Pattern Recognition

NAME
Ammar Yasser Abdallah

ID
22010465

Githup link:

https://github.com/AmmarYasser72/PatternRecognition1.git

First I import this Libraries

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
```

This code loads data from two separate files (adult.data and adult.test) into pandas DataFrames and setting appropriate column names

```python
train_file_path = "D:\\Pattern Recognition\\adult.data"
test_file_path = "D:\\Pattern Recognition\\adult.test"
# Define column names
columns = ["age", "workclass", "fnlwgt", "education", "education-num", "marital-status",
           "occupation", "relationship", "race", "sex", "capital-gain", "capital-loss",
           "hours-per-week", "native-country", "income"]

train_data = pd.read_csv(train_file_path, header=None, names=columns, na_values=" ?")
test_data = pd.read_csv(test_file_path, header=1, names=columns, na_values=" ?") # Skip row 1

print(train_data.head())
print("\n")
print("Shape of train_data before dropping:", train_data.shape)
print(".................................................................................")
print(test_data.head())
print("\n")
print("Shape of test_data before dropping:", test_data.shape)
```

OUTPUT : *DATA FROM ADULT.DATA*

```
   age         workclass  fnlwgt   education  education-num  \
0   39         State-gov   77516   Bachelors             13
1   50  Self-emp-not-inc   83311   Bachelors             13
2   38           Private  215646     HS-grad              9
3   53           Private  234721        11th              7
4   28           Private  338409   Bachelors             13

       marital-status         occupation    relationship   race      sex  \
0       Never-married       Adm-clerical   Not-in-family  White     Male
1  Married-civ-spouse    Exec-managerial         Husband  White     Male
2            Divorced  Handlers-cleaners   Not-in-family  White     Male
3  Married-civ-spouse  Handlers-cleaners         Husband  Black     Male
4  Married-civ-spouse     Prof-specialty            Wife  Black   Female

   capital-gain  capital-loss  hours-per-week  native-country  income
0          2174             0              40   United-States   <=50K
1             0             0              13   United-States   <=50K
2             0             0              40   United-States   <=50K
3             0             0              40   United-States   <=50K
4             0             0              40            Cuba   <=50K


Shape of train_data before dropping: (32561, 15)
..................................................................................
```

OUTPUT : *DATA FROM ADULT.TEST*

```
   age   workclass  fnlwgt      education  education-num       marital-status  \
0   38     Private   89814        HS-grad              9   Married-civ-spouse
1   28   Local-gov  336951     Assoc-acdm             12   Married-civ-spouse
2   44     Private  160323   Some-college             10   Married-civ-spouse
3   18         NaN  103497   Some-college             10        Never-married
4   34     Private  198693           10th              6        Never-married

          occupation   relationship   race     sex  capital-gain  \
0     Farming-fishing       Husband  White    Male             0
1     Protective-serv       Husband  White    Male             0
2   Machine-op-inspct       Husband  Black    Male          7688
3                 NaN     Own-child  White  Female             0
4       Other-service Not-in-family  White    Male             0

   capital-loss  hours-per-week  native-country   income
0             0              50   United-States   <=50K.
1             0              40   United-States    >50K.
2             0              40   United-States    >50K.
3             0              30   United-States   <=50K.
4             0              30   United-States   <=50K.


Shape of test_data before dropping: (16280, 15)
```

This code deals with handling missing values in the DataFrames, followed by displaying the shapes of the datasets after dropping the missing values.

```
train_data.dropna(inplace=True)
test_data.dropna(inplace=True)

# Output shape after dropping
print("Shape of train_data after dropping:", train_data.dropna().shape)
print("Shape of test_data after dropping:", test_data.dropna().shape)
```

Output

```
Shape of train_data after dropping: (30162, 15)
Shape of test_data after dropping: (15059, 15)
```

This code segment preprocesses the data by converting the "income" column into binary values, performs one-hot encoding on categorical variables, and splits the combined dataset into training and testing sets. And displays the shapes of the training and testing data

```python
# Convert income column to binary
train_data["income"] = train_data["income"].apply(lambda x: 1 if x == ' >50K' else 0)
test_data["income"] = test_data["income"].apply(lambda x: 1 if x == ' >50K.' else 0)

# One-hot encode
combined_data = pd.concat([train_data, test_data])
combined_data = pd.get_dummies(combined_data, columns=["workclass", "education", "marital-status",
                                                       "occupation", "relationship", "race", "sex", "native-country"])

# Split the combined data back into training and testing sets
X_train = combined_data[:len(train_data)].drop("income", axis=1)
y_train = combined_data[:len(train_data)]["income"]
X_test = combined_data[len(train_data):].drop("income", axis=1)
y_test = combined_data[len(train_data):]["income"]

# Display the shapes of training and testing data
print("Shape of X_train:", X_train.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_test:", y_test.shape)
```

Output

```
Shape of X_train: (30162, 104)
Shape of y_train: (30162,)
Shape of X_test: (15059, 104)
Shape of y_test: (15059,)
```

This code segment trains a Naive Bayes classifier on the training data predicts income levels for the testing data and computes sensitivity and specificity using the confusion matrix. If the confusion matrix has only one row it prints an error message else it calculates sensitivity and specificity and prints their values

```
# Train Naive Bayes Classifier
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)

# Predict income level for testing data
y_pred = nb_classifier.predict(X_test)

# Compute Sensitivity and Specificity
conf_matrix = confusion_matrix(y_test, y_pred)

# Check if the confusion matrix has multiple rows (indicating predictions for both classes)
if conf_matrix.shape[0] < 2:
    print("Error: Confusion matrix has only one row, indicating predictions for only one class.")
else:
    # Extract values from confusion matrix
    TP = conf_matrix[1, 1]
    FP = conf_matrix[0, 1]
    TN = conf_matrix[0, 0]
    FN = conf_matrix[1, 0]

    sensitivity = TP / (TP + FN)
    specificity = TN / (TN + FP)

    print("Sensitivity:", sensitivity)
    print("Specificity:", specificity)
```

Output

```
Sensitivity: 0.3062162162162162
Specificity: 0.9458579100272911
```

This code segment predicts the probabilities of each class for the testing data using the trained Naive Bayes classifier then extracts the probability of the positive class "**making over 50K a year**" by selecting the second column of the posterior probabilities then prints the posterior probabilities of making over 50K a year.

```
# Predict probabilities for testing data
posterior_probs = nb_classifier.predict_proba(X_test)

# Extract the probability of the positive class (making over 50K a year)
positive_class_probs = posterior_probs[:, 1]

# Print the posterior probabilities
print("Posterior Probabilities of making over 50K a year:")
print(positive_class_probs)
```

Output

```
Posterior Probabilities of making over 50K a year:
[0.01595412 0.00665536 1.         ... 0.02297512 0.99999491 0.02961601]
```

BY

Ammar Yasser Abdallah

22010465