Page No.
Date

Name :- Rohini J. Devkar
Roll no :- 28272
PRN no :- 72030818G
TE-2.
DSBDA practical.

Practical No. 6.

Aim :- 1) Implement Simple Naive Bayes classification algorithm using python/R on iris.csv dataset.
2) Compute confusion matrix to find TP, FP, TN, FN, Accuracy, Error Rate, Precision, Recall on the given dataset.

Theory :-

* Naive Bayes classification :-

The Naive Bayes classification algorithm is a probabilistic classifier, and it belongs to Supervised learning. It is based on probability models that incorporate strong independence assumptions. The independence assumptions often do not have an impact on reality. Therefore they are considered naive.

* The types of Naive Bayes classification :-

1) Multinomial Naive Bayes :-

It is a common Bayesian learning approach in a natural language processing. Using the Bayes theorem, the program estimates the tag of a text, such as an email or a newspaper piece.

2) The Bernoulli Naive Bayes :-

It is a part of the family of naive bayes. It only takes binary values. There may be multiple features, but each is assumed to be a

binary-valued variable. Therefore, this class requires samples to be represented as binary-valued feature vectors.

3) Gaussian Naive Bayes :-

It is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. To build a simple model using Gaussian Naive Bayes, we assume that the data is characterized by a Gaussian distribution with no covariance between the parameters.

* Implementation of Naive Bayes algorithm :-

1) Data pre-processing step
2) Fitting Naive Bayes to the training set.
3) Predicting the test result.
4) Test accuracy of the result (creation of confusion matrix).
5) Visualizing the test set result.

* Confusion matrix of Naive Bayes classification :-

The confusion matrix is a two by two table that contains four outcomes produced by a binary classifier.

Confusion matrix		Predicted	
		Positive	Negative
Observed	Positive	TP	FN
	Negative	FP	TN

- Also, it is useful to calculate accuracy, error rate, precision and recall.
- The above table has given following cases :-
  - True Negative :- correct positive prediction
  - False positive :- incorrect positive prediction.

- True Negative :- correct Negative prediction.
- False Negative :- incorrect negative prediction

- We can perform various calculations for the model, such as model's accuracy, using this matrix. These calculations are given below :-

① Accuracy :- - It is calculated as the number of all correct predictions divided by the total number of the dataset.
- The best accuracy is 1.0, whereas the worst is 0.0.

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} = \frac{TP + TN}{P + N}$$

② Error Rate :- - Error rate is calculated as the number of all incorrect predictions divided by the total number of the dataset.
- The best error rate is 0.0, whereas the worst is 1.0.

$$ERR = \frac{FP + FN}{TP + TN + FN + FP} = \frac{FP + FN}{P + N}$$

③ Precision :- - Precision is calculated as the number of correct positive predictions divided by the total number of positive predictions.
- It is also called positive predictive value (PPV).
- The best precision is 1.0, whereas the worst is 0.0.

$$PREC = \frac{TP}{TP + FP}.$$

④ Recall :- - It is calculated as the number of correct positive predictions divided by the total number of positives or positive observations.
- It is also called as sensitivity or True positive rate.
- The best recall is 1.0, whereas the worst is 0.0.

$$SN = \frac{TP}{TP + FN} = \frac{TP}{P}$$

Page No.
Date

* In this practical, we use iris.csv dataset for naive Bayes classification.

Now,

We know,

Confusion Matrix		Predicted	
		+ve	-ve
Observed	+ve	10	1
	-ve	1	18

TP = 10, FN = 1, FP = 1, TN = 18

then,

1) $ACC = \frac{TP+TN}{P+N} = \frac{10+18}{30} = 0.9333333$

2) $ERR = \frac{FP+FN}{P+N} = \frac{1+1}{30} = 0.06666667$

3) $PREC = \frac{TP}{TP+FP} = \frac{10}{10+1} = 0.909090909$

4) $Recall = \frac{TP}{TP+FN} = \frac{10}{10+1} = 0.909090909$

* Conclusion :-

In the practical, the accuracy, error rate, precision and recall are the best values for iris.csv dataset.

Data Science And Big Data Analytics

Name:- Rohini Janardan Devkar

PRN NO:- 72030818G

Roll no:- 23272

Class :- TE 2(COMP)

Problem Statement:-

Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.

Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

In [2]:
```
# Naive Bayes Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
```

In [3]:
```
# Importing the dataset
dataset = pd.read_csv('iris.csv')
```

In [4]:
```
#Looking at the first 5 values of the dataset
dataset.head()
```

Out[4]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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 [5]:
```
%matplotlib inline
img=mpimg.imread('iris_types.jpg')
plt.figure(figsize=(20,40))
plt.axis('off')
plt.imshow(img)
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x234fdea9eb0>
```

In [6]:
```
#Spliting the dataset in independent and dependent variables
X = dataset.iloc[:,:4].values
y = dataset['variety'].values
```

In [7]:
```
# Splitting the dataset into the Training set and Test set
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_stat
```

In [8]:
```
# Feature Scaling to bring the variable in a single scale
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [9]:
```
# Fitting Naive Bayes Classification to the Training set with linear kernel
from sklearn.naive_bayes import GaussianNB
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, y_train)
```

Out[9]:
```
GaussianNB()
```

In [10]:
```
# Predicting the Test set results
y_pred = nvclassifier.predict(X_test)
print(y_pred)
```

```
['Virginica' 'Virginica' 'Setosa' 'Setosa' 'Setosa' 'Virginica'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Virginica' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Virginica' 'Versicolor'
 'Setosa' 'Versicolor' 'Setosa' 'Virginica' 'Setosa' 'Virginica'
 'Virginica' 'Versicolor' 'Virginica' 'Setosa' 'Virginica' 'Versicolor']
```

In [11]:
```
#lets see the actual and predicted value side by side
y_compare = np.vstack((y_test,y_pred)).T
#actual value on the left side and predicted value on the right hand side
#printing the top 5 values
y_compare[:5,:]
```

Out[11]:
```
array([['Virginica', 'Virginica'],
       ['Virginica', 'Virginica'],
       ['Setosa', 'Setosa'],
       ['Setosa', 'Setosa'],
       ['Setosa', 'Setosa']], dtype=object)
```

In [12]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[11  0  0]
 [ 0  8  1]
 [ 0  1  9]]
```

In [13]:
```
#finding accuracy from the confusion matrix.
a = cm.shape
corrPred = 0
falsePred = 0

for row in range(a[0]):
    for c in range(a[1]):
        if row == c:
            corrPred +=cm[row,c]
        else:
            falsePred += cm[row,c]
print('Correct predictions: ', corrPred)
print('False predictions:', falsePred)
print ('\n\nAccuracy of the Naive Bayes Clasification is: ', corrPred/(cm.sum()))
print ('\n\nErroRate of the Naive Bayes Clasification is: ', falsePred/(cm.sum()))
```

```
Correct predictions:  28
False predictions: 2


Accuracy of the Naive Bayes Clasification is:  0.9333333333333333


ErroRate of the Naive Bayes Clasification is:  0.06666666666666667
```