Assignment 6 Data Analytics III 1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset. II. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Step 1: Importing the Libraries #As always, the first step will always include importing the libraries which are the NumPy, Pandas and the Matplotlib.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Step 2: Importing the dataset In this step, we shall import the Iris Flower dataset which is stored in my github repository as IrisDataset.csv and save it to the variable dataset.

In [32]:
```
dataset = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Classification/master/IrisDataset.csv')
```

In [19]:
```
dataset.head()
```

Out[19]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [22]:
```
gk=dataset.groupby('species')
```

In [23]:
```
gk.first()
```

Out[23]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.1	3.5	1.4	0.2
versicolor	7.0	3.2	4.7	1.4
virginica	6.3	3.3	6.0	2.5

Step 3: After this, we assign the 4 independent variables to X and the dependent variable 'species' to Y. The first 5 rows of the dataset are displayed.

In [24]:
```
X = dataset.iloc[:,:4].values
y = dataset['species'].values
```

Step 4: Splitting the dataset into the Training set and Test set Once we have obtained our data set, we have to split the data into the training set and the test set. In this data set, there are 150 rows with 50 rows of each of the 3 classes. As each class is given in a continuous order, we need to randomly split the dataset. Here, we have the test_size=0.2, which means that 20% of the dataset will be used for testing purpose as the test set and the remaining 80% will be used as the training set for training the Naive Bayes classification model.

In [25]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Step 5: Feature Scaling The dataset is scaled down to a smaller range using the Feature Scaling option. In this, both the X_train and X_test values are scaled down to smaller values to improve the speed of the program.

In [26]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Step 6: Training the Naive Bayes Classification model on the Training Set In this step, we introduce the class GaussianNB that is used from the sklearn.naive_bayes library. Here, we have used a Gaussian model, there are several other models such as Bernoulli, Categorical and Multinomial. Here, we assign the GaussianNB class to the variable classifier and fit the X_train and y_train values to it for training purpose.

In [27]:
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[27]:
```
GaussianNB(priors=None)
```

Step 7: Predicting the Test set results Once the model is trained, we use the the classifier.predict() to predict the values for the Test set and the values predicted are stored to the variable y_pred.

In [28]:
```
y_pred = classifier.predict(X_test)
y_pred
```

Out[28]:
```
array(['versicolor', 'setosa', 'setosa', 'virginica', 'virginica',
       'versicolor', 'setosa', 'versicolor', 'virginica', 'virginica',
       'setosa', 'virginica', 'versicolor', 'virginica', 'virginica',
       'versicolor', 'setosa', 'setosa', 'setosa', 'setosa', 'virginica',
       'versicolor', 'versicolor', 'setosa', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'setosa'], dtype='<U10')
```

Step 8: Confusion Matrix and Accuracy This is a step that is mostly used in classification techniques. In this, we see the Accuracy of the trained model and plot the confusion matrix. The confusion matrix is a table that is used to show the number of correct and incorrect predictions on a classification problem when the real values of the Test Set are known. It is of the format

In [29]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
cm
```

```
Accuracy :  1.0
```

Out[29]:
```
array([[10,  0,  0],
       [ 0,  7,  0],
       [ 0,  0, 13]], dtype=int64)
```

From the above confusion matrix, we infer that, out of 30 test set data, 30 were correctly classified . This gives us a high accuracy of 100%

Step 9: Comparing the Real Values with Predicted Values In this step, a Pandas DataFrame is created to compare the classified values of both the original Test set (y_test) and the predicted results (y_pred).

In [30]:
```
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
df
```

Out[30]:

	Real Values	Predicted Values
0	versicolor	versicolor
1	setosa	setosa
2	setosa	setosa
3	virginica	virginica
4	virginica	virginica
5	versicolor	versicolor
6	setosa	setosa
7	versicolor	versicolor
8	virginica	virginica
9	virginica	virginica
10	setosa	setosa
11	virginica	virginica
12	versicolor	versicolor
13	virginica	virginica
14	virginica	virginica
15	versicolor	versicolor
16	setosa	setosa
17	setosa	setosa
18	setosa	setosa
19	setosa	setosa
20	virginica	virginica
21	versicolor	versicolor
22	versicolor	versicolor
23	setosa	setosa
24	virginica	virginica
25	virginica	virginica
26	virginica	virginica
27	virginica	virginica
28	virginica	virginica
29	setosa	setosa

In [ ]: