

**ÜSKÜDAR ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**

**YAPAY ZEKA MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI**

**YZM506/1 MAKİNE ÖĞRENMESİ FİNAL ÖDEVİ**

**224312031- KÜBRA ZEYBEK**

**DERS HOCASI: Dr. Öğr. Üyesi GÖKALP TULUM**

1. **Giriş**
   * Bu çalışmada, Pima Indians Diabetes veri seti kullanılarak dört farklı makine öğrenmesi algoritmasının performansını değerlendirdik: Naive Bayes, K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP) ve Support Vector Machine (SVM). Bu modellerin doğruluğunu, F1 skorunu ve ROC AUC skorunu karşılaştırarak en iyi performansı gösteren modeli belirlemeyi amaçladık.
2. **Veri Setinin Tanıtılması**
   * Pima Indians Diabetes veri seti, kadınların gebelik sayısı, glikoz düzeyi, kan basıncı, cilt kalınlığı, insülin seviyesi, vücut kitle indeksi, diyabet soy ağacı fonksiyonu ve yaş gibi özellikleri içerir. Veri setinde diyabet hastalığına sahip olup olmadıklarını belirten bir sonuç sütunu bulunmaktadır.
3. **Veri Ön İşleme**
   * Veri setinde eksik değerler sıfır olarak kodlanmıştır. Bu eksik değerler NaN ile değiştirilmiş ve sütun ortalamaları ile doldurulmuştur. Veri seti, eğitim (%70) ve test (%30) olarak ikiye ayrılmıştır.
4. **Modeller ve Yöntemler**
   * Naive Bayes
     + Naive Bayes modeli, bağımsızlık varsayımı altında çalışır ve Bayes teoremini kullanarak sınıflandırma yapar. Bu model, eğitim verisi üzerinde eğitilmiş ve test verisi üzerinde değerlendirilmiştir.
   * KNN
     + KNN algoritması, en yakın k komşusuna göre sınıflandırma yapar. En iyi k değeri belirlenmiş ve model bu k değeri ile eğitilmiştir.
   * MLP
     + MLP, çok katmanlı yapay sinir ağıdır. Bu model, 1000 iterasyon boyunca eğitilmiş ve test verisi üzerinde değerlendirilmiştir.
   * SVM
     + SVM, veriyi yüksek boyutlu bir uzaya dönüştürerek sınıflandırma yapar. Model, doğrusal bir çekirdek ile eğitilmiş ve test verisi üzerinde değerlendirilmiştir.
5. **Sonuçlar ve Karşılaştırma**
   * Her modelin performansını aşağıdaki metriklerle karşılaştırdık:
     + Naive Bayes: Accuracy: 0.75, F1 Score: 0.61, ROC AUC Score: 0.73
     + KNN: Accuracy: 0.75, F1 Score: 0.58, ROC AUC Score: 0.70
     + MLP: Accuracy: 0.75, F1 Score: 0.61, ROC AUC Score: 0.73
     + SVM: Accuracy: 0.79, F1 Score: 0.65, ROC AUC Score: 0.75
6. **Sonuç ve Değerlendirme**
   * SVM modeli, diğer modellere göre daha yüksek doğruluk, F1 skoru ve ROC AUC skoru elde etmiştir. Bu sonuçlar, SVM modelinin Pima Indians Diabetes veri seti üzerinde en iyi performansı gösterdiğini ortaya koymaktadır.
7. **Kaynaklar**
   * Scikit-learn Documentation
   * Pima Indians Diabetes Data Set

**Python Kodları**

import pandas as pd

import numpy as np

# Veri setinin yüklenmesi

data = pd.read\_csv('veriseti.txt', delimiter='\t', header=None)

data.columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']

# Veri setinin ilk birkaç satırını ve temel istatistiklerini inceleme

print(data.head())

print(data.describe())

print(data['Outcome'].value\_counts())

# Eksik değerleri kontrol etme

print(data.isnull().sum())

# Eksik değerlerin yerini doldurma (sıfırları NaN ile değiştirme ve ortalama ile doldurma)

data.replace(0, np.nan, inplace=True)

data.fillna(data.mean(), inplace=True)

# Eksik değerlerin yeniden kontrolü

print(data.isnull().sum())

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

X = data.drop('Outcome', axis=1)

y = data['Outcome']

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

# Eğitim ve test veri setlerinin boyutlarını kontrol etme

print(f"X\_train shape: {X\_train.shape}")

print(f"y\_train value counts:\n{y\_train.value\_counts()}")

print(f"X\_test shape: {X\_test.shape}")

print(f"y\_test value counts:\n{y\_test.value\_counts()}")

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix, accuracy\_score, f1\_score, roc\_auc\_score, roc\_curve, mean\_squared\_error, precision\_score, recall\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Model değerlendirme fonksiyonu

def evaluate\_model(y\_test, y\_pred, model\_name):

labels = [0, 1]

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

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

f1 = f1\_score(y\_test, y\_pred, labels=labels)

precision = precision\_score(y\_test, y\_pred, labels=labels)

recall = recall\_score(y\_test, y\_pred, labels=labels)

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

sensitivity = recall

specificity = cm[0, 0] / (cm[0, 0] + cm[0, 1]) if (cm[0, 0] + cm[0, 1]) != 0 else 0

print(f"{model\_name} Model Performance")

print(f"Confusion Matrix:\n{cm}")

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

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

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

print(f"Recall: {recall:.2f}")

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

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

print(f"Mean Squared Error: {mse:.2f}")

# ROC AUC skoru hesaplama ve ROC eğrisi çizme

if len(np.unique(y\_test)) > 1 and len(np.unique(y\_pred)) > 1:

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

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

plt.figure()

plt.plot(fpr, tpr, label=f'{model\_name} (area = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title(f'{model\_name} ROC Curve')

plt.legend(loc='best')

plt.savefig(f"{model\_name}\_roc\_curve.png")

plt.show()

else:

print(f"ROC AUC Score: Not defined (single class present in y\_true or y\_pred)")

plt.figure()

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)

plt.title(f'{model\_name} Confusion Matrix')

plt.ylabel('True label')

plt.xlabel('Predicted label')

plt.savefig(f"{model\_name}\_confusion\_matrix.png")

plt.show()

# Naive Bayes

nb\_model = GaussianNB()

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

y\_pred\_nb = nb\_model.predict(X\_test)

evaluate\_model(y\_test, y\_pred\_nb, "Naive Bayes")

# KNN

k\_values = range(1, 21)

accuracy\_scores = []

for k in k\_values:

knn\_model = KNeighborsClassifier(n\_neighbors=k)

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

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

accuracy\_scores.append(accuracy\_score(y\_test, y\_pred\_knn))

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

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

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

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

evaluate\_model(y\_test, y\_pred\_knn, "KNN")

# MLP

mlp\_model = MLPClassifier(max\_iter=1000)

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

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

evaluate\_model(y\_test, y\_pred\_mlp, "MLP")

# SVM

svm\_model = SVC(probability=True)

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

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

evaluate\_model(y\_test, y\_pred\_svm, "SVM")

**SONUÇLAR :**

**Naive Bayes**

**Confusion Matrix:**

|  | **Predicted Negative** | **Predicted Positive** |
| --- | --- | --- |
| **Actual Negative** | **127** | **24** |
| **Actual Positive** | **33** | **47** |

**ROC Curve:**

**KNN**

**Confusion Matrix:**

|  | **Predicted Negative** | **Predicted Positive** |
| --- | --- | --- |
| **Actual Negative** | **132** | **19** |
| **Actual Positive** | **38** | **42** |

**ROC Curve:**

**MLP**

**Confusion Matrix:**

|  | **Predicted Negative** | **Predicted Positive** |
| --- | --- | --- |
| **Actual Negative** | **125** | **26** |
| **Actual Positive** | **31** | **49** |

**ROC Curve:**

**SVM**

**Confusion Matrix:**

|  | **Predicted Negative** | **Predicted Positive** |
| --- | --- | --- |
| **Actual Negative** | **136** | **15** |
| **Actual Positive** | **33** | **47** |