علی آرمان ۴۰۲۳۶۲۴۰۰۲

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 ۸۰ درصد داده اموزشی شده و شافل غیر فعال:

```
# Step 4: Train-Test Split
def split_data(X, y):
    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:

```
def calculate_prior_probabilities(y):
    classes, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    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.

```
def calculate_class_likelihoods(X, y, epsilon=1e-9):
    likelihoods = {}
    for class_val in np.unique(y):
        class_data = X[y == class_val]
        class_likelihood = (class_data.sum(axis=0) + 1) / (class_data.shape[0] + 2 + epsilon)
        likelihoods[class_val] = class_likelihood
    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.

```
def naive_bayes_predict(X, prior_probabilities, class_likelihoods):
    predictions = []
    for sample in X.values:
        class_scores = {}
        for class_val, class_likelihood in class_likelihoods.items():
            class_score = np.log(prior_probabilities[class_val]) +
np.sum(np.log(class_likelihood[sample != 0]))
            class_scores[class_val] = class_score
        predicted_class = max(class_scores, key=class_scores.get)
        predictions.append(predicted_class)
    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:

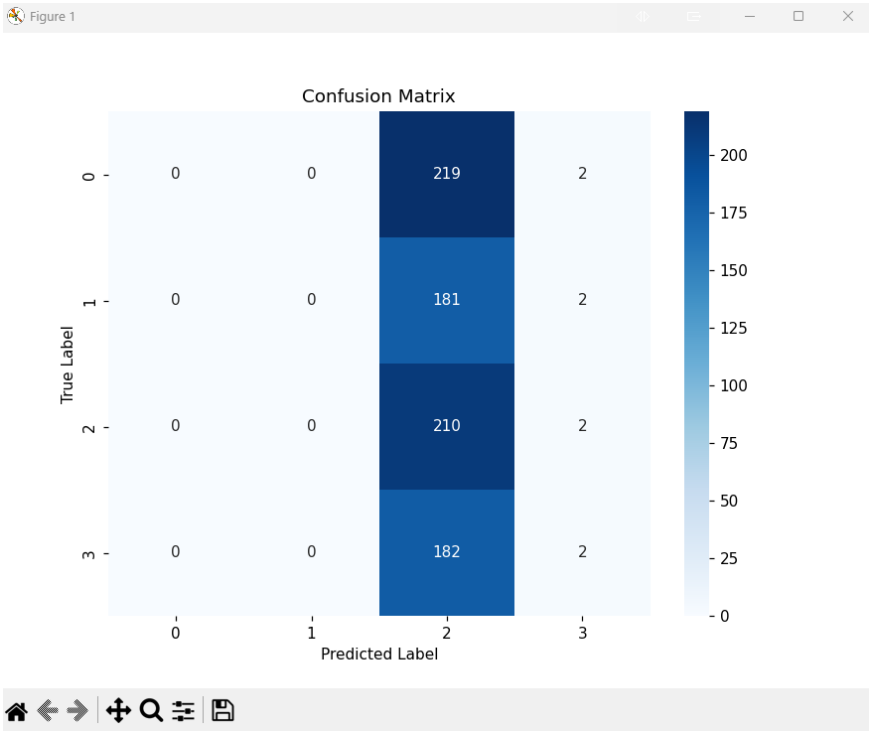
```
1. # Step 7: Evaluate the classifier
2. print("Training Classification Report:")
3. print(classification_report(y_train, y_pred_train, zero_division=1))
4. plot_confusion_matrix(y_train, y_pred_train, classes=np.unique(y))
5.
6. print("\nTesting Classification Report:")
7. print(classification_report(y_test, 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:

Training Classification Report:

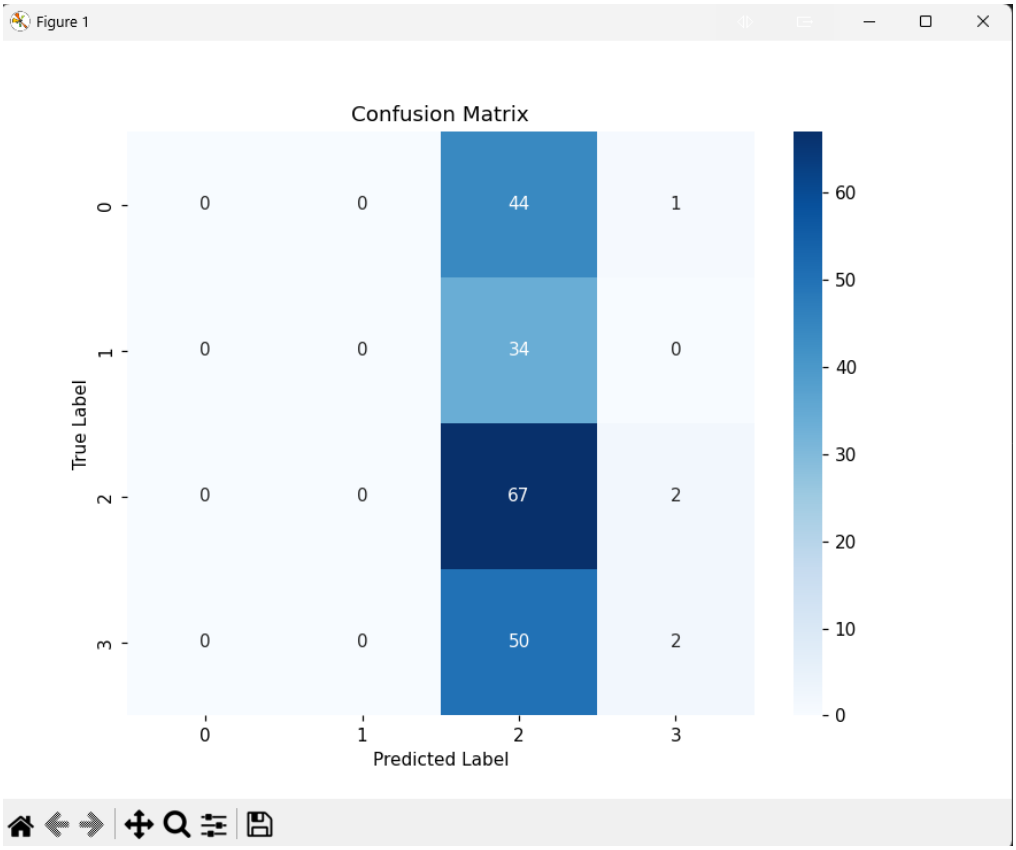
	precision	recall	f1-score	support
0	1.00	0.00	0.00	221
1	1.00	0.00	0.00	183
2	0.27	0.99	0.42	212
3	0.25	0.01	0.02	184
accuracy			0.27	800
macro avg	0.63	0.25	0.11	800
weighted avg	0.63	0.27	0.12	800



برای داده تست:

Testing Classification Report:

	precision	recall	f1-score	support
0	1.00	0.00	0.00	45
1	1.00	0.00	0.00	34
2	0.34	0.97	0.51	69
3	0.40	0.04	0.07	52
accuracy			0.34	200
macro avg	0.69	0.25	0.14	200
weighted avg	0.62	0.34	0.19	200



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