Muhammad Tahir K214503

LAB 04

1. Initially perform the following three exercises on the following Links:

```
https://www.analyticsvidhya.com/blog/2021/01/a-guide-to-the-naive-bayes-algorithm/
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [1, 2, 3]].values
y = dataset.iloc[:, -1].values

print(X)
```

```
[['Male' 19 19000]
 ['Male' 35 20000]
 ['Female' 26 43000]
 ...
 ['Female' 50 20000]
 ['Male' 36 33000]
 ['Female' 49 36000]]
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
▾ GaussianNB ⓘ ?
GaussianNB()
```

```
y_pred  =  classifier.predict(X_test)
```

```
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1])
```

```
y_test
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test,y_pred)
```

```
cm
```

```
array([[56,  2],
       [ 4, 18]])
```

```
ac
```

```
0.925
```

```
https://www.statology.org/k-fold-cross-validation-in-python/
```

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from numpy import mean
from numpy import absolute
from numpy import sqrt
import pandas as pd
```

```
df = pd.DataFrame({'y': [6, 8, 12, 14, 14, 15, 17, 22, 24, 23],
                   'x1': [2, 5, 4, 3, 4, 6, 7, 5, 8, 9],
                   'x2': [14, 12, 12, 13, 7, 8, 7, 4, 6, 5]})
```

```
#define predictor and response variables
X = df[['x1', 'x2']]
y = df['y']

#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)

#build multiple linear regression model
model = LinearRegression()

#use k-fold CV to evaluate model
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)

#view mean absolute error
mean(absolute(scores))
```

```
3.146154808346972
```

```
#define predictor and response variables
X = df[['x1', 'x2']]
y = df['y']

#define cross-validation method to use
cv = KFold(n_splits=5, random_state=1, shuffle=True)

#build multiple linear regression model
model = LinearRegression()
```

```
#use LOOCV to evaluate model
scores = cross_val_score(model, X, y, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)

#view RMSE
sqrt(mean(absolute(scores)))
```

```
4.284373111711817
```

```
https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn
```

```
# Assigning features and label variables
Weather = ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny',
'Rainy','Sunny','Overcast','Overcast','Rainy']
Temp = ['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

Play =['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
```

```
# Import LabelEncoder
from sklearn import preprocessing

#creating labelEncoder
le = preprocessing.LabelEncoder()

# Converting string labels into numbers.
weather_encoded = le.fit_transform(Weather)
print(weather_encoded)
```

```
[2 2 0 1 1 1 0 2 2 1 2 0 0 1]
```

```
# Converting string labels into numbers
temp_encoded = le.fit_transform(Temp)
label = le.fit_transform(Play)

print("Temp:", temp_encoded)
print("Play :", label)
```

```
Temp: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
Play : [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

```
# Combinig weather and temp into single listof tuples
features = [tup for tup in zip(weather_encoded, temp_encoded)]
print(features)
```

```
[(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]
```

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
model = GaussianNB()

# Train the model using the training sets
model.fit(features,label)

#Predict Output
predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
print("Predicted Value:", predicted)
```

```
Predicted Value: [1]
```

2. In the following step, you will be working on building your own Naïve Bayes Classifier.

```
data = {
    'Tid': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Refund': ['Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'No'],
    'Marital Status': ['Single', 'Married', 'Single', 'Married', 'Divorced', 'Married', 'Divorced', 'Single', 'Married'
    'Taxable Income': ['125K', '100K', '70K', '120K', '95K', '60K', '220K', '85K', '75K', '90K'],
    'Evade': ['No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes']
}

df = pd.DataFrame(data)

df['Taxable Income'] = df['Taxable Income'].str.replace('K', '').astype('int64')
df['Income'] = pd.cut(df['Taxable Income'], bins=[0, 80, 150, np.inf], labels=['Low', 'Medium', 'High'])

df.drop('Taxable Income', axis=1, inplace=True)
```

```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()

df['Refund'] = le.fit_transform(df['Refund'])
df['Marital Status'] = le.fit_transform(df['Marital Status'])
df['Income'] = le.fit_transform(df['Income'])
df['Evade'] = le.fit_transform(df['Evade'])

df
```

	Tid	Refund	Marital Status	Evade	Income
0	1	1	2	0	2
1	2	0	1	0	2
2	3	0	2	0	1
3	4	1	1	0	2
4	5	0	0	1	2
5	6	0	1	0	1
6	7	1	0	1	0
7	8	0	2	1	2
8	9	0	1	0	1
9	10	0	2	1	2

Next steps: Generate code with df   View recommended plots   New interactive sheet

```
X = df[['Refund', 'Marital Status', 'Income']]
y = df['Evade']
```

```
class NaiveBayesClassifier:
    def fit(self, X, y):
        n_samples, n_features = X.shape
        print("No. of samples: ", n_samples,"\nNo. of features: ", n_features)
        self.classes = np.unique(y)
        n_classes = len(self.classes)
        print("No. of classes: ", n_classes)

        # Calculate priors
        self.priors = np.zeros(n_classes)
        for idx, c in enumerate(self.classes):
            self.priors[idx] = np.sum(y == c) / n_samples

        print("\nprior probability of NO: ", self.priors[0])
        print("prior probability of YES: ", self.priors[1])

        # Likelihoods with Laplace Smoothing
        self.likelihoods = {}
        for feature_idx in range(n_features):
            self.likelihoods[feature_idx] = {}
            for idx, c in enumerate(self.classes):
                X_c = X[y == c]
                feature_values = np.unique(X[:, feature_idx])
                self.likelihoods[feature_idx][c] = {}
                for value in feature_values:
                    self.likelihoods[feature_idx][c][value] = (np.sum(X_c[:, feature_idx] == value) + 1) / (len(X_c) + 

    def predict(self, X):
        y_pred = []
        for x in X:
            posteriors = []
            for idx, c in enumerate(self.classes):
                prior = np.log(self.priors[idx])
                likelihood = np.sum([np.log(self.likelihoods[feature_idx][c].get(x[feature_idx], 1e-6)) for feature_idx
                posterior = prior + likelihood
                posteriors.append(posterior)
            y_pred.append(self.classes[np.argmax(posteriors)])
        return np.array(y_pred)
```

```
classifier = NaiveBayesClassifier()
classifier.fit(X.values, y.values)
```

```
No. of samples:  10
No. of features:  3
No. of classes:  2

prior probability of NO:  0.6
prior probability of YES:  0.4
```

```
predictions = classifier.predict(X.values)
df['Predicted Evade'] = predictions

df[['Tid', 'Evade', 'Predicted Evade']]
```

	Tid	Evade	Predicted Evade
0	1	0	0
1	2	0	0
2	3	0	0
3	4	0	0
4	5	1	1
5	6	0	0
6	7	1	1
7	8	1	1
8	9	0	0
9	10	1	1

```
X_test = np.array([
    [1, 0, 1],  # X1: Refund = No, Status = Married, Income = Low (60K)
    [0, 1, 2]   # X2: Refund = Yes, Status = Divorced, Income = Medium (90K)
])

y_test_pred = classifier.predict(X_test)
print("Class \nNO : 0, YES : 1")
print("Refund = No, Status = Married, Income = Low (60K), Predicted Class: ", y_test_pred[0])
print("Refund = Yes, Status = Divorced, Income = Medium (90K), Predicted Class: ", y_test_pred[1])
```

```
Class
NO : 0, YES : 1
Refund = No, Status = Married, Income = Low (60K), Predicted Class:  0
Refund = Yes, Status = Divorced, Income = Medium (90K), Predicted Class:  0
```

.

.