A logo with a tower in the center

Description automatically generated

**ÜSKÜDAR ÜNİVERSİTESİ**

Fen Bilimleri Enstitisü

**2023-2024 Bahar Dönemi**

Makine Öğrenmesi

(YZM506/1)

Final Ödevi

**Danışman Öğretim Elemanı**

  Dr. Öğr. Üyesi GÖKALP TULUM

**Öğrenci Adı-Soyadı:** Emre AKBULUT

**Öğrenci Numarası :** 234329051

**Öğrenci Bölüm Adı:** Yapay Zeka Mühendisliği(Yüksek Lisans)

**MAKİNE ÖĞRENMESİ FİNAL ÖDEVİ SORULARININ RAPORLAMASI**

**1. ve 2. Sorular:**

Öncelikle elimizde bulunan bu veri setini analiz etmek ve görselleştirmek için gerekli adımları detaylı olarak açıklayalım. Bu süreçte veri setini nasıl yükleyebileceğinizi, veriyi nasıl keşfedeceğimizi, ön işlemler yaparak nasıl analiz edebileceğinizi ve çeşitli grafikler ve bağlantılar (correlations) çizebileceğimiz gösterilmektedir.

**1. Veri Setinin Yüklenmesi ve İncelenmesi**

İlk olarak, veri setini yükleyip temel istatistiksel incelemeler yapalım.

* **Gerekli Kütüphanelerin Kurulumu ve İçe Aktarılması**

Import işlemleri gerçekleştirilmiştir.

* **Veri Setinin Yüklenmesi**

Bu aşamada veri setimiz yüklenmiştir, aşağıda verin ilk 5 satırına göz atıyoruz.

A screenshot of a graph

Description automatically generated

**2. Veri Setinin Keşfi**

Veri setinin genel yapısını ve istatistiklerini inceleyelim.

A screenshot of a computer

Description automatically generated

**3. Veri Görselleştirme**

**Histogram ve Boxplot**

Veri dağılımlarını ve aykırı değerleri incelemek için histogram ve boxplot çizelim.

* Histogram

**A group of blue and white graphs

Description automatically generated**

* Boxplot

A collage of a graph

Description automatically generated

**Korelasyon Matrisi**

Değişkenler arasındaki ilişkiyi incelemek için korelasyon matrisi ve heatmap çizelim.

A diagram of a number of data

Description automatically generated with medium confidence

**4. Verinin Ön İşlemesi**

Eksik verileri ve outlier'ları ele alalım, veriyi normalize edelim.

**Eksik Veriler İnceleme ve Doldurma**

A white background with black dots

Description automatically generated

**Outlier'ları Belirleme**

Outlier'ları ele almak önemli bir adımdır, çünkü outlier'lar model performansını olumsuz yönde etkileyebilir. Pima Indians Diabetes Dataset'inde outlier'ları belirlemek ve ele almak için birkaç yöntem kullanabiliriz. Yaygın olarak kullanılan yöntemlerden bazıları z-skoru, IQR (interquartile range) yöntemi ve görsel incelemedir.

Öncelikle, outlier'ları belirlemek için z-skoru yöntemini kullanabiliriz. Z-skoru, bir verinin ortalamadan kaç standart sapma uzakta olduğunu gösterir. Genellikle, z-skoru 3'ten büyük veya -3'ten küçük olan veriler outlier olarak kabul edilir.

A white background with black numbers

Description automatically generated with medium confidence

**Verinin Normalize Edilmesi**

A white background with black numbers

Description automatically generated

**5. Veri Setinin Bölünmesi**

Veriyi eğitim ve test setlerine bölelim.

# Kullandığım kod parçacığı aşağıdaki gibidir.

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

**6. Model Eğitimi ve Değerlendirme**

Naive Bayes modelini eğitip performansını değerlendirelim.

A blue squares with white text

Description automatically generated

A graph of a function

Description automatically generated with medium confidence

* **Accuracy:** Modelin genel doğruluğu.
* **Sensitivity (Recall):** Modelin pozitif sınıfları doğru tahmin etme oranı.
* **Specificity:** Modelin negatif sınıfları doğru tahmin etme oranı.
* **Precision:** Modelin pozitif tahminlerinin doğruluk oranı.
* **F1-Score:** Precision ve Recall'ın harmonik ortalaması.
* **ROC Curve:** Farklı eşik değerlerinde modelin performansını gösterir.
* **AUC:** ROC curve'nin altında kalan alan, modelin genel performansını özetler.

**3. Soru:**

**1. Veriyi Bölme**

Öncelikle, veri setini %70 eğitim ve %30 test olacak şekilde rastgele bölelim.

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

**2. KNN Modeli Eğitme ve En İyi k Değerini Belirleme**

KNN modelini çeşitli k değerleri için eğitip performansını değerlendireceğiz. En iyi k değerini belirlemek için cross-validation kullanıyorum.

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import cross\_val\_score

# Farklı k değerleri için KNN modelini değerlendirme

k\_values = range(1, 21)

cv\_scores = []

for k in k\_values:

knn = KNeighborsClassifier(n\_neighbors=k)

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

cv\_scores.append(scores.mean())

# En iyi k değerini belirleme

best\_k = k\_values[cv\_scores.index(max(cv\_scores))]

print(f"En iyi k değeri: {best\_k}")

**3. En İyi k Değeri ile KNN Modeli Eğitme**

En iyi k değeri ile KNN modelini eğitip test setinde tahmin yapacağız.

# En iyi k değeri ile KNN modelini eğitme

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

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

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

**4. Performans Metriklerinin Hesaplanması ve Görselleştirilmesi**

Confusion matrix, accuracy, sensitivity, specificity, precision, f1-score ve ROC curve gibi metrikleri hesaplayıp görselleştireceğiz.

**Confusion Matrix Görselleştirme**

import seaborn as sns

import matplotlib.pyplot as plt

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

# Confusion matrix hesaplama

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

# Confusion matrix'i görselleştirme

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,

xticklabels=['Non-Diabetic', 'Diabetic'],

yticklabels=['Non-Diabetic', 'Diabetic'])

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

**Performans Metriklerinin Hesaplanması**

# Confusion matrix bileşenlerini alma

TN, FP, FN, TP = conf\_matrix.ravel()

# Performans metriklerini hesaplama

accuracy = (TP + TN) / (TP + TN + FP + FN)

sensitivity = TP / (TP + FN)

specificity = TN / (TN + FP)

precision = TP / (TP + FP)

f1 = 2 \* (precision \* sensitivity) / (precision + sensitivity)

print(f"Accuracy: {accuracy:.2f}")

print(f"Sensitivity (Recall): {sensitivity:.2f}")

print(f"Specificity: {specificity:.2f}")

print(f"Precision: {precision:.2f}")

print(f"F1-Score: {f1:.2f}")

**ROC Curve ve AUC**

# ROC curve ve AUC

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

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

roc\_auc = auc(fpr, tpr)

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='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()

**Tam Süreç**

Tam süreci bir araya getirirlerek ve adım adım KNN modelini eğitip performansını değerlendirebiliriz.

Tam Süreç dosyaların içerisinde bulunan ML\_FinalOdev.ipynb dosyasında bulunmaktadır.

A blue squares with white text

Description automatically generated

A graph with a line and a blue dotted line

Description automatically generated

**4. Soru:**

Bu soruda bazı hatalar ile karşılaştım ve kodlarımı optimize etmek için farklı yöntemler kullandım. Bu denemelerden bazılarını aşağıda belirtmiş bulunmaktayım.

**1. Gizli Katman Sayısını ve Düğüm Sayısını Ayarlamak**

Modelin karmaşıklığını artırarak veya azaltarak performansını iyileştirebiliriz. Örneğin, daha fazla gizli katman veya daha fazla nöron kullanabiliriz.

mlp = MLPClassifier(hidden\_layer\_sizes=(100, 50), max\_iter=1000, learning\_rate\_init=0.001, random\_state=42, momentum=0.9)

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

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

**2. Öğrenme Oranını ve Momentum Değerini Ayarlamak**

Öğrenme oranını ve momentum değerini değiştirerek modelin daha hızlı ve kararlı bir şekilde öğrenmesini sağlayabiliriz.

mlp = MLPClassifier(hidden\_layer\_sizes=(100,), max\_iter=1000, learning\_rate\_init=0.0005, random\_state=42, momentum=0.95)

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

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

**3. Erken Durdurma Kullanmak**

Erken durdurma, modelin aşırı uyumlanmasını engelleyebilir ve daha iyi bir performans elde etmemize yardımcı olabilir.

mlp = MLPClassifier(hidden\_layer\_sizes=(100,), max\_iter=1000, learning\_rate\_init=0.001, random\_state=42, momentum=0.9, early\_stopping=True)

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

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

**4. Eğitim ve Test Verilerini Tekrar Kontrol Etmek**

Veri setinizdeki sorunları tespit etmek için verilerinizi tekrar kontrol edin. Outlier'ları ve eksik verileri temizlediğinizden emin olun.

# Outlier'ları belirleme ve kaldırma

z\_scores = np.abs(stats.zscore(df.drop('Outcome', axis=1)))

df\_no\_outliers = df[(z\_scores < 3).all(axis=1)]

# Veriyi normalize etme

scaler = StandardScaler()

X = df\_no\_outliers.drop('Outcome', axis=1)

y = df\_no\_outliers['Outcome']

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

# Veriyi yeniden eğitim ve test setlerine bölme

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

**Gibi bazı farklı işlemler uyguladım**.

**5. ve 6. Soru :**

# 5.soruda “Size atanan veri setine tüm algoritmalar için optimizasyon uygulayınız” olarak belirtildiği için tekrardan veri optimizasyon işlemleri uyguluyorum

# Veri setinin yüklenmesi

url = "veri-seti.txt"

column\_names = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

df = pd.read\_csv(url, header=None, names=column\_names)

# Eksik verileri kontrol etme

print((df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] == 0).sum())

# Sıfır değerleri ortalama ile doldurma

df['Glucose'].replace(0, df['Glucose'].mean(), inplace=True)

df['BloodPressure'].replace(0, df['BloodPressure'].mean(), inplace=True)

df['SkinThickness'].replace(0, df['SkinThickness'].mean(), inplace=True)

df['Insulin'].replace(0, df['Insulin'].mean(), inplace=True)

df['BMI'].replace(0, df['BMI'].mean(), inplace=True)

# Outlier'ları belirleme (z-skoru yöntemi)

z\_scores = np.abs(stats.zscore(df.drop('Outcome', axis=1)))

df\_no\_outliers = df[(z\_scores < 3).all(axis=1)]

# Veriyi normalize etme

scaler = StandardScaler()

X = df\_no\_outliers.drop('Outcome', axis=1)

y = df\_no\_outliers['Outcome']

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

**6. soru rapor:**

A screenshot of a graph

Description automatically generated

A graph of a curve

Description automatically generated with medium confidence



**Sonuçların Yorumlanması**

**Naive Bayes Modeli Sonuçları:**

* **Accuracy**: Modelin doğru tahmin etme oranıdır.
* **Sensitivity (Recall)**: Gerçek pozitiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Specificity**: Gerçek negatiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Precision**: Pozitif tahminlerin ne kadarının doğru olduğunu gösterir.
* **F1-Score**: Precision ve recall'ın harmonik ortalamasıdır.
* **ROC AUC**: ROC eğrisinin altında kalan alandır ve modelin sınıflandırma performansını özetler.
* **Mean Squared Error (MSE)**: Tahmin edilen ve gerçek değerler arasındaki ortalama karesel hatadır.

Bu metrikler, modelin performansını değerlendirmenize yardımcı olacaktır. Naive Bayes modeli, özellikle doğruluk ve F1 skoru gibi metrikler açısından oldukça iyi performans göstermektedir. Ortalama karesel hata, sınıflandırma hatalarını özetler ve bu hata ne kadar düşükse model o kadar iyi performans göstermiş olur.

**7. Soru :**

A blue squares with white text

Description automatically generated

A graph with blue and orange lines

Description automatically generatedA number and numbers on a white background

Description automatically generated



**Sonuçların Yorumlanması**

**KNN Modeli Sonuçları:**

* **Accuracy**: Modelin doğru tahmin etme oranıdır.
* **Sensitivity (Recall)**: Gerçek pozitiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Specificity**: Gerçek negatiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Precision**: Pozitif tahminlerin ne kadarının doğru olduğunu gösterir.
* **F1-Score**: Precision ve recall'ın harmonik ortalamasıdır.
* **ROC AUC**: ROC eğrisinin altında kalan alandır ve modelin sınıflandırma performansını özetler.
* **Mean Squared Error (MSE)**: Tahmin edilen ve gerçek değerler arasındaki ortalama karesel hatadır.

Bu metrikler, modelin performansını değerlendirmenize yardımcı olacaktır. KNN modeli, özellikle doğruluk ve F1 skoru gibi metrikler açısından oldukça iyi performans göstermektedir. Ortalama karesel hata, sınıflandırma hatalarını özetler ve bu hata ne kadar düşükse model o kadar iyi performans göstermiş olur.

**8. Soru :**

Öncelikle veri setini rastgele olarak %70 eğitim ve %30 test olacak şekilde ayıralım. Ardından, Multi-Layer Perceptron (MLP) ve Support Vector Machines (SVM) sınıflandırıcılarını uygulayarak eğitim ve test adımlarında elde edilen sonuçları raporlayalım.

**Grafikler ve diğer kodlar MFinalOdev. İpynb dosyasında bulunmaktadır.**

**Sonuçların Yorumlanması**

**MLP Modeli Sonuçları:**

* **Accuracy**: Modelin doğru tahmin etme oranıdır.
* **Sensitivity (Recall)**: Gerçek pozitiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Specificity**: Gerçek negatiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Precision**: Pozitif tahminlerin ne kadarının doğru olduğunu gösterir.
* **F1-Score**: Precision ve recall'ın harmonik ortalamasıdır.
* **ROC AUC**: ROC eğrisinin altında kalan alandır ve modelin sınıflandırma performansını özetler.

**SVM Modeli Sonuçları:**

* **Accuracy**: Modelin doğru tahmin etme oranıdır.
* **Sensitivity (Recall)**: Gerçek pozitiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Specificity**: Gerçek negatiflerin ne kadarını doğru tahmin ettiğimizi gösterir.
* **Precision**: Pozitif tahminlerin ne kadarının doğru olduğunu gösterir.
* **F1-Score**: Precision ve recall'ın harmonik ortalamasıdır.
* **ROC AUC**: ROC eğrisinin altında kalan alandır ve modelin sınıflandırma performansını özetler.

Bu metrikler, modelin performansını değerlendirmenize yardımcı olacaktır. Hem MLP hem de SVM modelleri, özellikle doğruluk ve F1 skoru gibi metrikler açısından oldukça iyi performans göstermektedir. Ortalama karesel hata, sınıflandırma hatalarını özetler ve bu hata ne kadar düşükse model o kadar iyi performans göstermiş olur.

**SONUÇ**

Aşağıdaki kod ile performans metriklerini bir tabloya yerleştirip tek bir tabloda izleme imkanımız olacaktır.

data = {

'Model': ['MLP', 'SVM', 'KNN', 'NBayes'],

'Accuracy': [accuracy\_mlp, accuracy\_svm, accuracy\_knn, accuracy\_nb],

'Recall': [sensitivity\_mlp, sensitivity\_svm, sensitivity\_knn, sensitivity\_nb],

'Specf.': [specificity\_mlp, specificity\_svm, specificity\_knn, specificity\_nb],

'Prec.': [precision\_mlp, precision\_svm, precision\_knn, precision\_nb],

'F1-Score': [f1\_mlp, f1\_svm, f1\_knn, f1\_nb],

'ROC-AUC': [roc\_auc\_mlp, roc\_auc\_svm, roc\_auc\_knn, roc\_auc\_nb]

}

df\_results = pd.DataFrame(data)

print(df\_results)

**Sonuç Tablosu**

A number and numbers on a white background

Description automatically generated

MLP aslında çok başarılı sonuçlar verebilmektedir bazı değer ile oynayarak örnek: gizli katman sayısı değiştirilerek ya da learning rate oranı ile oynayarak veya farklı optimizasyon teknikleri kullanarak sonucu farklı depğerlerde görebilmemiz mümkündür.

Emre AKBULUT 05.06.2024

Üsküdar – İSTANBUL Saygılar