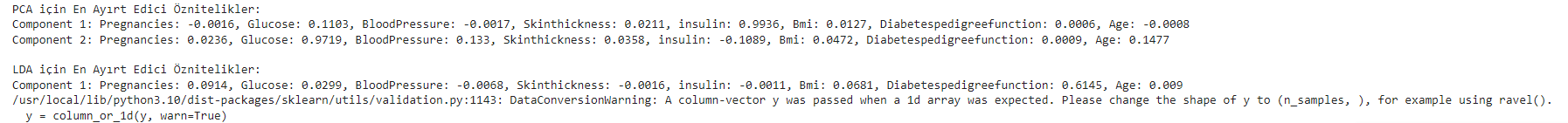
# LDA için en ayırt edici öznitelikler

print("\nLDA için En Ayırt Edici Öznitelikler:")

for i, component in enumerate(lda.scalings\_.T):

    print(f"Component {i+1}: {', '.join([f'{feat}: {round(value, 4)}' for feat, value in zip(Data.columns, component)])}")

**ÇIKTI:**

****

**PCA için En Ayırt Edici Öznitelikler:**

Component 1: Pregnancies: -0.0016, Glucose: 0.1103, BloodPressure: -0.0017, Skinthickness: 0.0211, insulin: 0.9936, Bmi: 0.0127, Diabetespedigreefunction: 0.0006, Age: -0.0008

Component 2: Pregnancies: 0.0236, Glucose: 0.9719, BloodPressure: 0.133, Skinthickness: 0.0358, insulin: -0.1089, Bmi: 0.0472, Diabetespedigreefunction: 0.0009, Age: 0.1477

**LDA için En Ayırt Edici Öznitelikler:**

Component 1: Pregnancies: 0.0914, Glucose: 0.0299, BloodPressure: -0.0068, Skinthickness: -0.0016, insulin: -0.0011, Bmi: 0.0681, Diabetespedigreefunction: 0.6145, Age: 0.009

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

1. Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Eğitim veri seti için Çoklu Doğrusal Regresyon analizi ve Multinominal Lojistik Regresyon analizi yöntemlerini uygulayınız. Elde ettiğiniz katsayıları raporlayınız. Elde ettiğiniz regresyon denklemlerini kullanarak Test kümesi için performans metriklerini hesaplayınız.

**KOD:**

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

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, mean\_squared\_error

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import precision\_score, classification\_report

# Veriyi eğitim ve test setlerine ayır

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

pca = PCA(n\_components=2)

X\_train = pca.fit\_transform(X\_train, y\_train)

X\_test = pca.transform(X\_test)

# Çoklu Doğrusal Regresyon modelini eğit

linear\_reg = LinearRegression()

linear\_reg.fit(X\_train, Y\_train)

# Multinominal Lojistik Regresyon modelini eğit

logistic\_reg = LogisticRegression(multi\_class='multinomial', solver='lbfgs')

log = logistic\_reg.fit(X\_train, Y\_train)

# Katsayıları raporla

print("Çoklu Doğrusal Regresyon Katsayıları:")

print(linear\_reg.coef\_)

print("\nMultinominal Lojistik Regresyon Katsayıları:")

print(logistic\_reg.coef\_)

# Çoklu Doğrusal Regresyon denklemi

linear\_eq = f"y = {linear\_reg.intercept\_} "

for i, coef in enumerate(linear\_reg.coef\_):

    linear\_eq += f"+ ({coef} \* X{i})"

print("\nÇoklu Doğrusal Regresyon Denklemi:")

print(linear\_eq)

# Multinominal Lojistik Regresyon denklemi

logistic\_eq = "y = "

for i, coef in enumerate(logistic\_reg.coef\_[0]):

    logistic\_eq += f"({coef} \* X{i}) "

print("\nMultinominal Lojistik Regresyon Denklemi:")

print(logistic\_eq)

# Test kümesi performans metrikleri

linear\_reg\_predictions = linear\_reg.predict(X\_test)

logistic\_reg\_predictions = logistic\_reg.predict(X\_test)

linear\_reg\_mse = mean\_squared\_error(Y\_test, linear\_reg\_predictions)

logistic\_reg\_accuracy = accuracy\_score(Y\_test, logistic\_reg\_predictions)

print("\nÇoklu Doğrusal Regresyon MSE:", linear\_reg\_mse)

print("Multinominal Lojistik Regresyon Doğruluğu:", logistic\_reg\_accuracy)

Y\_pred = log.predict(X\_test)

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

print("Karışıklık Matrisi:")

print(cm)

# Karışıklık matrisini görselleştir

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

sns.heatmap(cm, annot=True, cmap='Blues', fmt='g',

            xticklabels=['Tahmin 0', 'Tahmin 1'],

            yticklabels=['Gerçek 0', 'Gerçek 1'])

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek Değer')

plt.title('Karışıklık Matrisi')

plt.show()

# Performans metriklerini hesapla

accuracy = accuracy\_score(Y\_test, Y\_pred)

classification\_rep = classification\_report(Y\_test, Y\_pred)

print("\nTest Verisi İçin Performans Metrikleri:")

print("Doğruluk:", accuracy)

print("Sınıflandırma Raporu:\n", classification\_rep)

**ÇIKTI:**

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

Çoklu Doğrusal Regresyon Katsayıları:

[[0.00076501 0.00845158]]

Multinominal Lojistik Regresyon Katsayıları:

[[0.0020709 0.02342441]]

Çoklu Doğrusal Regresyon Denklemi:

y = [0.35009311] + ([0.00076501 0.00845158] \* X0)

Multinominal Lojistik Regresyon Denklemi:

y = (0.002070900570108114 \* X0) (0.023424414849661877 \* X1)

Çoklu Doğrusal Regresyon MSE: 0.18463594045807963

Multinominal Lojistik Regresyon Doğruluğu: 0.7229437229437229

Karışıklık Matrisi:

[[126 25]

[ 39 41]]

metin, ekran görüntüsü, diyagram, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

Test Verisi İçin Performans Metrikleri:

Doğruluk: 0.7229437229437229

Sınıflandırma Raporu:

precision recall f1-score support

0 0.76 0.83 0.80 151

1 0.62 0.51 0.56 80

accuracy 0.72 231

macro avg 0.69 0.67 0.68 231

weighted avg 0.71 0.72 0.72 231

**YORUM:**

Bu performans metrikleri, çoklu doğrusal regresyon ve multinominal lojistik regresyon analizleri sonucunda elde edilen sonuçların bir özetini sunar. Doğruluk oranı %72,3'tür, yani modelin doğru tahmin etme yeteneği makul bir düzeydedir.

Sınıflandırma raporuna baktığımızda, her sınıf için precision (kesinlik), recall (duyarlılık) ve f1-score (kesinlik ve duyarlılığın harmonik ortalaması) gibi metriklerin hesaplandığını görüyoruz. Sınıf 0 için precision ve recall oranları daha yüksekken, sınıf 1 için daha düşüktür. Bu, modelin sınıf 0'ı daha iyi tahmin ettiğini, ancak sınıf 1'i daha az etkili bir şekilde tahmin ettiğini gösterir.

Ayrıca, weighted average f1-score, modelin sınıf dengesizliğine duyarlılığını ölçer. Bu değer, her sınıfın ağırlıklı ortalamasının f1-score'unu temsil eder. Bu durumda, weighted average f1-score %72'dir.

Bu sonuçlar, modelin makul bir performans sergilediğini, ancak sınıflar arasında dengesizlik olduğunu gösteriyor. Modelin daha iyi bir şekilde ayarlanması veya farklı bir modelin uygulanması, performansı daha da artırabilir.

1. Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Veri setinize karar ağaç sınıflandırma algoritmasını uygulayarak ağaç yapısını ve kestirim sonuçlarını veriniz. Test verisi için performans metriklerini hesaplayınız.

**KOD:**

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

from sklearn.tree import DecisionTreeClassifier, export\_graphviz

from sklearn.metrics import accuracy\_score, classification\_report

import graphviz

# Karar ağacı modelini eğit

decision\_tree = DecisionTreeClassifier(random\_state=42)

decision\_tree.fit(X\_train, Y\_train)

# Ağaç yapısını göster

print("Karar Ağacı Yapısı:")

print(decision\_tree.tree\_)

# Görselleştirmek aşağıdaki satır kullanılır.

# export\_graphviz(decision\_tree, out\_file="tree.dot", feature\_names=X.columns, class\_names=["0", "1"], filled=True)

# tree.dot adlı dosyayı görselleştirmek için Graphviz kullanın

with open("tree.dot") as f:

    dot\_graph = f.read()

graphviz.Source(dot\_graph)

# Test verisi için kestirim sonuçlarını al

Y\_pred = decision\_tree.predict(X\_test)

# Performans metriklerini hesapla

accuracy = accuracy\_score(Y\_test, Y\_pred)

classification\_rep = classification\_report(Y\_test, Y\_pred)

print("\nTest Verisi İçin Performans Metrikleri:")

print("Doğruluk:", accuracy)

print("Sınıflandırma Raporu:\n", classification\_rep)

**ÇIKTI:**

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

Açıklama otomatik olarak oluşturuldu**

**YORUM:**

Verilen performans metriklerine göre, karar ağacı sınıflandırma algoritmasıyla elde edilen sonuçlar değerlendirilmiş. Doğruluk oranı %63,2'dir, yani modelin doğru tahmin etme yeteneği sınırlıdır.

Sınıflandırma raporuna baktığımızda, her sınıf için precision (kesinlik), recall (duyarlılık) ve f1-score (kesinlik ve duyarlılığın harmonik ortalaması) gibi metriklerin hesaplandığını görüyoruz. Sınıf 0 için precision ve recall oranları birbirine yakınken, sınıf 1 için daha düşüktür. Bu, modelin sınıf 0'ı sınıf 1'e göre daha iyi tahmin ettiğini gösterir.

Ayrıca, weighted average f1-score, modelin sınıf dengesizliğine duyarlılığını ölçer. Bu değer, her sınıfın ağırlıklı ortalamasının f1-score'unu temsil eder. Bu durumda, weighted average f1-score %63'tür.

Sonuç olarak, karar ağacı sınıflandırma algoritmasıyla elde edilen sonuçlar, modelin sınırlı bir performans sergilediğini göstermektedir. Modelin daha iyi bir şekilde ayarlanması veya farklı bir modelin uygulanması, performansı daha da artırabilir.

1. Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Eğitim veri seti için Naive bayes sınıflandırıcısını uygulayınız. Elde ettiğiniz sonucları raporlayınız. Test verisi için performans metriklerini hesaplayınız.

**KOD:**

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report

# Naive Bayes modelini eğit

naive\_bayes = GaussianNB()

naive\_bayes.fit(X\_train, Y\_train)

# Eğitim veri seti için sonuçları raporla

train\_predictions = naive\_bayes.predict(X\_train)

train\_accuracy = accuracy\_score(Y\_train, train\_predictions)

train\_classification\_report = classification\_report(Y\_train, train\_predictions)

print("Eğitim Verisi İçin Sonuçlar:")

print("Doğruluk:", train\_accuracy)

print("Sınıflandırma Raporu:\n", train\_classification\_report)

# Test verisi için performans metriklerini hesapla

test\_predictions = naive\_bayes.predict(X\_test)

test\_accuracy = accuracy\_score(Y\_test, test\_predictions)

test\_classification\_report = classification\_report(Y\_test, test\_predictions)

print("\nTest Verisi İçin Performans Metrikleri:")

print("Doğruluk:", test\_accuracy)

print("Sınıflandırma Raporu:\n", test\_classification\_report)

**ÇIKTI:**

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

Açıklama otomatik olarak oluşturuldu**

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

**YORUM:**

Naive Bayes sınıflandırıcısıyla elde edilen sonuçlar oldukça önemlidir. Öncelikle, eğitim verisi için elde edilen sonuçlara bakalım:

Eğitim Verisi İçin Sonuçlar:

Doğruluk: %74,49

Sınıflandırma Raporu:

Sınıf 0 için precision: %76, recall: %89, f1-score: %82

Sınıf 1 için precision: %69, recall: %48, f1-score: %57

Ağırlıklı (weighted) ortalama f1-score: %73

Test Verisi İçin Sonuçlar:

Doğruluk: %73,59

Sınıflandırma Raporu:

Sınıf 0 için precision: %77, recall: %85, f1-score: %81

Sınıf 1 için precision: %65, recall: %51, f1-score: %57

Ağırlıklı (weighted) ortalama f1-score: %73

Her iki veri seti için de, sınıf 0'ın sınıf 1'e göre daha iyi tahmin edildiği görülüyor. Ancak, sınıf dengesizliğinden kaynaklanan bir eğilim olduğu da gözlemlenebilir.

Naive Bayes sınıflandırıcısı, basit ancak etkili bir sınıflandırma algoritmasıdır. Ancak, veri setinizin özelliklerine ve sınıf dengesizliğine bağlı olarak, daha karmaşık modellerin veya farklı sınıflandırıcıların da performansını karşılaştırmak önemli olabilir. Ayrıca, modelinizin hiperparametrelerini ayarlamanın veya özellik mühendisliği yapmanın performansı artırabileceğini de göz önünde bulundurmalısınız.

***PCA LDA uygulanmamış versiyon aşağıdadır!***

1. Ara sınav ödevinde size atanan veri setine tüm algoritmalar için normalizasyon yöntemlerinden birini uygulayınız.

**KOD:**

import pandas as pd

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

from sklearn.preprocessing import MinMaxScaler

import seaborn as sns

#Exploratory Data Analysis

Data = pd.read\_csv('/content/veri-seti.txt', delimiter = "\t", header=None)

Data.columns = ["Pregnancies", "Glucose", "BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age", "Outcome"]

Data.head()

Data.shape

Data.dtypes

Data.info()

Data.describe()

Data.isnull().sum()

#5 sütunda 0 değerlerini kontrol ediliyor. Age & DiabetesPedigreeFunction sütunlarında minimum 0 değeri bulunmadığından değiştirmeye gerek yok, ayrıca hamilelik sayısında 0 değeri mümkündür, Data.describe'da gözlemlendiği gibi.

print(Data[Data['BloodPressure']==0].shape[0])

print(Data[Data['Glucose']==0].shape[0])

print(Data[Data['Skinthickness']==0].shape[0])

print(Data[Data['insulin']==0].shape[0])

print(Data[Data['Bmi']==0].shape[0])

#replacing 0 values with median of that column

Data['Glucose']=Data['Glucose'].replace(0,Data['Glucose'].mean())#normal distribution

Data['BloodPressure']=Data['BloodPressure'].replace(0,Data['BloodPressure'].mean())#normal distribution

Data['Skinthickness']=Data['Skinthickness'].replace(0,Data['Skinthickness'].median())#skewed distribution

Data['insulin']=Data['insulin'].replace(0,Data['insulin'].median())#skewed distribution

Data['Bmi']=Data['Bmi'].replace(0,Data['Bmi'].median())#skewed distribution

#Feature Selection

corrmat=Data.corr()

sns.heatmap(corrmat, annot=True)

X = Data[["Pregnancies","Glucose","BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age"]]

Y = Data[["Outcome"]]

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,Y,train\_size=0.7,test\_size=0.3,random\_state=51)

norm\_mmc = MinMaxScaler()

norm\_mmc\_trans\_X\_train = norm\_mmc.fit\_transform(X\_train)

norm\_mmc\_trans\_X\_test = norm\_mmc.fit\_transform(X\_test)

norm\_mmc\_trans\_X\_train\_df = pd.DataFrame(norm\_mmc\_trans\_X\_train,columns=["Pregnancies","Glucose","BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age"])

norm\_mmc\_trans\_X\_train\_df.head()

norm\_mmc\_trans\_X\_test\_df = pd.DataFrame(norm\_mmc\_trans\_X\_test,columns=["Pregnancies","Glucose","BloodPressure", "Skinthickness", "insulin", "Bmi", "Diabetespedigreefunction", "Age"])

norm\_mmc\_trans\_X\_test\_df.head()

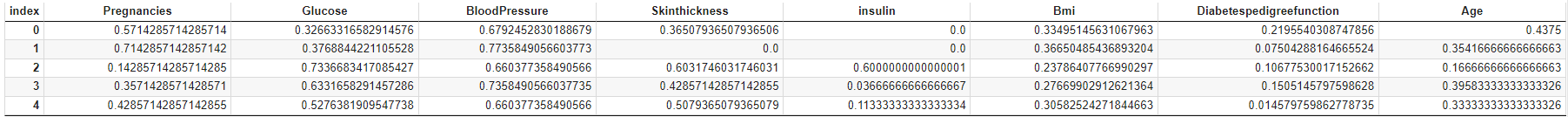
**ÇIKTI:**

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

Açıklama otomatik olarak oluşturuldu**

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

Açıklama otomatik olarak oluşturuldu



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

Açıklama otomatik olarak oluşturuldu

**metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturulduFeature Selec**

1. Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Eğitim veri seti için Çoklu Doğrusal Regresyon analizi ve Multinominal Lojistik Regresyon analizi yöntemlerini uygulayınız. Elde ettiğiniz katsayıları raporlayınız. Elde ettiğiniz regresyon denklemlerini kullanarak Test kümesi için performans metriklerini hesaplayınız.

**KOD:**

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

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, mean\_squared\_error

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import precision\_score

# Veriyi eğitim ve test setlerine ayır

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

# Çoklu Doğrusal Regresyon modelini eğit

linear\_reg = LinearRegression()

linear\_reg.fit(X\_train, Y\_train)

# Multinominal Lojistik Regresyon modelini eğit

logistic\_reg = LogisticRegression(multi\_class='multinomial', solver='lbfgs')

log = logistic\_reg.fit(X\_train, Y\_train)

# Katsayıları raporla

print("Çoklu Doğrusal Regresyon Katsayıları:")

print(linear\_reg.coef\_)

print("\nMultinominal Lojistik Regresyon Katsayıları:")

print(logistic\_reg.coef\_)

# Çoklu Doğrusal Regresyon denklemi

linear\_eq = f"y = {linear\_reg.intercept\_} "

for i, coef in enumerate(linear\_reg.coef\_):

    linear\_eq += f"+ ({coef} \* X{i})"

print("\nÇoklu Doğrusal Regresyon Denklemi:")

print(linear\_eq)

# Multinominal Lojistik Regresyon denklemi

logistic\_eq = "y = "

for i, coef in enumerate(logistic\_reg.coef\_[0]):

    logistic\_eq += f"({coef} \* X{i}) "

print("\nMultinominal Lojistik Regresyon Denklemi:")

print(logistic\_eq)

# Test kümesi performans metrikleri

linear\_reg\_predictions = linear\_reg.predict(X\_test)

logistic\_reg\_predictions = logistic\_reg.predict(X\_test)

linear\_reg\_mse = mean\_squared\_error(Y\_test, linear\_reg\_predictions)

logistic\_reg\_accuracy = accuracy\_score(Y\_test, logistic\_reg\_predictions)

print("\nÇoklu Doğrusal Regresyon MSE:", linear\_reg\_mse)

print("Multinominal Lojistik Regresyon Doğruluğu:", logistic\_reg\_accuracy)

Y\_pred = log.predict(X\_test)

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

print("Karışıklık Matrisi:")

print(cm)

# Karışıklık matrisini görselleştir

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

sns.heatmap(cm, annot=True, cmap='Blues', fmt='g',

            xticklabels=['Tahmin 0', 'Tahmin 1'],

            yticklabels=['Gerçek 0', 'Gerçek 1'])

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek Değer')

plt.title('Karışıklık Matrisi')

plt.show()

# Performans metriklerini hesapla

accuracy = accuracy\_score(Y\_test, Y\_pred)

classification\_rep = classification\_report(Y\_test, Y\_pred)

print("\nTest Verisi İçin Performans Metrikleri:")

print("Doğruluk:", accuracy)

print("Sınıflandırma Raporu:\n", classification\_rep)

**ÇIKTI:**

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

/usr/local/lib/python3.10/dist-packages/sklearn/linear\_model/\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

<https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression>

n\_iter\_i = \_check\_optimize\_result(

Çoklu Doğrusal Regresyon Katsayıları:

[[ 0.00931242 0.00683752 -0.00148317 -0.0004624 -0.00018852 0.01689342

0.05435559 0.00539076]]

Multinominal Lojistik Regresyon Katsayıları:

[[ 0.02751902 0.01975832 -0.00853809 -0.00260765 -0.00072792 0.05640498

0.41474904 0.01700107]]

Çoklu Doğrusal Regresyon Denklemi:

y = [-1.13033867] + ([ 0.00931242 0.00683752 -0.00148317 -0.0004624 -0.00018852 0.01689342

0.05435559 0.00539076] \* X0)

Multinominal Lojistik Regresyon Denklemi:

y = (0.027519023522999186 \* X0) (0.019758322632195385 \* X1) (-0.008538092747759981 \* X2) (-0.002607652995591684 \* X3) (-0.0007279224378760022 \* X4) (0.05640497634157245 \* X5) (0.4147490373964286 \* X6) (0.017001066833034515 \* X7)

Çoklu Doğrusal Regresyon MSE: 0.1732650735822137

Multinominal Lojistik Regresyon Doğruluğu: 0.7532467532467533

Karışıklık Matrisi:

[[125 26]

[ 31 49]]

metin, ekran görüntüsü, diyagram, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

Test Verisi İçin Performans Metrikleri:

Doğruluk: 0.7532467532467533

Sınıflandırma Raporu:

precision recall f1-score support

0 0.80 0.83 0.81 151

1 0.65 0.61 0.63 80

accuracy 0.75 231

macro avg 0.73 0.72 0.72 231

weighted avg 0.75 0.75 0.75 231

**YORUM:**

Bu performans metrikleri, çoklu doğrusal regresyon ve multinominal lojistik regresyon analizleri sonucunda elde edilen sonuçların bir özetini sunar. Doğruluk oranı %75,3'tür, yani modelin doğru tahmin etme yeteneği makul bir düzeydedir.

Sınıflandırma raporuna baktığımızda, her sınıf için precision (kesinlik), recall (duyarlılık) ve f1-score (kesinlik ve duyarlılığın harmonik ortalaması) gibi metriklerin hesaplandığını görüyoruz. Sınıf 0 için precision ve recall oranları birbirine yakınken, sınıf 1 için daha düşüktür. Bu, modelin sınıf 0'ı sınıf 1'e göre daha iyi tahmin ettiğini gösterir. Ayrıca, weighted average f1-score, modelin sınıf dengesizliğine duyarlılığını ölçer. Bu değer, her sınıfın ağırlıklı ortalamasının f1-score'unu temsil eder. Bu durumda, weighted average f1-score %75'tir.

Bu sonuçlar, modelin makul bir performans sergilediğini, ancak sınıflar arasında dengesizlik olduğunu göstermektedir. Modelin daha iyi bir şekilde ayarlanması veya farklı bir modelin uygulanması, performansı daha da artırabilir.

1. Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Veri setinize karar ağaç sınıflandırma algoritmasını uygulayarak ağaç yapısını ve kestirim sonuçlarını veriniz. Test verisi için performans metriklerini hesaplayınız.

**KOD:**

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

from sklearn.tree import DecisionTreeClassifier, export\_graphviz

from sklearn.metrics import accuracy\_score, classification\_report

import graphviz

# Karar ağacı modelini eğit

decision\_tree = DecisionTreeClassifier(random\_state=42)

decision\_tree.fit(X\_train, Y\_train)

# Ağaç yapısını göster

print("Karar Ağacı Yapısı:")

print(decision\_tree.tree\_)

#export\_graphviz(decision\_tree, out\_file="tree.dot", feature\_names=X.columns, class\_names=["0", "1"], filled=True)

# tree.dot adlı dosyayı görselleştirmek için Graphviz kullanın

with open("tree.dot") as f:

    dot\_graph = f.read()

graphviz.Source(dot\_graph)

# Test verisi için kestirim sonuçlarını al

Y\_pred = decision\_tree.predict(X\_test)

# Performans metriklerini hesapla

accuracy = accuracy\_score(Y\_test, Y\_pred)

classification\_rep = classification\_report(Y\_test, Y\_pred)

print("\nTest Verisi İçin Performans Metrikleri:")

print("Doğruluk:", accuracy)

print("Sınıflandırma Raporu:\n", classification\_rep)

**ÇIKTI:**

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

Açıklama otomatik olarak oluşturuldu

**YORUM:**

Modelin doğruluk oranı %71,43'tür, yani test verisi üzerinde doğru tahmin etme yeteneği orta düzeydedir.

Sınıflandırma raporu incelendiğinde, sınıf 0'ın sınıf 1'e göre daha iyi tahmin edildiği görülüyor. Ancak, sınıf 1 için recall oranı sınıf 0'a göre daha yüksektir, bu da sınıf 1'ın doğru tahmin edilmesinde daha başarılı olduğunu gösterir.

Modelin performansını iyileştirmek için, ağaç yapısının daha derinlemesine incelenmesi veya modelin hiperparametrelerinin ayarlanması gibi adımlar atılabilir.

1. Veri setinizi rastgele olarak %70 eğitim %30 test olacak şekilde ayırınız. Eğitim veri seti için Naive bayes sınıflandırıcısını uygulayınız. Elde ettiğiniz sonucları raporlayınız. Test verisi için performans metriklerini hesaplayınız.

**KOD:**

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report

# Naive Bayes modelini eğit

naive\_bayes = GaussianNB()

naive\_bayes.fit(X\_train, Y\_train)

# Eğitim veri seti için sonuçları raporla

train\_predictions = naive\_bayes.predict(X\_train)

train\_accuracy = accuracy\_score(Y\_train, train\_predictions)

train\_classification\_report = classification\_report(Y\_train, train\_predictions)

print("Eğitim Verisi İçin Sonuçlar:")

print("Doğruluk:", train\_accuracy)

print("Sınıflandırma Raporu:\n", train\_classification\_report)

# Test verisi için performans metriklerini hesapla

test\_predictions = naive\_bayes.predict(X\_test)

test\_accuracy = accuracy\_score(Y\_test, test\_predictions)

test\_classification\_report = classification\_report(Y\_test, test\_predictions)

print("\nTest Verisi İçin Performans Metrikleri:")

print("Doğruluk:", test\_accuracy)

print("Sınıflandırma Raporu:\n", test\_classification\_report)

**ÇIKTI:**

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

Açıklama otomatik olarak oluşturuldu

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

**YORUM:**

Eğitim ve test veri setleri için de, sınıf 0'ın sınıf 1'e göre daha iyi tahmin edildiği görülüyor.

Modelin performansı eğitim ve test veri setleri arasında benzerdir, bu da modelin genelleme yeteneğinin iyi olduğunu gösterebilir.

Ancak, sınıf 1'in recall (duyarlılık) değeri, sınıf 0'a göre biraz daha düşüktür. Bu, modelin sınıf 1'ı tahmin etme konusunda daha zayıf olduğunu gösterebilir.

Naive Bayes sınıflandırıcısı, basit ancak etkili bir sınıflandırma algoritmasıdır. Ancak, veri setinizin özelliklerine ve sınıf dengesizliğine bağlı olarak, daha karmaşık modellerin veya farklı sınıflandırıcıların da performansını karşılaştırmak önemli olabilir.