Neural Networks & Deep Learning: ICP5

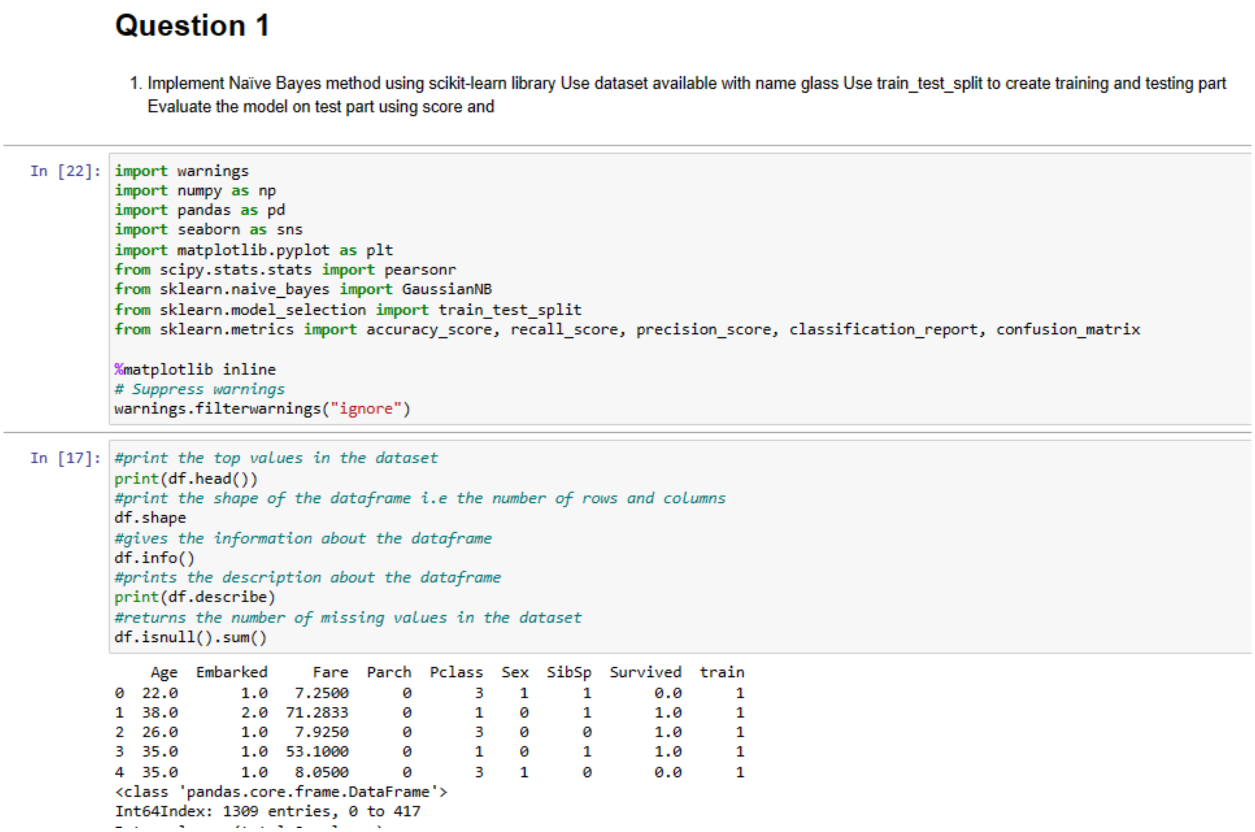
Nirmala Yarlagadda

700733102

GitHub link: https://github.com/niryarjessy22/ICP-5.git

Video link:
https://drive.google.com/file/d/1BLyTkCxDBy9mObO9oZg5WlixzqCEFa82/view?usp=share_link

1. Implement Naïve Bayes method using scikit-learn library
Use dataset available with name **glass**
Use **train_test_split** to create training and testing part
Evaluate the model on **test part** using score and
`classification_report(y_true, y_pred)`



Question 1

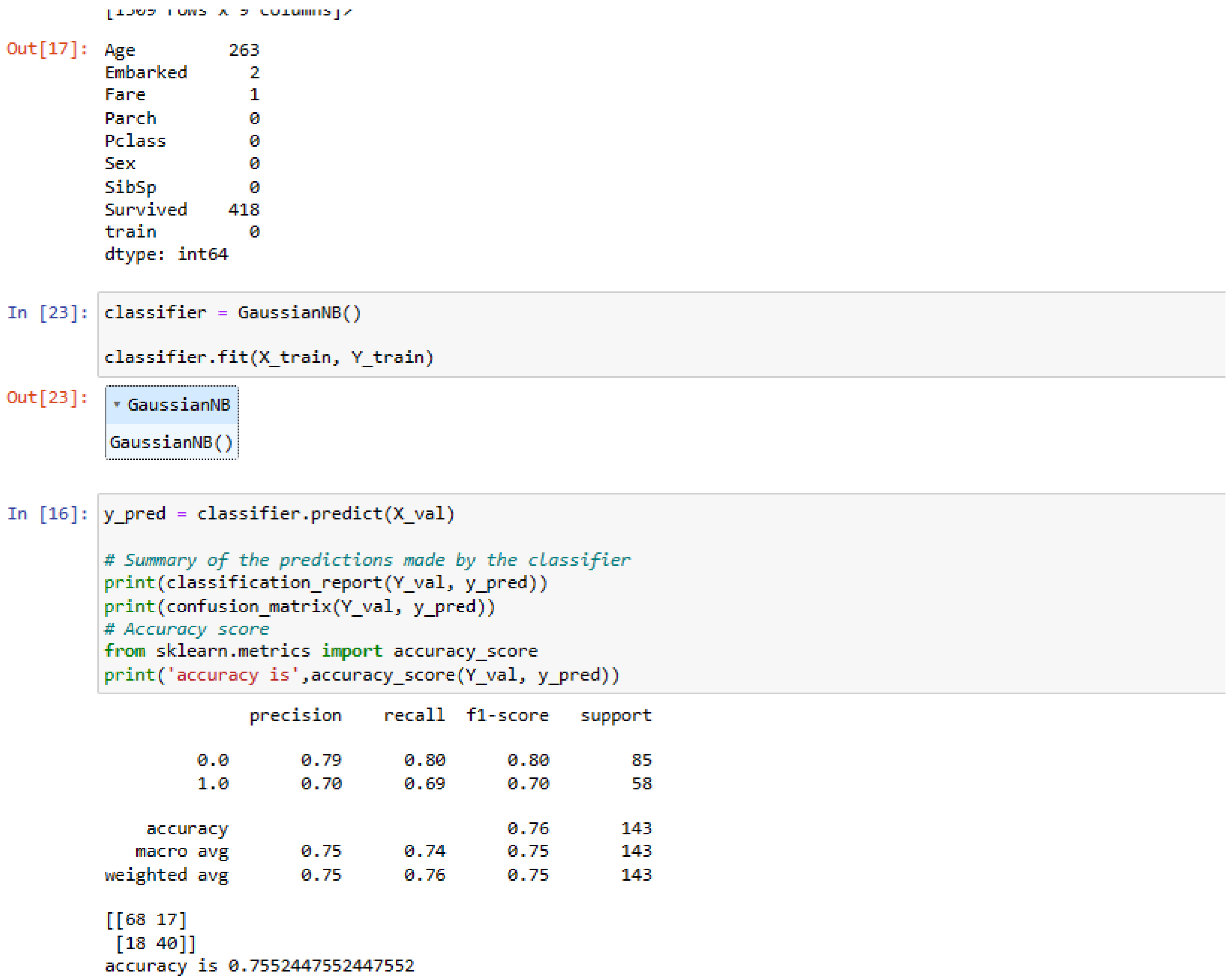
1. Implement Naïve Bayes method using scikit-learn library Use dataset available with name glass Use train_test_split to create training and testing part Evaluate the model on test part using score and

In [22]:
```
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats.stats import pearsonr
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score, classification_report, confusion_matrix

%matplotlib inline
# Suppress warnings
warnings.filterwarnings("ignore")
```

In [17]:
```
#print the top values in the dataset
print(df.head())
#print the shape of the dataframe i.e the number of rows and columns
df.shape
#gives the information about the dataframe
df.info()
#prints the description about the dataframe
print(df.describe)
#returns the number of missing values in the dataset
df.isnull().sum()
```

```
    Age  Embarked     Fare  Parch  Pclass  Sex  SibSp  Survived  train
0  22.0       1.0   7.2500      0       3    1      1       0.0      1
1  38.0       2.0  71.2833      0       1    0      1       1.0      1
2  26.0       1.0   7.9250      0       3    0      0       1.0      1
3  35.0       1.0  53.1000      0       1    0      1       1.0      1
4  35.0       1.0   8.0500      0       3    1      0       0.0      1
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
```



```
[1309 rows x 9 columns]>
```

```
Out[17]: Age          263
         Embarked       2
         Fare           1
         Parch          0
         Pclass         0
         Sex            0
         SibSp          0
         Survived     418
         train          0
         dtype: int64
```

```
In [23]: classifier = GaussianNB()

         classifier.fit(X_train, Y_train)
```

```
Out[23]: ▾ GaussianNB
         GaussianNB()
```

```
In [16]: y_pred = classifier.predict(X_val)

         # Summary of the predictions made by the classifier
         print(classification_report(Y_val, y_pred))
         print(confusion_matrix(Y_val, y_pred))
         # Accuracy score
         from sklearn.metrics import accuracy_score
         print('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.79      0.80      0.80        85
         1.0       0.70      0.69      0.70        58

    accuracy                           0.76       143
   macro avg       0.75      0.74      0.75       143
weighted avg       0.75      0.76      0.75       143

[[68 17]
 [18 40]]
accuracy is 0.7552447552447552
```

2. Implement linear SVM method using scikit library
Use the same dataset above
Use **train_test_split** to create training and testing part
Evaluate the model on **test part** using score and

```
classification_report(y_true, y_pred)
```

Question 2

2. Implement linear SVM method using scikit library Use the same dataset above Use train_test_split to create training and testing part Evaluate the model on test part using score and

In [7]:
```
glass=pd.read_csv("glass.csv")
```

In [8]:
```
glass.head()
```

Out[8]:

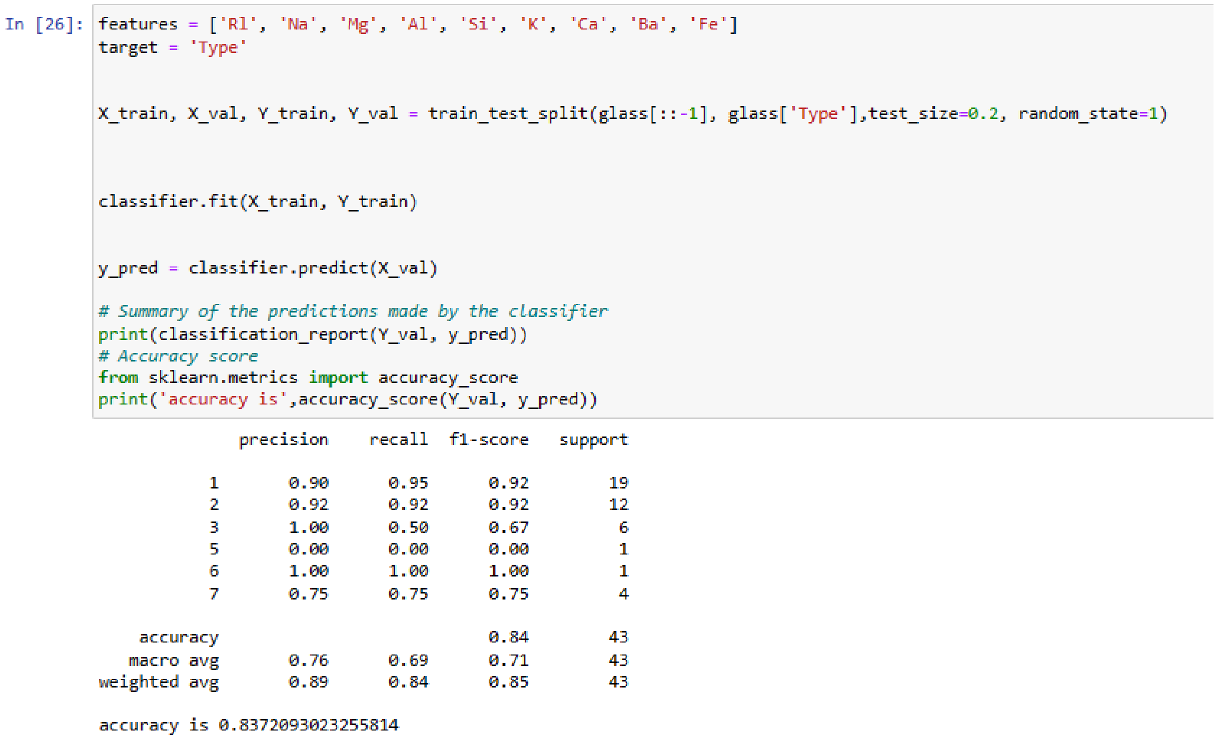
	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

In [26]:
```
features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
target = 'Type'

X_train, X_val, Y_train, Y_val = train_test_split(glass[::-1], glass['Type'],test_size=0.2, random_state=1)


classifier.fit(X_train, Y_train)


y_pred = classifier.predict(X_val)
```



In [26]:
```
features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
target = 'Type'

X_train, X_val, Y_train, Y_val = train_test_split(glass[::-1], glass['Type'],test_size=0.2, random_state=1)



classifier.fit(X_train, Y_train)


y_pred = classifier.predict(X_val)

# Summary of the predictions made by the classifier
print(classification_report(Y_val, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.95      0.92        19
           2       0.92      0.92      0.92        12
           3       1.00      0.50      0.67         6
           5       0.00      0.00      0.00         1
           6       1.00      1.00      1.00         1
           7       0.75      0.75      0.75         4

    accuracy                           0.84        43
   macro avg       0.76      0.69      0.71        43
weighted avg       0.89      0.84      0.85        43

accuracy is 0.8372093023255814
```

Which algorithm you got better accuracy? Can you justify why?

After analyzing results got from training data with Naives Bayes and SVM model, from the above results of accuracy We can say Naives Bayes Algorithm is better than SVM accuracy of Naive Bayes greater accuracy of SVM.

We are not able to predict probabilities of happening type feature with other features with good accurancy but SVM(linear) accuaracy is good when compared to Naives bayes approach because we are able to draw support vectors and margin to predict the data with high accuracy.

The SVM approach performs better than the naive Bayes classifier method in terms of accuracy. This is because the SVM method considers the interactions between the features to some extent and also uses a non-linear kernel, whereas the naive Bayes method treats each feature individually. SVM is better at accurately collecting interactions and calculating scores as a result.

GitHub link: https://github.com/niryarjessy22/ICP-5.git

Video link:
https://drive.google.com/file/d/1BLyTkCxDBy9mObO9oZg5WlixzqCEFa82/view?usp=share_link