Spring 2024: CS5720

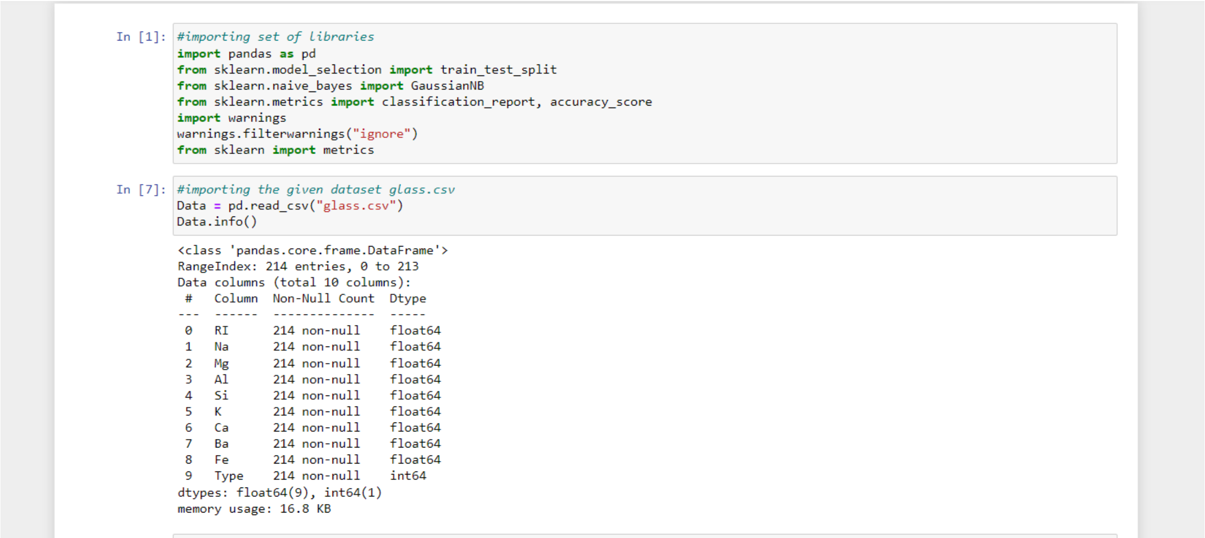
NEURAL NETWORK AND DEEP LEARNING ICP-5

Name: Afroz Mohammad [700758012]

GitHub link: https://github.com/Afrozmohammad19/Assignment5

Video link: https://drive.google.com/file/d/1h8n1MT92gi0CJ71q7DjVuUa_pR-ZLKrx/view?usp=sharing

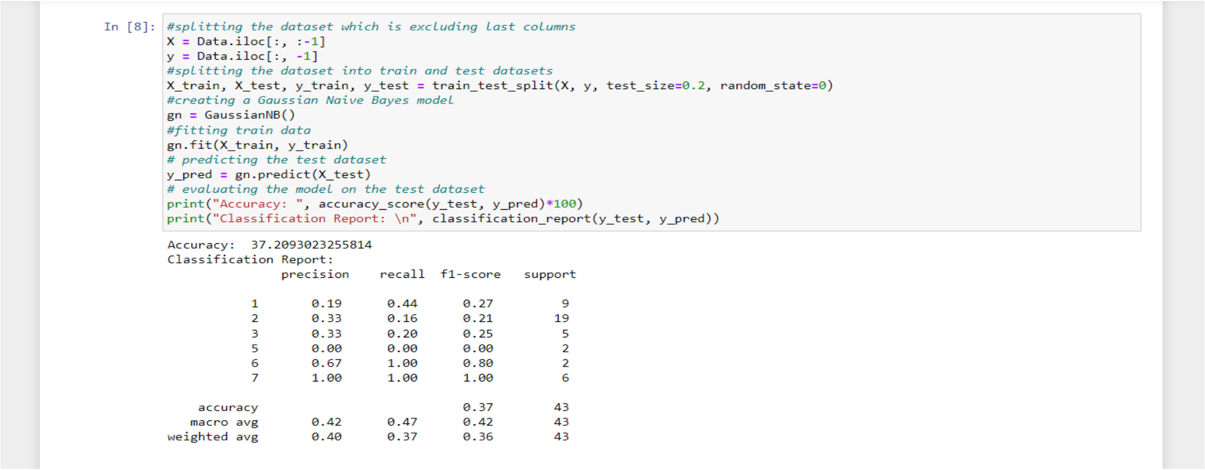
1. Implement Naïve Bayes method using scikit-learn library
   Use dataset available with name glass
   Use train_test_split to create training and testing part
   Evaluate the model on test part using score and classification_report(y_true, y_pred)



```
In [1]: #importing set of libraries
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.naive_bayes import GaussianNB
        from sklearn.metrics import classification_report, accuracy_score
        import warnings
        warnings.filterwarnings("ignore")
        from sklearn import metrics
```

```
In [7]: #importing the given dataset glass.csv
        Data = pd.read_csv("glass.csv")
        Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```



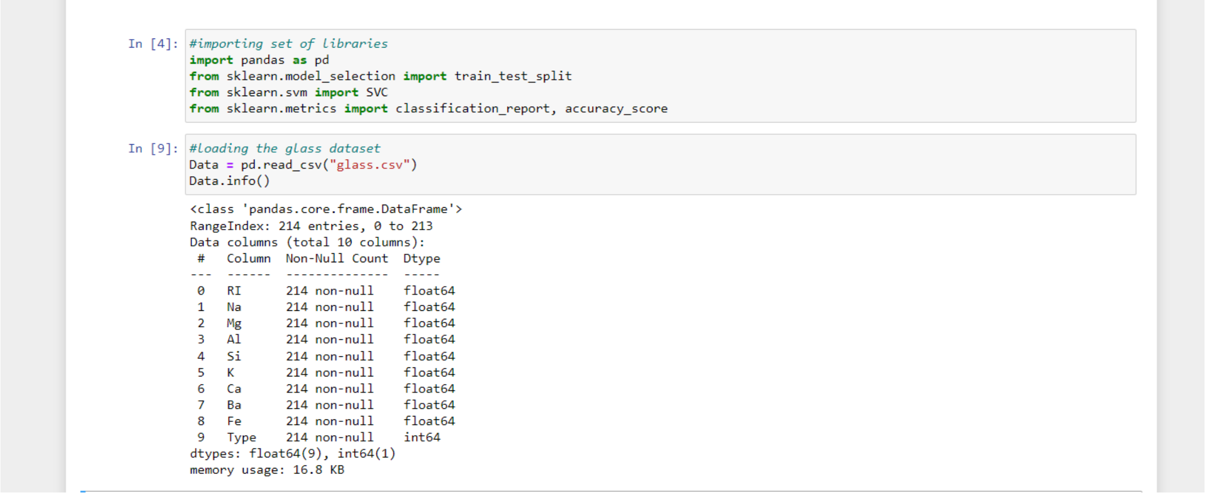
```
In [8]: #splitting the dataset which is excluding last columns
        X = Data.iloc[:, :-1]
        y = Data.iloc[:, -1]
        #splitting the dataset into train and test datasets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
        #creating a Gaussian Naive Bayes model
        gn = GaussianNB()
        #fitting train data
        gn.fit(X_train, y_train)
        # predicting the test dataset
        y_pred = gn.predict(X_test)
        # evaluating the model on the test dataset
        print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
        print("Classification Report: \n", classification_report(y_test, y_pred))
```

```
Accuracy:  37.2093023255814
Classification Report: 
               precision    recall  f1-score   support

           1       0.19      0.44      0.27         9
           2       0.33      0.16      0.21        19
           3       0.33      0.20      0.25         5
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       1.00      1.00      1.00         6

    accuracy                           0.37        43
   macro avg       0.42      0.47      0.42        43
weighted avg       0.40      0.37      0.36        43
```

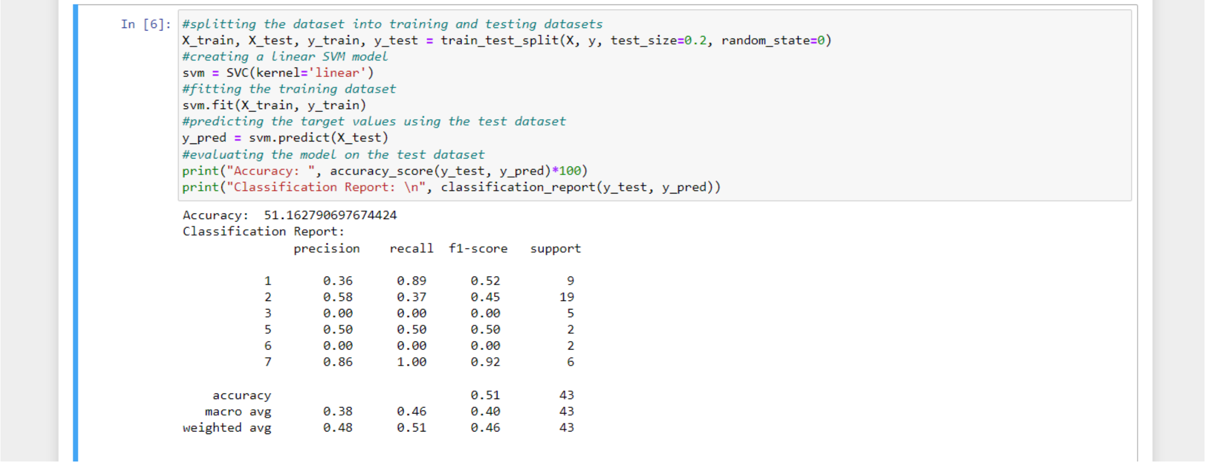
2. Implement linear SVM method using scikit library
   Use the same dataset above
   Use train_test_split to create training and testing part
   Evaluate the model on test part using score and classification_report(y_true, y_pred)



```
In [4]: #importing set of libraries
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.svm import SVC
        from sklearn.metrics import classification_report, accuracy_score

In [9]: #loading the glass dataset
        Data = pd.read_csv("glass.csv")
        Data.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 214 entries, 0 to 213
        Data columns (total 10 columns):
         #   Column  Non-Null Count  Dtype
        ---  ------  --------------  -----
         0   RI      214 non-null    float64
         1   Na      214 non-null    float64
         2   Mg      214 non-null    float64
         3   Al      214 non-null    float64
         4   Si      214 non-null    float64
         5   K       214 non-null    float64
         6   Ca      214 non-null    float64
         7   Ba      214 non-null    float64
         8   Fe      214 non-null    float64
         9   Type    214 non-null    int64
        dtypes: float64(9), int64(1)
        memory usage: 16.8 KB
```



```
In [6]: #splitting the dataset into training and testing datasets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
        #creating a linear SVM model
        svm = SVC(kernel='linear')
        #fitting the training dataset
        svm.fit(X_train, y_train)
        #predicting the target values using the test dataset
        y_pred = svm.predict(X_test)
        #evaluating the model on the test dataset
        print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
        print("Classification Report: \n", classification_report(y_test, y_pred))

        Accuracy:  51.162790697674424
        Classification Report:
                       precision    recall  f1-score   support

                   1       0.36      0.89      0.52         9
                   2       0.58      0.37      0.45        19
                   3       0.00      0.00      0.00         5
                   5       0.50      0.50      0.50         2
                   6       0.00      0.00      0.00         2
                   7       0.86      1.00      0.92         6

            accuracy                           0.51        43
           macro avg       0.38      0.46      0.40        43
        weighted avg       0.48      0.51      0.46        43
```

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

According to the given both cases the SVM having the better accuracy than Naïve Bayes method because it is having high accuracy, the accuracy is depended on the precision and recall of both cases of the program. By this we can say that in both the algorithms SVM is having better accuracy. The Naïve Bayes method is completely deals independently whereas SVM can handle high dimensional data is effective in these cases with limited training samples, and can handle non-linear classification using kernel functions.