Name : Rana Ahmed Mohamed Gaber

ID : 2203180

Department : Intelligent Systems

Subject : pattern recognition

---

[2]:
```
import pandas as pd
import numpy as np
import sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix , accuracy_score , f1_score , recall_score
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
df = pd.read_csv('/kaggle/input/adult-data/adult.data')
```

[4]:
```
df.head()
```

[4]:

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K

Before i did anything , I imported all the necessary libraries and saved the dataset in the variable df , then displayed the first 5 records.

[36]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32560 entries, 0 to 32559
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   39              32560 non-null  int64 
 1    State-gov      32560 non-null  object
 2    77516          32560 non-null  int64 
 3    Bachelors      32560 non-null  object
 4    13             32560 non-null  int64 
 5    Never-married  32560 non-null  object
 6    Adm-clerical   32560 non-null  object
 7    Not-in-family  32560 non-null  object
 8    White          32560 non-null  object
 9    Male           32560 non-null  object
 10   2174           32560 non-null  int64 
 11   0              32560 non-null  int64 
 12   40             32560 non-null  int64 
 13   United-States  32560 non-null  object
 14   <=50K          32560 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Then , I was checking if there is any null value so I needed to get info about the data

Apparently there wasn’t any null values , so I checked once more and summed up the null values

[37]:
```
df.isnull().sum()
```

[37…
```
39                0
 State-gov        0
 77516            0
 Bachelors        0
 13               0
 Never-married    0
 Adm-clerical     0
 Not-in-family    0
 White            0
 Male             0
 2174             0
 0                0
 40               0
 United-States    0
 <=50K            0
dtype: int64
```

[39]:
```
encoder = OneHotEncoder()
```

[40]:
```
x = df.iloc[: , :-1]
y = df[df.columns[-1]]
```

Then I saved the OneHotEncoder in a variable to use it when I encode the categorical features ,then I separated the features in variable x and the target column in variable y

[43]:
```
x = np.asarray(x)
y = np.asarray(y)
```

[44]:
```
y = y.reshape(-1 , 1)
```

[45]:
```
x = encoder.fit_transform(x)
y = encoder.fit_transform(y)
```

[46]:
```
x = x.toarray()
y = y.toarray()
```

[47]:
```
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size = 0.2 , random_state = 42)
```

Then i turned the x and y into arrays to make them suitable to apply the encoder on them , and after encoding x and y I needed to cast them into arrays again .

Then I divided the data into x_train , x_test , y_train and y_test using the train_test_split.

[48]:
```
model = GaussianNB()
```

[49]:
```
y_t = []
for i in range(len(y_train)):
        y_t.append(y_train[i][0])
```

[50]:
```
y_s = []
for i in range(len(y_test)):
    y_s.append(y_test[i][0])
```

[51]:
```
y_t = np.array(y_t)
```

[52]:
```
print(x_train.shape[0] , y_train.shape[0])
```

```
26048 26048
```

Now i needed to make the y_train and y_test a 1D array , so i only took the first element of each row as the encoded column is a 2D array of the shape [0 , 1] or [1 , 0] so getting the first element of each row was enough for the model .

And finally i checked the size of the x_train and y_train variables to make sure that there is no inconsistency in the data.

[51]:
```
y_t = np.array(y_t)
```

[52]:
```
print(x_train.shape[0] , y_train.shape[0])
```
```
26048 26048
```

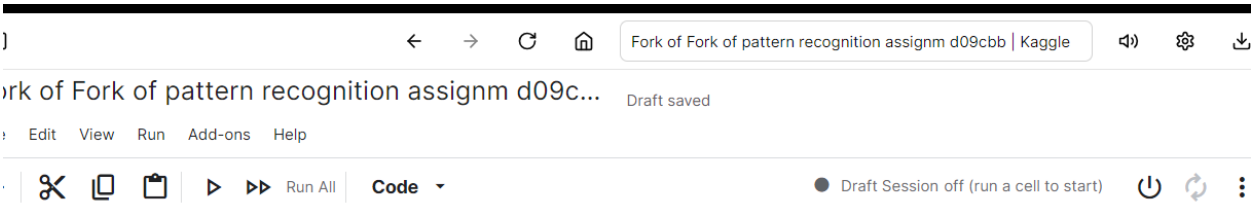
[53]:
```
print(x_train.shape[0] , y_t.shape[0])
```
```
26048 26048
```

[54]:
```
model.fit(x_train , y_t)
```

[54…
```
GaussianNB()
```

[55]:
```
pred = model.predict(x_test)
```

Then i trained the model using the ‘fit’ function, then made predictions on the x_test and saved the predictions in the ‘pred’ variable.



[56]:
```
cm = confusion_matrix(y_s , pred)
```

[57]:
```
TN = cm[0 , 0]
FP = cm[0 , 1]
FN = cm[1 , 0]
TP = cm[1 , 1]
```

Now as I need to get the sensitivity , specificity and the posterior probability of the data , i needed to get the confusion matrix , and then i stored each of its values in a variable.

Accuracy:

```
[27]: accuracy_score(y_s , pred)
```

```
[27… 0.4946253071253071
```

Sensitivity:

```
[28]: recall_score(y_s , pred)
```

```
[28… 0.38375407166123776
```

Specificity:

```
[60]: print(TN / (TN + FP))
```

```
0.835
```

Posterior probability of making over 50k a year:

```
[61]: print(TP /(TP + FP))
```

```
0.8771521637971149
```

And finally I got the 4 requested metrics using the values of the confusion matrix.