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.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

2. Veri Setinin Keşfi

Veri setinin genel yapısını ve istatistiklerini inceleyelim.

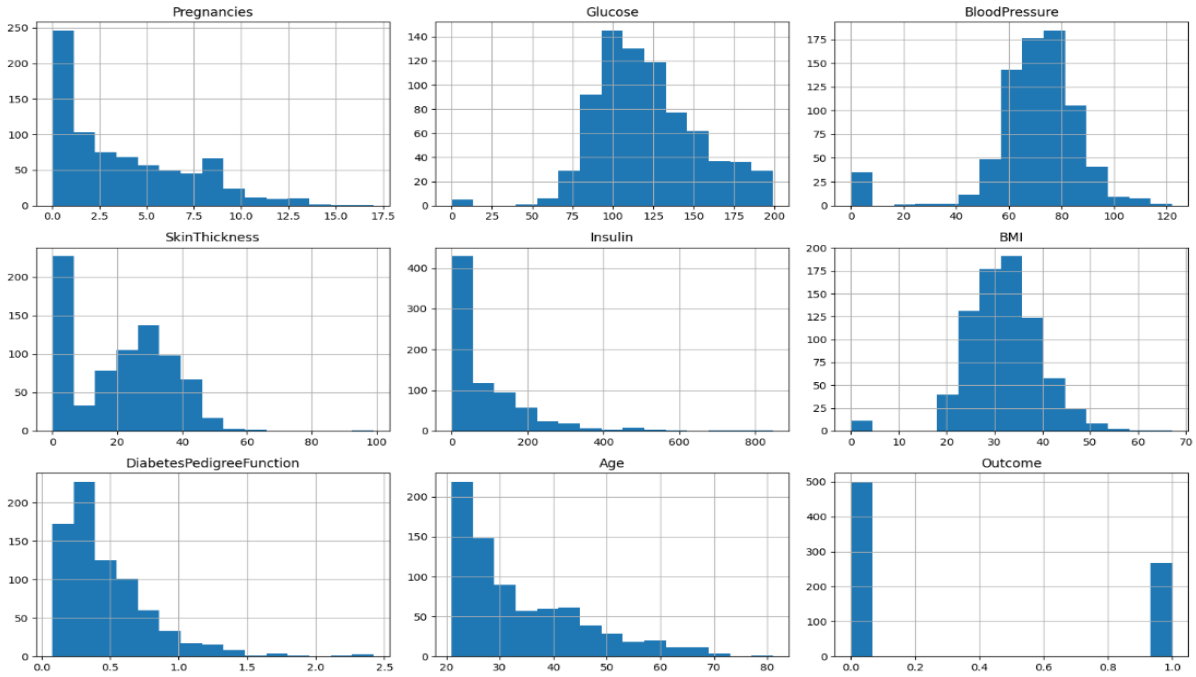
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
Outcome
0    500
1    268
Name: count, dtype: int64
```

3. Veri Görselleştirme

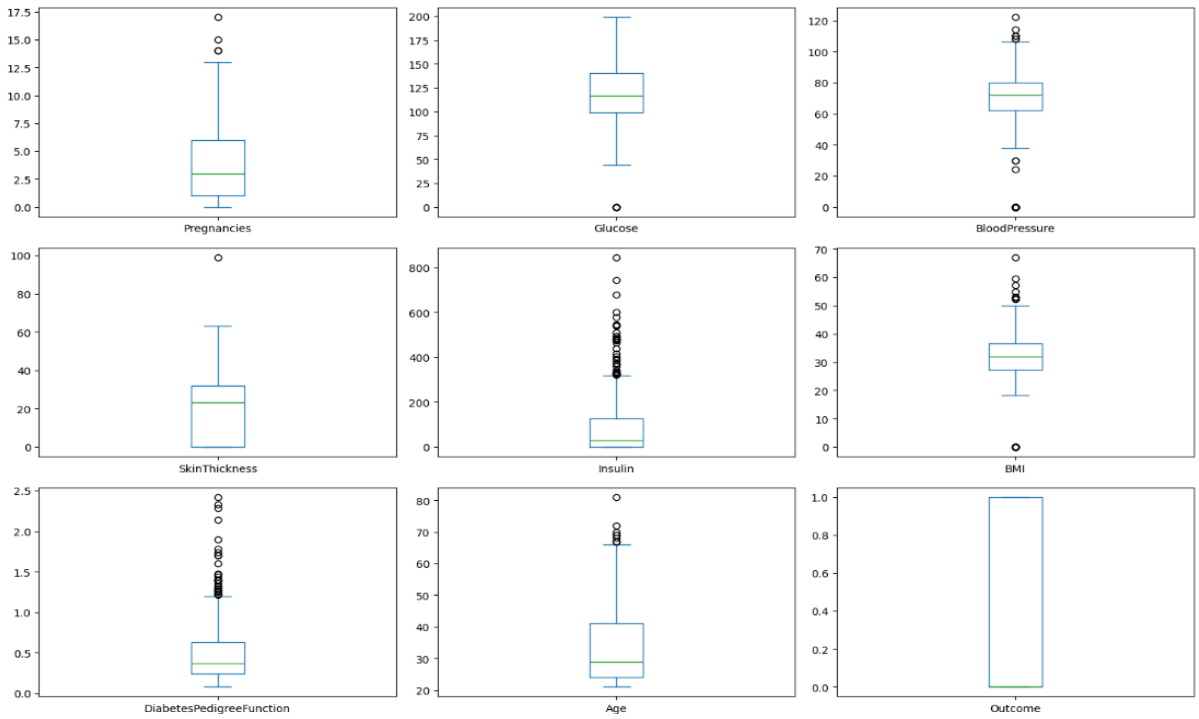
Histogram ve Boxplot

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

- Histogram

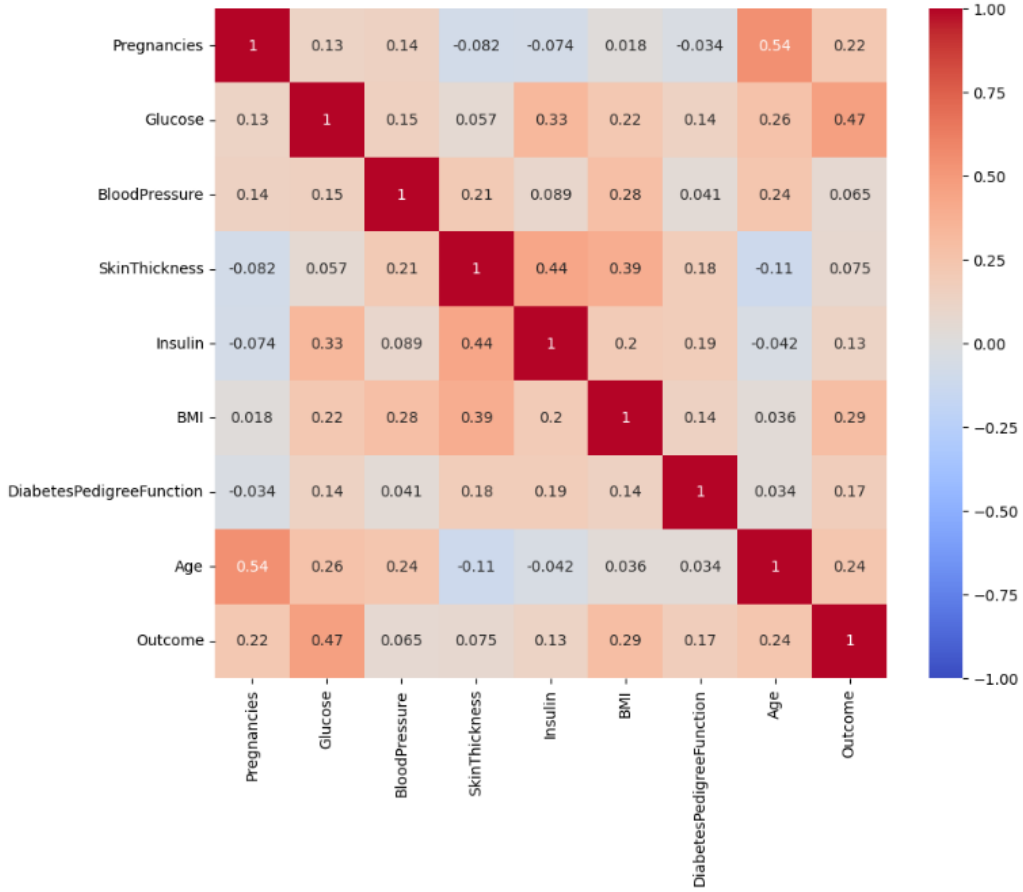


- Boxplot



Korelasyon Matrisi

Değişkenler arasındaki ilişkiyi incelemek için korelasyon matrisi ve heatmap çizelim.



4. Verinin Ön İşlemesi

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

Eksik Veriler İnceleme ve Doldurma

```
Glucose            5
BloodPressure     35
SkinThickness    227
Insulin          374
BMI               11
dtype: int64
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
dtype: int64
```

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.

```
(array([  4,   8,  13,  18,  43,  45,  57,  58,  88, 106, 111, 120, 120,
       123, 125, 125, 153, 159, 177, 177, 186, 220, 228, 228, 247, 248,
       286, 298, 330, 370, 370, 371, 392, 395, 409, 415, 445, 445, 445,
       453, 455, 459, 486, 549, 579, 584, 593, 597, 621, 645, 655, 666,
       673, 684, 691, 695, 753]), array([6, 4, 4, 2, 2, 6, 3, 6, 0, 2, 4, 3, 5, 7, 2, 5, 4, 0, 2, 5, 4, 4,
       4, 6, 4, 4, 4, 0, 6, 4, 6, 6, 4, 6, 4, 4, 3, 5, 6, 7, 0, 7, 4, 2,
       3, 4, 6, 2, 6, 4, 4, 7, 5, 7, 2, 4, 4]))
(array([  4,   8,  13,  18,  43,  45,  57,  58,  88, 106, 111, 120, 120,
       123, 125, 125, 153, 159, 177, 177, 186, 220, 228, 228, 247, 248,
       286, 298, 330, 370, 370, 371, 392, 395, 409, 415, 445, 445, 445,
       453, 455, 459, 486, 549, 579, 584, 593, 597, 621, 645, 655, 666,
       673, 684, 691, 695, 753]), array([6, 4, 4, 2, 2, 6, 3, 6, 0, 2, 4, 3, 5, 7, 2, 5, 4, 0, 2, 5, 4, 4,
       4, 6, 4, 4, 4, 0, 6, 4, 6, 6, 4, 6, 4, 4, 3, 5, 6, 7, 0, 7, 4, 2,
       3, 4, 6, 2, 6, 4, 4, 7, 5, 7, 2, 4, 4]))
(768, 9)
(718, 9)
```

Verinin Normalize Edilmesi

```
[[ 0.66162588  0.94337401 -0.01210076 ...  0.22724301  0.62799488
   1.5101737 ]
 [-0.86585766 -1.1820162  -0.54087872 ... -0.85404002 -0.34823249
  -0.16982219]
 [ 1.27261929  2.12414636 -0.71713803 ... -1.36378774  0.78716239
  -0.08140136]
 ...
 [ 0.35612917  0.03249249 -0.01210076 ... -0.91582763 -0.7231604
  -0.25824303]
 [-0.86585766  0.20117426 -1.06965667 ... -0.31339851 -0.35530661
   1.24491119]
 [-0.86585766 -0.91212538 -0.18836008 ... -0.26705781 -0.4755665
  -0.87718888]]
```

5. Veri Setinin Bölünmesi

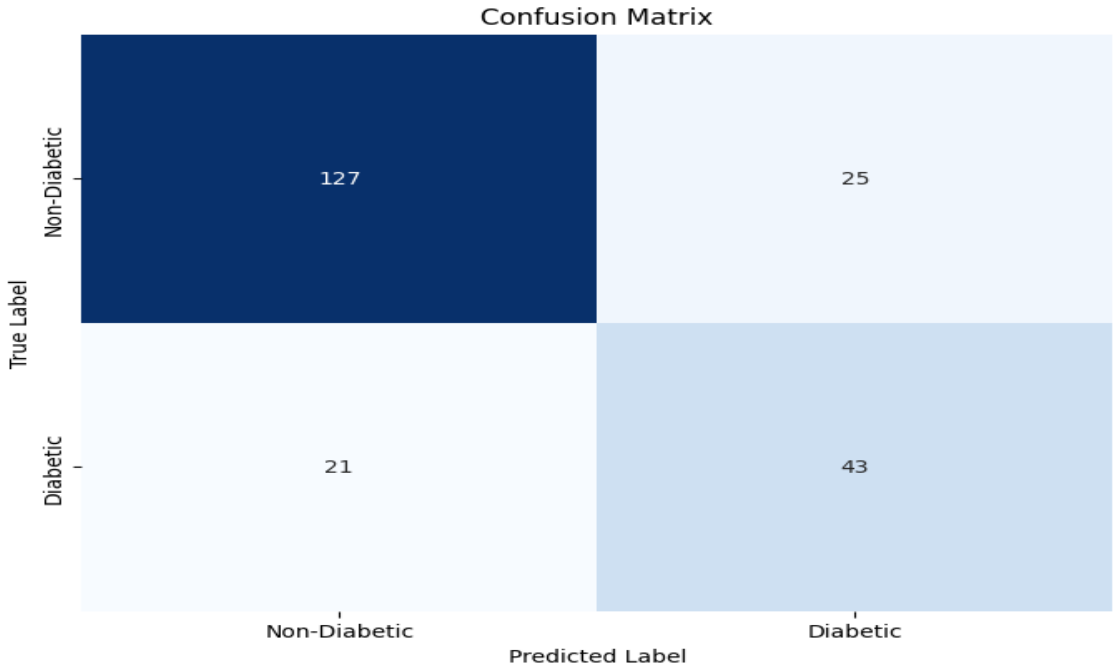
Veriyi eğitim ve test setlerine bölelim.

Kullandığım kod parçacığı aşağıdaki gibidir.

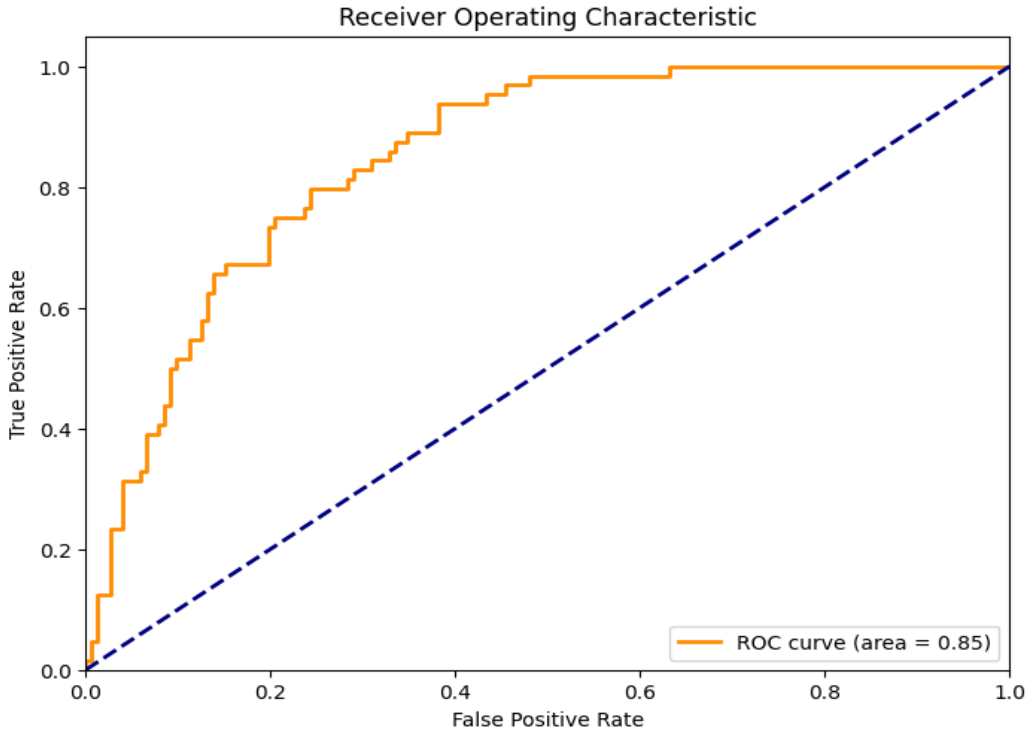
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

6. Model Eğitimi ve Değerlendirme

Naive Bayes modelini eğitip performansını değerlendirelim.




```
Accuracy: 0.79
Sensitivity (Recall): 0.67
Specificity: 0.84
Precision: 0.63
F1-Score: 0.65
```



- **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

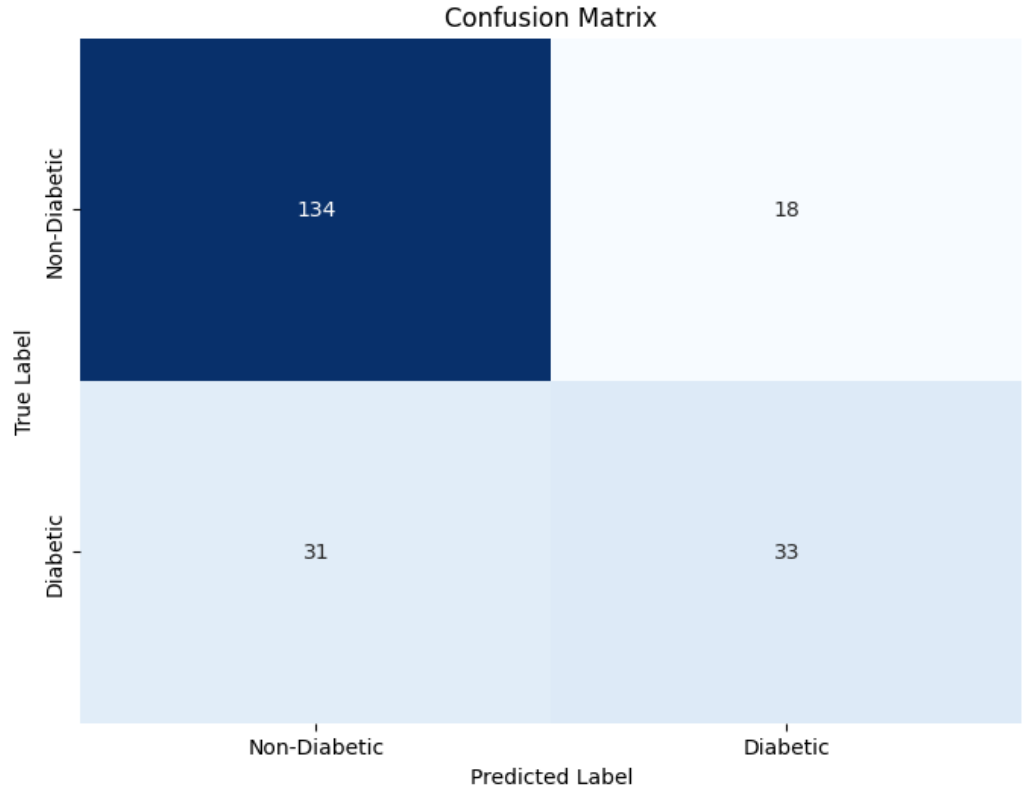
```
# ROC curve ve AUC

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

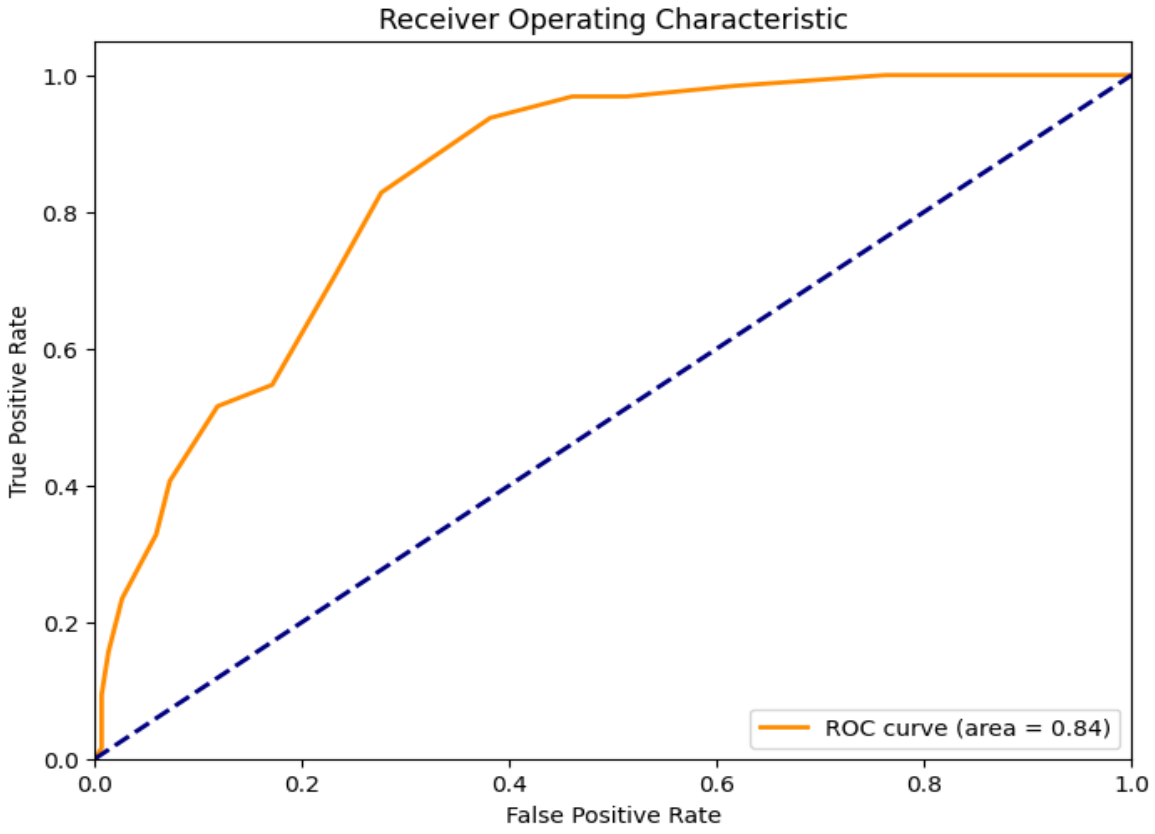
Tam süreci bir araya getirirlerek ve adım adım KNN modelini eğitip performansını değerlendirebiliriz.

Tam Süreç dosyaların içerisinde bulunan ML_FinalOdev.ipynb dosyasında bulunmaktadır.

```
En iyi k değeri: 19
```




```
Accuracy: 0.77
Sensitivity (Recall): 0.52
Specificity: 0.88
Precision: 0.65
F1-Score: 0.57
```



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)

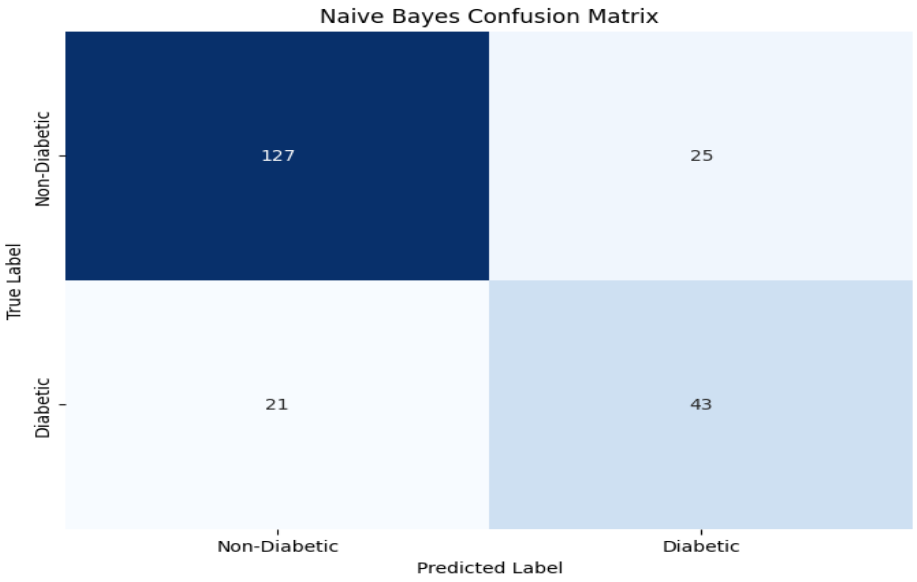
Outlier'ları belirleme (z-skoru yöntemi)

z_scores = np.abs(stats.zscore(df.drop('Outcome', axis=1)))

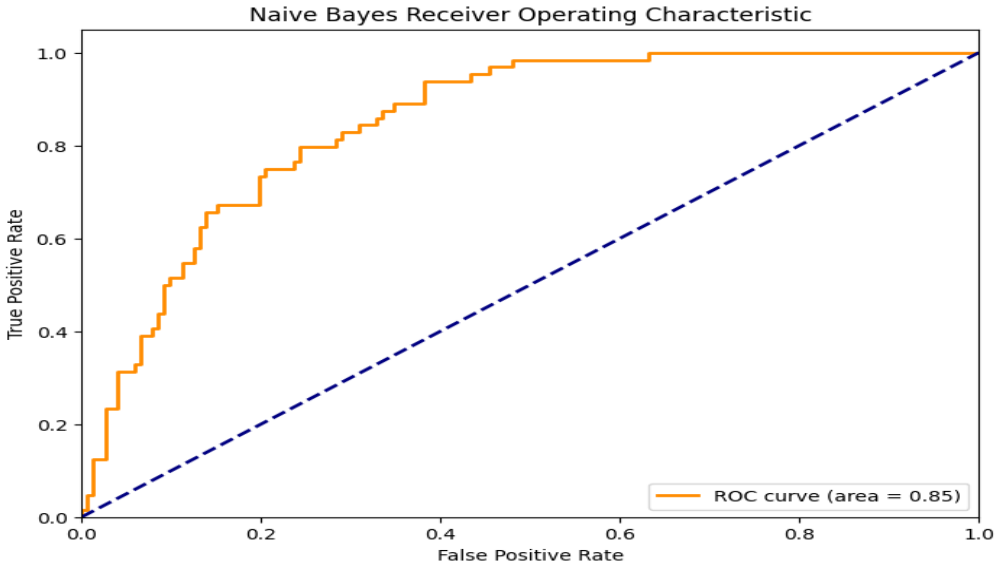
df_no_outliers = df[(z_scores < 3).all(axis=1)]

```
# 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:




```
Naive Bayes Accuracy: 0.79
Naive Bayes Sensitivity (Recall): 0.67
Naive Bayes Specificity: 0.84
Naive Bayes Precision: 0.63
Naive Bayes F1-Score: 0.65
```




```
Naive Bayes Mean Squared Error: 0.21
```

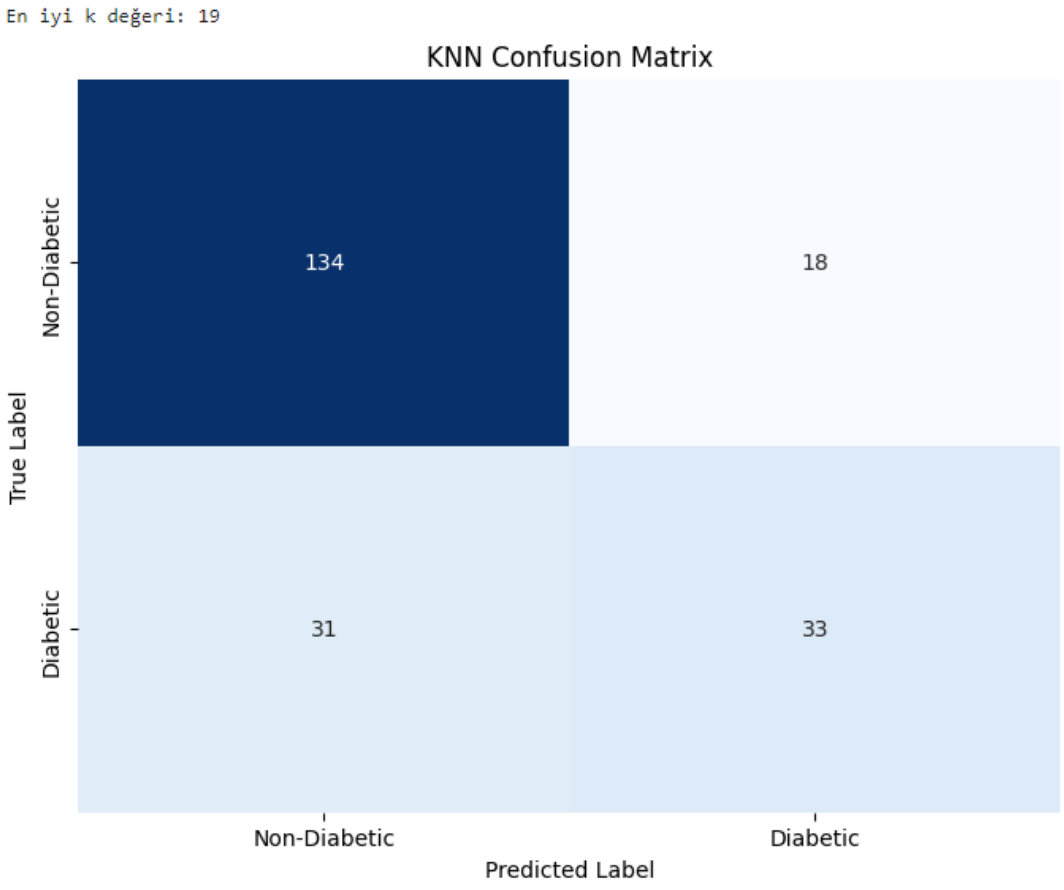
Sonuçların Yorumlanması

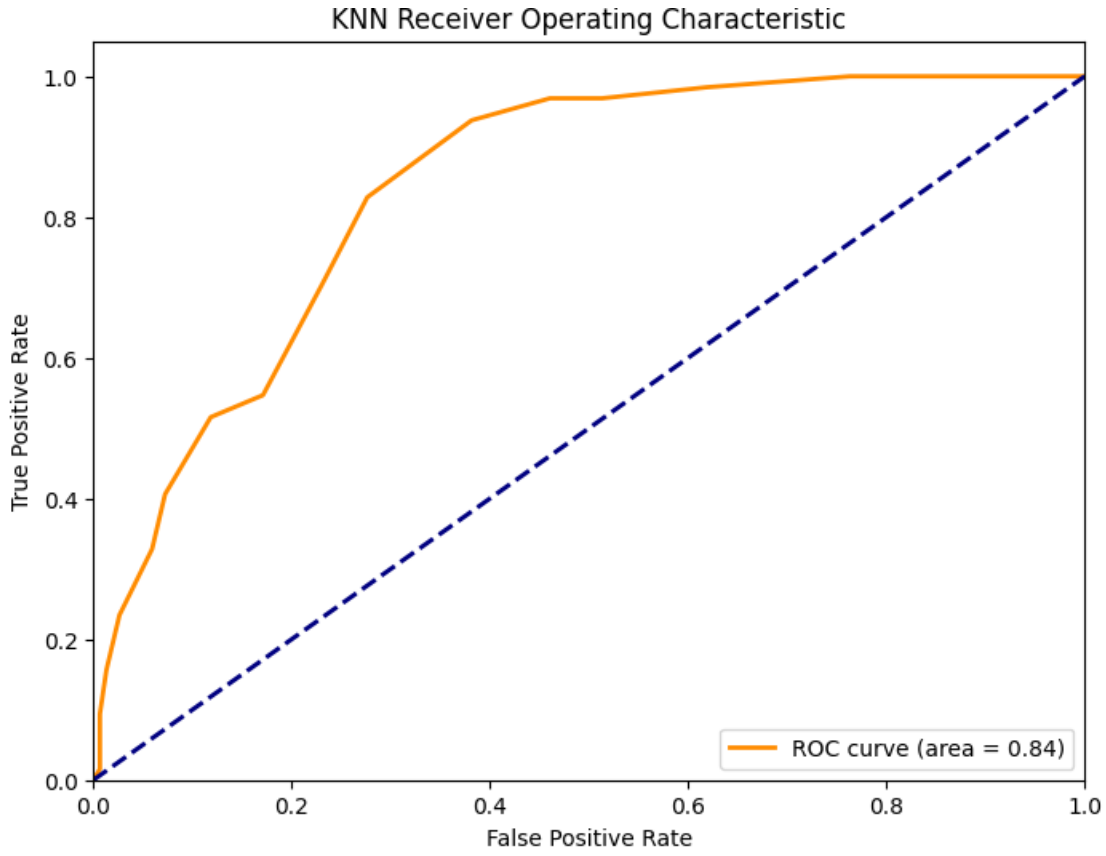
Naive Bayes Modeli Sonuçları:

- **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 :






```
KNN Accuracy: 0.77
KNN Sensitivity (Recall): 0.52
KNN Specificity: 0.88
KNN Precision: 0.65
KNN F1-Score: 0.57

KNN Mean Squared Error: 0.23
```

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

	Model	Accuracy	Recall	Specf.	Prec.	F1-Score	ROC-AUC
0	MLP	0.750000	0.562500	0.828947	0.580645	0.571429	0.771484
1	SVM	0.782407	0.562500	0.875000	0.654545	0.605042	0.853875
2	KNN	0.773148	0.515625	0.881579	0.647059	0.573913	0.841386
3	NBayes	0.787037	0.671875	0.835526	0.632353	0.651515	0.850740

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