علی آرمان 4023624002

در پیاده سازی ابتدا یک بار کد ها نوشته شد و در نهایت مرتب شد و به صورت بهینه کنار هم قرار گرفت.

در ابتدا نگارش کد ها گزارش شده و بعد از آن اجرای کد.

در ابتدا کتابخانه های لازم را ایمپورت میکنیم.

1. import numpy as np

2. import pandas as pd

3. import seaborn as sns

4. import matplotlib.pyplot as plt

5. from sklearn.preprocessing import LabelEncoder

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

7. from sklearn.metrics import classification\_report, confusion\_matrix

فرایند پیش پردازش همانند پروژه های قبلی می باشد:

ابتدا فایل را باز میکنیم:

1. # Step 1: Opening file

2. def read\_dataset(file\_path):

3. return pd.read\_csv(file\_path)

در مرحله تمام داده ها را انکود کرده و سپس فیت ترنسفورم میکنیم:

1. # Step 2: Encode categorical data using LabelEncoder

2. def encode\_categorical\_data(data):

3. label\_encoder = LabelEncoder()

4. for column in data.select\_dtypes(include=['object']).columns:

5. data[column] = label\_encoder.fit\_transform(data[column])

6. return data

در مرحله بعدی داده هایی که حذف شده اند را با میانگین همان ستون پر میکنیم:

1. # Step 3: Handle missing values by replacing with the mean

2. def handle\_missing\_values(data):

3. data\_filled = data.fillna(data.mean())

4. return data\_filled

در مرحله بعدی طبق تنظیمات 80 درصد داده اموزشی شده و شافل غیر فعال:

1. # Step 4: Train-Test Split

2. def split\_data(X, y):

3. return train\_test\_split(X, y, test\_size=0.2, random\_state=17, shuffle=False)

تا اینجا پیش پردازش تمام شده و از اینجا به بعد پیاده سازی خود الگورتیم هست:

در اینجا احتمالات پیشینی داده های اموزشی را حساب میکنیم و تعداد را به اندازه تقسیم میکنیم و آن کلاس را با احتمالش به صورت دیکشنری بازگشت میدهیم:

1. def calculate\_prior\_probabilities(y):

2. classes, counts = np.unique(y, return\_counts=True)

3. probabilities = counts / len(y)

4. return dict(zip(classes, probabilities))

در مرحله بعدی احتمال را حساب میکنیم که x\_train و y\_train را دریافت میکنیم و تعداد مشاهده X را با تقسیم حساب میکنیم که از اپسیلون برای اسموفینگ استفاده شده تا حدی نرمال سازی شود و از تقسیم بر صفر جلوگیری شود.

1. def calculate\_class\_likelihoods(X, y, epsilon=1e-9):

2. likelihoods = {}

3. for class\_val in np.unique(y):

4. class\_data = X[y == class\_val]

5. class\_likelihood = (class\_data.sum(axis=0) + 1) / (class\_data.shape[0] + 2 + epsilon)

6. likelihoods[class\_val] = class\_likelihood

7. return likelihoods

در نهایت در تابع naive\_bayes\_predict ما یک لیست برای پیش بینی اماده کردیم و در تمامی داده های اموزشی مقدار مورد نظیر را با استفاده از لگاریتم احتمال پسین و لاکلی هود امتیاز میدیم و در نهایت بیشترین امتیاز را با تابع ماکس انتخاب میکنیم و به لیست اولیه اضافه میکنیم و در نهایت کل لیست را بازگشت میدهیم.

1. def naive\_bayes\_predict(X, prior\_probabilities, class\_likelihoods):

2. predictions = []

3. for sample in X.values:

4. class\_scores = {}

5. for class\_val, class\_likelihood in class\_likelihoods.items():

6. class\_score = np.log(prior\_probabilities[class\_val]) + np.sum(np.log(class\_likelihood[sample != 0]))

7. class\_scores[class\_val] = class\_score

8. predicted\_class = max(class\_scores, key=class\_scores.get)

9. predictions.append(predicted\_class)

10. return predictions

در نهایت یک تابع برای نمایش پلات هم در نظر گرفته شده:

1. def plot\_confusion\_matrix(y\_true, y\_pred, classes):

2. cm = confusion\_matrix(y\_true, y\_pred)

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

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

5. plt.title("Confusion Matrix")

6. plt.xlabel("Predicted Label")

7. plt.ylabel("True Label")

8. plt.show()

قسمت اجرا:

برای اجرا ابتدا با استفاده از توابع نوشته شده در بالا، فایل باز شده، اندکود و ترنسفورم انجام شده و مقادیر از دست رفته با میانگین پر شده:

1. # Main Execution

2. file\_path = 'Telecust1.csv'

3. data = read\_dataset(file\_path)

4. data\_encoded = encode\_categorical\_data(data)

5. data\_filled = handle\_missing\_values(data\_encoded)

در قسمت بعدی از ستون اول تا اخر به عنوان x و ستون اخر به عنوان y انتخاب شده و طبق همین با استفاده از تابع split\_data داده برای تست و آموزش از هم جدا شده اند:

1. # Split attributes and labels

2. X = data\_filled.iloc[:, 1:-1]

3. y = data\_filled['custcat']

4. X\_train, X\_test, y\_train, y\_test = split\_data(X, y)

در نهایت ابتدا احتمالا پیشینی و لایکلی هود حساب شده و با استفاده از این دو تابع naïve\_bayes\_predict برای داده های اموزشی و تست فراخوانی شده:

1. # Calculate prior probabilities and class likelihoods

2. prior\_probabilities = calculate\_prior\_probabilities(y\_train)

3. class\_likelihoods = calculate\_class\_likelihoods(X\_train, y\_train)

4.

5. # Step 6: Implement Naive Bayes Classifier

6. y\_pred\_train = naive\_bayes\_predict(X\_train, prior\_probabilities, class\_likelihoods)

7. y\_pred\_test = naive\_bayes\_predict(X\_test, prior\_probabilities, class\_likelihoods)

در نهایت با استفاده از classification\_report هر دو داده آموزشی و تست گزارش شده و نمودار مربوطه نمایش داده شده. همچنین جهت جلوگیری از احتمال تقسیم بر صفر zero\_devision قرار داده شده:

1. # Step 7: Evaluate the classifier

2. print("Training Classification Report:")

3. print(classification\_report(y\_train, y\_pred\_train, zero\_division=1))

4. plot\_confusion\_matrix(y\_train, y\_pred\_train, classes=np.unique(y))

5.

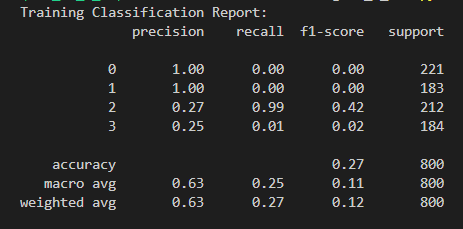
6. print("\nTesting Classification Report:")

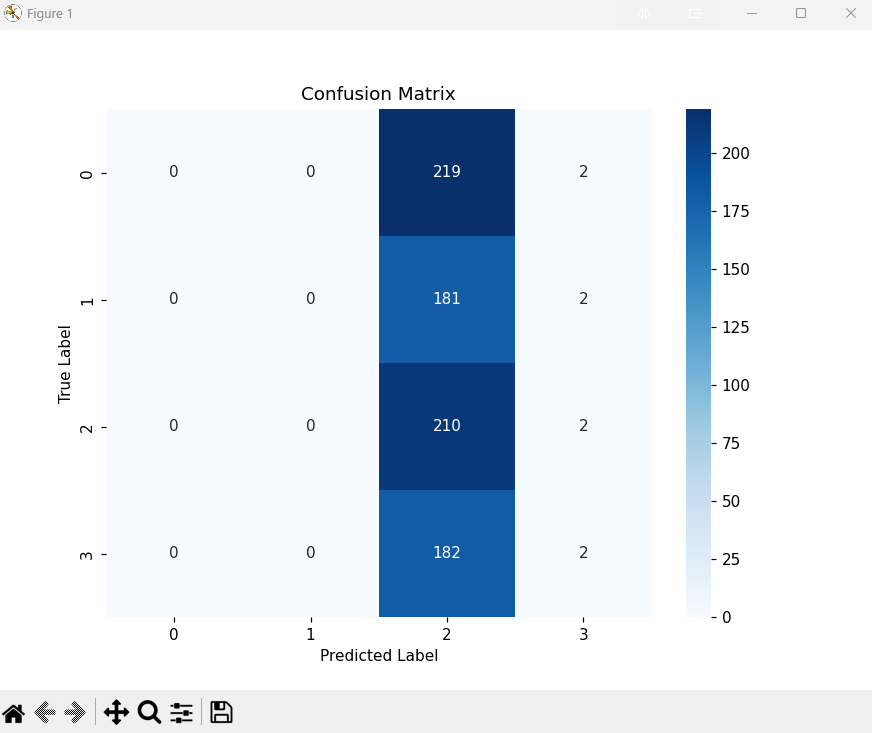
7. print(classification\_report(y\_test, y\_pred\_test, zero\_division=1))

8. plot\_confusion\_matrix(y\_test, y\_pred\_test, classes=np.unique(y))

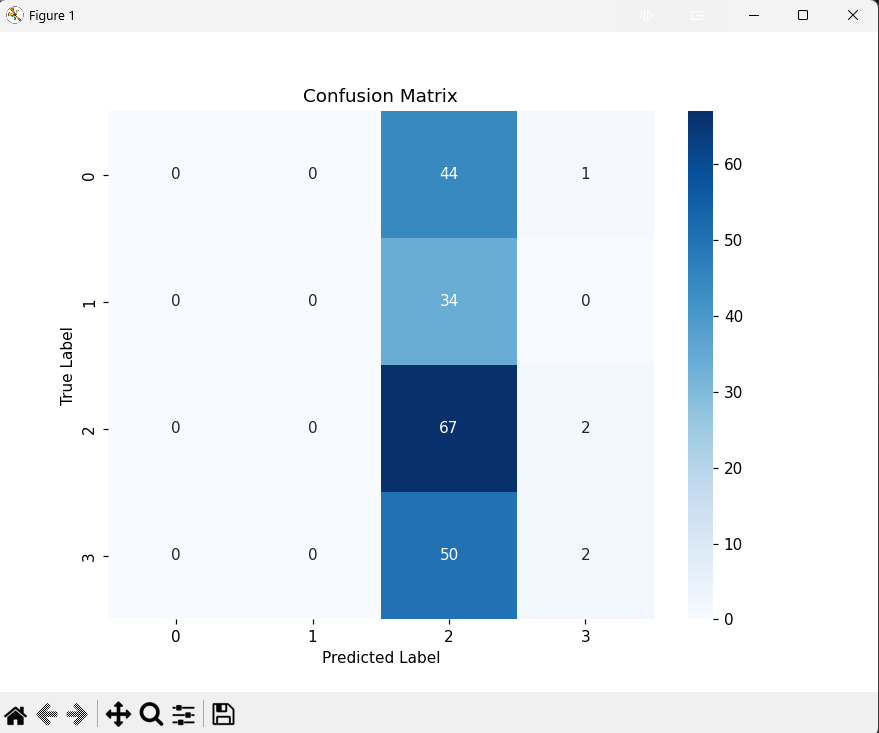
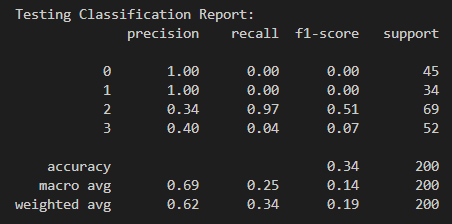
خروجی و گزارش:

برای داده اموزشی:





برای داده تست:



همان طور که قابل مشاهده هست کد نوشته شده با دقت خوبی دسته بندی را انجام میدهد.