Revathi Atchi

700742168

NEURAL NETWORK AND DEEP LEARNING ASSIGNMENT-5

GITHUB LINK: https://github.com/revathiatchi/NeuralAssignment5.git

RECORDINGLINK: https://github.com/revathiatchi/NeuralAssignment5/assets/156601745/cf9ab956-0d5f-44a1-9e46-b1104b3af5e7

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)

```
import pandas as pd
df = pd.read_csv('glass.csv')
df.head()
```

[3] ✓ 0.0s

	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

```
df.info()
```

[4] ✓ 0.0s

```
<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
```

```
df.describe()
```

[5] ✓ 0.1s

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.780374
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.103739
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.000000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

```
df.columns.values
```
[6] ✓ 0.0s

```
array(['RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe', 'Type'],
      dtype=object)
```

```
df['Type'].value_counts()
```
[7] ✓ 0.0s

```
Type
2    76
1    70
7    29
3    17
5    13
6     9
Name: count, dtype: int64
```

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
# Splitting the data using train_test_split for creating train and test data
X = df.drop("Type", axis=1)
Y = df["Type"]

X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.2, random_state=42)
```
[11] ✓ 0.0s

2. Implement linear SVM method using scikit library

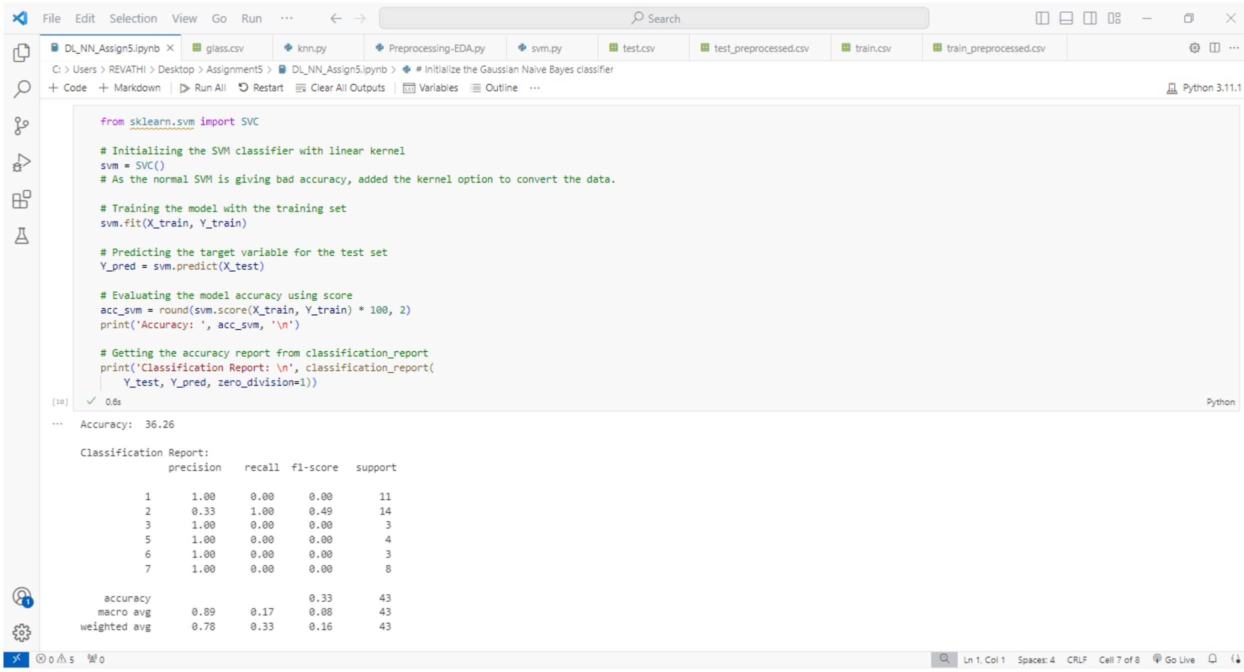
Use the same dataset above

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)

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



Justification: -

The simplicity and efficiency of the Naive Bayes algorithm allow it to perform better than the Linear SVM algorithm. The "glass" dataset features are actually approximately independent given the class labels.