Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exe24 - Naive Bayes Classification Exercise

Name: Chong Mun Chen

IC Number: 960327-07-5097

Date : 24/7/2023

Introduction : Practising on this exercise using Naive Bayes classifier algorithm.

Conclusion : Succeeded in training the data with the Naive Bayes classifier algorithm and achieving a decent accuracy score for the prediction model.

Naive Bayes exercise

Naive Bayes classification walkthrough

In [1]:
```
#Import scikit-learn dataset library
import sklearn as sk
from sklearn import datasets

#Load dataset
wine = datasets.load_wine()
```

In [2]:
```
# print the names of the 13 features
print(wine.feature_names)

# print the label type of wine(class_0, class_1, class_2)
# print(wine.keys())
print(wine.target_names)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total
_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'colo
r_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
['class_0' 'class_1' 'class_2']
```

In [3]:
```
# print data(feature)shape
import pandas as pd
import numpy as np

df = pd.DataFrame(data = wine.data, columns = wine.feature_names)
df.shape
```

Out[3]: (178, 13)

In [4]:
```
# print the wine data features (top 5 records)
df.head()
```

Out[4]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonfla
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

In [5]:
```
# print the wine labels (0:Class_0, 1:class_2, 2:class_2)
print(wine.target_names)
```

```
['class_0' 'class_1' 'class_2']
```

In [6]:
```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X = df
y = wine.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [7]:
```
#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(X_train, y_train);

#Predict the response for test dataset
y_pred = model.predict(X_test)
y_pred
```

Out[7]:
```
array([0, 0, 2, 0, 1, 0, 1, 2, 1, 2, 0, 2, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 2, 2, 2, 1, 1, 1, 0, 0, 1, 2, 0, 0, 0])
```

In [8]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn.metrics import accuracy_score

# Model Accuracy, how often is the classifier correct?
accuracy_score(y_test, y_pred)
```

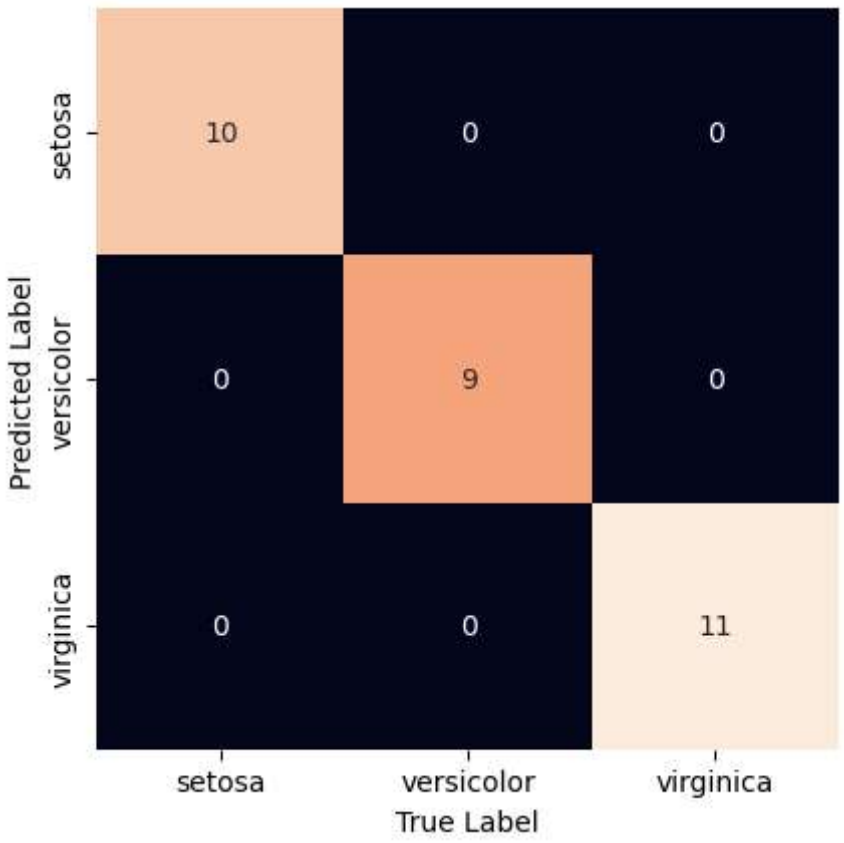
Out[8]:
```
1.0
```

Exercise 1 : Perform NB classification using the Iris dataset

In [9]:

```
## Exercise 1 : Perform NB classification using the iris dataset

# Load libraries
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import seaborn as sns

# Load iris dataset
iris = datasets.load_iris()

# Create feature matrix
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mat = confusion_matrix(y_test, y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('True Label')
plt.ylabel('Predicted Label');

# Create target vector
print(y)

# View the first observation's feature values
print(X[0])
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2
 2 2]
[5.1 3.5 1.4 0.2]
```



Exercise 2 : Perform NB classification using the Titanic dataset

In [10]:
```
# Exercise 2 : Perform NB classification using the Titanic dataset

# Load libraries
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

# Load iris dataset
titanic_df = pd.read_csv(r'../data_samples2/titanic.csv')

# Create feature matrix
reset = ({'female': 0, 'male': 1})
titanic_df['Sex'] = titanic_df['Sex'].replace(reset)

X = titanic_df.drop(['Survived', 'Name'], axis=1)
y = titanic_df['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy_score(y_test, y_pred)
```

Out[10]:
```
0.7359550561797753
```