Rapor: Basit Bir Tahmin Modeli Oluşturma

Giriş

Bu rapor, Iris veri seti kullanılarak oluşturulan basit bir tahmin modeli projesini kapsamaktadır. Proje kapsamında veriler ön işlenmiş, lojistik regresyon modeli eğitilmiş, modelin performansı değerlendirilmiş ve sonuçlar yorumlanmıştır.

Veri Seti

Iris veri seti, 150 örnekten oluşur ve her örnek 4 özellik içerir: sepal length, sepal width, petal length ve petal width. Hedef değişken ise 3 farklı türden birini belirtir: setosa, versicolor veya virginica.

Modelleme Süreci

1. Veri Yükleme ve Keşifsel Veri Analizi (EDA)

İlk adımda, Iris veri seti pandas kütüphanesi kullanılarak yüklendi. Veri setinin genel istatistiksel özeti çıkarıldı ve eksik değerler kontrol edildi. Hedef değişkenin dağılımı incelendi.

Kod:

import pandas as pd

from sklearn.datasets import load\_iris

# Iris veri setini yükleyin

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

# Veri setinin ilk birkaç satırını görüntüleyin

print("Veri Setinin İlk 5 Satırı:")

print(df.head())

# Keşifsel Veri Analizi (EDA)

# Veri setinin genel istatistiksel özetini alın

print("\nVeri Setinin Genel İstatistiksel Özeti:")

print(df.describe())

# Veri setinde eksik değer olup olmadığını kontrol edin

print("\nEksik Değer Kontrolü:")

print(df.isnull().sum())

# Hedef değişkenin dağılımını kontrol edin

print("\nHedef Değişkenin Dağılımı:")

print(df['target'].value\_counts())

2. Veri Ön İşleme

Veri seti, özellikler (X) ve hedef değişken (y) olarak ayrıldı. Veriler eğitim ve test setlerine ayrıldı (eğitim seti %80, test seti %20). Özellikler, StandardScaler kullanılarak standartlaştırıldı.

Kod:

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

from sklearn.preprocessing import StandardScaler

# Veriyi özellikler (X) ve hedef (y) olarak ayırın

X = df.drop('target', axis=1)

y = df['target']

# Veriyi eğitim ve test setlerine ayırın

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standartlaştırıcıyı oluşturun ve eğitin, ardından eğitim ve test verilerini dönüştürün

scaler = StandardScaler()

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

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

3. Model Eğitimi

Lojistik regresyon modeli sklearn kütüphanesi kullanılarak oluşturuldu ve eğitim seti ile eğitildi. Modelin eğitimi için azami iterasyon sayısı 200 olarak belirlendi.

Kod:

from sklearn.linear\_model import LogisticRegression

# Lojistik regresyon modelini oluşturun

model = LogisticRegression(max\_iter=200)

# Modeli eğitim verileriyle eğitin

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

4. Model Performansını Değerlendirme

Eğitim tamamlandıktan sonra, model test seti üzerinde değerlendirildi. Modelin doğruluk (accuracy), kesinlik (precision) ve duyarlılık (recall) metrikleri hesaplandı. Ayrıca, karışıklık matrisi ve sınıflandırma raporu oluşturuldu.

Kod:

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix, classification\_report

# Modelin test setindeki tahminlerini alın

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

# Performans metriklerini hesaplayın

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

precision = precision\_score(y\_test, y\_pred, average='macro')

recall = recall\_score(y\_test, y\_pred, average='macro')

# Karışıklık matrisi

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

# Sınıflandırma raporu

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

# Performans metriklerini ve karışıklık matrisini yazdırın

print(f"\nModel Performans Metrikleri:\nAccuracy: {accuracy}\nPrecision: {precision}\nRecall: {recall}")

print("\nConfusion Matrix:")

print(conf\_matrix)

print("\nClassification Report:")

print(class\_report)

5. Sonuçların Görselleştirilmesi

Karışıklık matrisini ve sınıflandırma raporunu görselleştirmek için aşağıdaki kod kullanılmıştır:

import matplotlib.pyplot as plt

import seaborn as sns

# Karışıklık matrisini görselleştirme

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

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

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

Değerlendirme Sonuçları

Modelin test seti üzerindeki performansı aşağıdaki gibidir:

\* Doğruluk (Accuracy): 0.93

\* Kesinlik (Precision): 0.94

\* Duyarlılık (Recall): 0.93

Karışıklık Matrisi

[[10 0 0]

[ 0 8 2]

[ 0 0 10]]

Sınıflandırma Raporu

precision recall f1-score support

0 1.00 1.00 1.00 10

1 1.00 0.80 0.89 10

2 0.83 1.00 0.91 10

accuracy 0.93 30

macro avg 0.94 0.93 0.93 30

weighted avg 0.94 0.93 0.93 30

Sonuçlar ve Yorumlar

Model, genel olarak yüksek doğruluk, kesinlik ve duyarlılık oranlarına sahiptir. Ancak, sınıf 1 için duyarlılık oranı (0.80) diğer sınıflara göre daha düşüktür. Bu, modelin sınıf 1'i diğer sınıflara kıyasla daha az doğru bir şekilde tahmin ettiğini göstermektedir.

Potansiyel İyileştirmeler

1. Daha Fazla Veri: Modelin performansını artırmak için daha fazla veri toplanabilir. Daha büyük veri setleri, modelin genelleme yeteneğini artırabilir.

2. Diğer Modellerin Denenmesi: Farklı makine öğrenimi modelleri (örneğin, destek vektör makineleri, karar ağaçları veya rastgele ormanlar) denenebilir ve performans karşılaştırması yapılabilir.

3. Özellik Mühendisliği: Yeni özellikler türetmek veya mevcut özellikleri iyileştirmek, modelin performansını artırabilir.

4. Modelin Parametre Ayarı: Modelin hiperparametreleri (örneğin, lojistik regresyon için düzenlileştirme parametresi) optimize edilebilir.

5. Veri Dengesizliği İle Başa Çıkma: Veri dengesizliği durumunda, SMOTE gibi tekniklerle veri çoğaltma veya sınıf ağırlıklarını ayarlama stratejileri kullanılabilir.

Sonuç

Bu proje, Iris veri seti kullanılarak basit bir tahmin modeli oluşturmanın temel adımlarını kapsamaktadır. Lojistik regresyon modeli, veri seti üzerinde yüksek performans göstermiştir. Ancak, performansın daha da artırılabilmesi için çeşitli iyileştirme yöntemleri uygulanabilir.