Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exe24 - Naive Bayes Classification Exercise

Name: Chuay Xiang Ze

IC Number: 021224070255

Date : 24/07/2023

Introduction : Learning about Naive Bayes Classification

Conclusion : Managed to complete tasks relating to the topic.

Naive Bayes exercise

Naive Bayes classification walkthrough

In [23]:

```
#Import scikit-learn dataset library
import sklearn as sk
from sklearn import datasets

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

Out[23]:

```
{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+0
0,
         1.065e+03],
        [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
         1.050e+03],
        [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
         1.185e+03],
        ...,
        [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
         8.350e+02],
        [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
         8.400e+02],
        [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
         5.600e+02]]),
 'target': array([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, 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, 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]),
 'frame': None,
 'target_names': array(['class_0', 'class_1', 'class_2'], dtype='<U7'),
 'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\n----------------
--------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 178
\n    :Number of Attributes: 13 numeric, predictive attributes and the cla
ss\n    :Attribute Information:\n \t\t- Alcohol\n \t\t- Malic acid\n \t\t-
Ash\n\t\t- Alcalinity of ash  \n \t\t- Magnesium\n\t\t- Total phenols\n \t
\t- Flavanoids\n \t\t- Nonflavanoid phenols\n \t\t- Proanthocyanins\n\t\t-
Color intensity\n \t\t- Hue\n \t\t- OD280/OD315 of diluted wines\n \t\t- P
roline\n\n    - class:\n            - class_0\n            - class_1\n
- class_2\n\t\t\n    :Summary Statistics:\n    \n    =====================
======== ==== ===== ======= =====\n                                   Min
Max   Mean     SD\n    ============================= ==== ===== ======= =
====\n    Alcohol:                      11.0  14.8    13.0   0.8\n    Malic
Acid:                   0.74  5.80    2.34  1.12\n    Ash:
1.36  3.23    2.36  0.27\n    Alcalinity of Ash:            10.6  30.0
19.5   3.3\n    Magnesium:                    70.0 162.0    99.7  14.3\n
Total Phenols:                0.98  3.88    2.29  0.63\n    Flavanoids:
0.34  5.08    2.03  1.00\n    Nonflavanoid Phenols:         0.13  0.66
0.36  0.12\n    Proanthocyanins:              0.41  3.58    1.59  0.57\n
Colour Intensity:              1.3  13.0     5.1   2.3\n    Hue:
0.48  1.71    0.96  0.23\n    OD280/OD315 of diluted wines: 1.27  4.00
2.61  0.71\n    Proline:                       278  1680     746   315\n
============================= ==== ===== ======= =====\n\n    :Missing Att
ribute Values: None\n    :Class Distribution: class_0 (59), class_1 (71),
class_2 (48)\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MAR
SHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThis is a copy of UCI
ML Wine recognition datasets.\nhttps://archive.ics.uci.edu/ml/machine-lear
ning-databases/wine/wine.data\n\nThe data is the results of a chemical ana
lysis of wines grown in the same\nregion in Italy by three different culti
vators. There are thirteen different\nmeasurements taken for different con
stituents found in the three types of\nwine.\n\nOriginal Owners: \n\nForin
a, M. et al, PARVUS - \nAn Extendible Package for Data Exploration, Classi
fication and Correlation. \nInstitute of Pharmaceutical and Food Analysis
and Technologies,\nVia Brigata Salerno, 16147 Genoa, Italy.\n\nCitation:\n
\nLichman, M. (2013). UCI Machine Learning Repository\n[https://archive.ic
s.uci.edu/ml]. Irvine, CA: University of California,\nSchool of Informatio
```

In [7]:

```
# print the names of the 13 features
print("Features:", wine.feature_names)
print()
# print the label type of wine(class_0, class_1, class_2)
print("Label:", wine.target_names)
```

```
Features: ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesiu
m', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanin
s', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']

Label: ['class_0' 'class_1' 'class_2']
```

In [13]:

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

```
(178, 13)
```

```
n and Computer Science. \n\n.. topic:: References\n\n  (1) S. Aeberhard,
D. Coomans and O. de Vel, \n  Comparison of Classifiers in High Dimensiona
l Settings, \n  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science an
d Dept. of  \n  Mathematics and Statistics, James Cook University of North
Queensland. \n  (Also submitted to Technometrics). \n\n  The data was used
with many others for comparing various \n  classifiers. The classes are se
parable, though only RDA \n  has achieved 100% correct classification. \n
(RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data)) \n  (Al
l results using the leave-one-out technique) \n\n  (2) S. Aeberhard, D. Co
omans and O. de Vel, \n  "THE CLASSIFICATION PERFORMANCE OF RDA" \n  Tech.
Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of \n  Mathema
tics and Statistics, James Cook University of North Queensland. \n  (Also
submitted to Journal of Chemometrics).\n',
 'feature_names': ['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
  'magnesium',
  'total_phenols',
  'flavanoids',
  'nonflavanoid_phenols',
  'proanthocyanins',
  'color_intensity',
  'hue',
  'od280/od315_of_diluted_wines',
  'proline']}
```

In [16]:

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

```
[[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e+00
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e+00
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]
 [1.437e+01 1.950e+00 2.500e+00 1.680e+01 1.130e+02 3.850e+00 3.490e+00
  2.400e-01 2.180e+00 7.800e+00 8.600e-01 3.450e+00 1.480e+03]
 [1.324e+01 2.590e+00 2.870e+00 2.100e+01 1.180e+02 2.800e+00 2.690e+00
  3.900e-01 1.820e+00 4.320e+00 1.040e+00 2.930e+00 7.350e+02]]
```

In [17]:

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

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

In [34]:

```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X = wine.data
y = wine.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
```

In [35]:

```
#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)
```

In [36]:

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

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

```
Accuracy: 1.0
```

Exercise 1 : Perform NB classification using the Iris dataset

In [48]:

```
## 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

# Create target vector
y = iris.target

# View the first observation's feature values
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: 0.9777777777777777
```

Exercise 2 : Perform NB classification using the Titanic dataset

In [65]:

```
import numpy as np
import pandas as pd

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 [ ]:

```

```