Forward
School

## Program Code: J620-002-4:2020

## Program Name: FRONT-END SOFTWARE DEVELOPMENT

## Title : Exe24 - Naive Bayes Classification Exercise

#### Name: Ooi Caaron

#### IC Number:990701-07-5837

#### Date :29/7/23

#### Introduction :

#### Conclusion :

Naive Bayes exercise

Naive Bayes classification walkthrough

In [4]:

```
#Import scikit-learn dataset library
import sklearn as sk
from sklearn import datasets
import pandas as pd
#Load dataset
wine = datasets.load_wine()
data = pd.DataFrame(wine.data, columns=wine.feature_names)
data
```

Out[4]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonf
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	
...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	

178 rows × 13 columns

In [5]:

```
# print the names of the 13 features
data.keys()
# print the label type of wine(class_0, class_1, class_2)
wine_labels = wine.target_names
print(wine_labels)
```

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

In [6]:

```
# print data(feature)shape
data.shape
```

Out[6]:

```
(178, 13)
```

In [7]:

```
# print the wine data features (top 5 records)
data.head()
```

Out[7]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
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 [8]:

```
# print the wine labels (0:Class_0, 1:class_2, 2:class_2)
labels_with_names = [(i, wine.target_names[label]) for i, label in enumerate(wine.target
print(labels_with_names)
```

```
[(0, 'class_0'), (1, 'class_0'), (2, 'class_0'), (3, 'class_0'), (4, 'clas
s_0'), (5, 'class_0'), (6, 'class_0'), (7, 'class_0'), (8, 'class_0'), (9,
'class_0'), (10, 'class_0'), (11, 'class_0'), (12, 'class_0'), (13, 'class
_0'), (14, 'class_0'), (15, 'class_0'), (16, 'class_0'), (17, 'class_0'),
(18, 'class_0'), (19, 'class_0'), (20, 'class_0'), (21, 'class_0'), (22,
'class_0'), (23, 'class_0'), (24, 'class_0'), (25, 'class_0'), (26, 'class
_0'), (27, 'class_0'), (28, 'class_0'), (29, 'class_0'), (30, 'class_0'),
(31, 'class_0'), (32, 'class_0'), (33, 'class_0'), (34, 'class_0'), (35,
'class_0'), (36, 'class_0'), (37, 'class_0'), (38, 'class_0'), (39, 'class
_0'), (40, 'class_0'), (41, 'class_0'), (42, 'class_0'), (43, 'class_0'),
(44, 'class_0'), (45, 'class_0'), (46, 'class_0'), (47, 'class_0'), (48,
'class_0'), (49, 'class_0'), (50, 'class_0'), (51, 'class_0'), (52, 'class
_0'), (53, 'class_0'), (54, 'class_0'), (55, 'class_0'), (56, 'class_0'),
(57, 'class_0'), (58, 'class_0'), (59, 'class_1'), (60, 'class_1'), (61,
'class_1'), (62, 'class_1'), (63, 'class_1'), (64, 'class_1'), (65, 'class
_1'), (66, 'class_1'), (67, 'class_1'), (68, 'class_1'), (69, 'class_1'),
(70, 'class_1'), (71, 'class_1'), (72, 'class_1'), (73, 'class_1'), (74,
'class_1'), (75, 'class_1'), (76, 'class_1'), (77, 'class_1'), (78, 'class
_1'), (79, 'class_1'), (80, 'class_1'), (81, 'class_1'), (82, 'class_1'),
(83, 'class_1'), (84, 'class_1'), (85, 'class_1'), (86, 'class_1'), (87,
'class_1'), (88, 'class_1'), (89, 'class_1'), (90, 'class_1'), (91, 'class
_1'), (92, 'class_1'), (93, 'class_1'), (94, 'class_1'), (95, 'class_1'),
(96, 'class_1'), (97, 'class_1'), (98, 'class_1'), (99, 'class_1'), (100,
'class_1'), (101, 'class_1'), (102, 'class_1'), (103, 'class_1'), (104, 'c
lass_1'), (105, 'class_1'), (106, 'class_1'), (107, 'class_1'), (108, 'cla
ss_1'), (109, 'class_1'), (110, 'class_1'), (111, 'class_1'), (112, 'class
_1'), (113, 'class_1'), (114, 'class_1'), (115, 'class_1'), (116, 'class_
1'), (117, 'class_1'), (118, 'class_1'), (119, 'class_1'), (120, 'class_
1'), (121, 'class_1'), (122, 'class_1'), (123, 'class_1'), (124, 'class_
1'), (125, 'class_1'), (126, 'class_1'), (127, 'class_1'), (128, 'class_
1'), (129, 'class_1'), (130, 'class_2'), (131, 'class_2'), (132, 'class_
2'), (133, 'class_2'), (134, 'class_2'), (135, 'class_2'), (136, 'class_
2'), (137, 'class_2'), (138, 'class_2'), (139, 'class_2'), (140, 'class_
2'), (141, 'class_2'), (142, 'class_2'), (143, 'class_2'), (144, 'class_
2'), (145, 'class_2'), (146, 'class_2'), (147, 'class_2'), (148, 'class_
2'), (149, 'class_2'), (150, 'class_2'), (151, 'class_2'), (152, 'class_
2'), (153, 'class_2'), (154, 'class_2'), (155, 'class_2'), (156, 'class_
2'), (157, 'class_2'), (158, 'class_2'), (159, 'class_2'), (160, 'class_
2'), (161, 'class_2'), (162, 'class_2'), (163, 'class_2'), (164, 'class_
2'), (165, 'class_2'), (166, 'class_2'), (167, 'class_2'), (168, 'class_
2'), (169, 'class_2'), (170, 'class_2'), (171, 'class_2'), (172, 'class_
2'), (173, 'class_2'), (174, 'class_2'), (175, 'class_2'), (176, 'class_
2'), (177, 'class_2')]
```

In [9]:

```
# Import train_test_split function
from sklearn.model_selection import train_test_split
X = wine.data
y = wine.target
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [10]:

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

#Create a Gaussian Classifier
gnb = GaussianNB()
#Train the model using the training sets
gnb.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = gnb.predict(X_test)
```

In [11]:

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

Exercise 1 : Perform NB classification using the Iris dataset

In [12]:

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

# Load libraries
from sklearn import datasets
import matplotlib.pyplot as plt

# Load iris dataset
iris = datasets.load_iris()

# Create feature matrix
X = iris.data
print("Shape of feature matrix (X):", X.shape)

# Create target vector
y = iris.target
print("Shape of target vector (y):", y.shape)

# View the first observation's feature values
first_observation_features = X[0]

print("Feature values for the first observation:")
print(first_observation_features)
```

```
Shape of feature matrix (X): (150, 4)
Shape of target vector (y): (150,)
Feature values for the first observation:
[5.1 3.5 1.4 0.2]
```

Exercise 2 : Perform NB classification using the Titanic dataset

In [13]:

```
data = pd.read_csv("./titanic.csv")
data.head()
reset = {"male": 0, "female": 1,}
data = data.replace({"Sex": reset})
X = data.drop(['Survived', 'Name'], axis=1)
y = data['Survived']
data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
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)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7640449438202247
```

In [ ]: