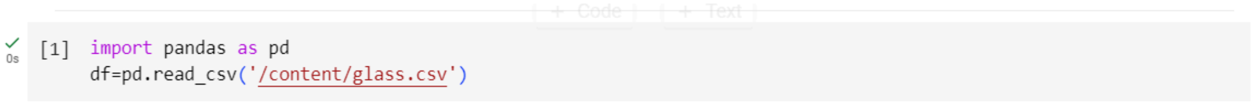
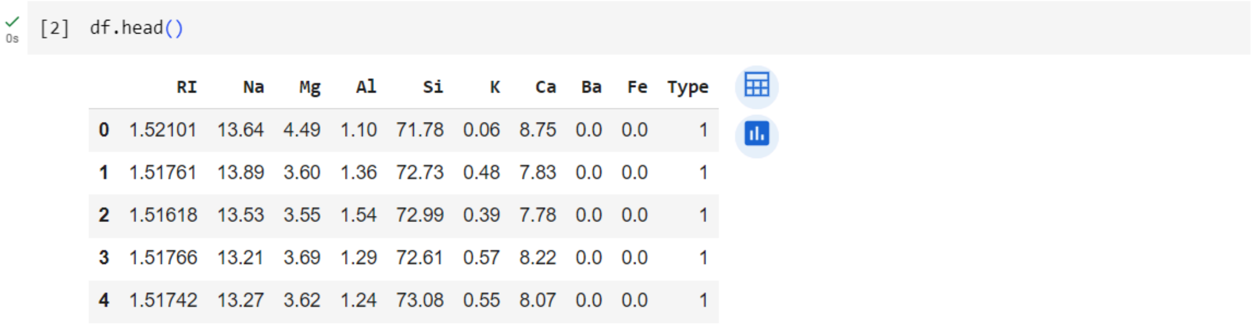
Spring 2024: CS5720 Neural Networks & Deep Learning - ICP-5
Assignment-5
NAME:Vinay Kumar Reddy Gunuguntla
STUDENT ID:700745726

Github Link:https://github.com/VinayGunuguntla/icp5.git
Video Link: NNDL_Assignment_5.mp4

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 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

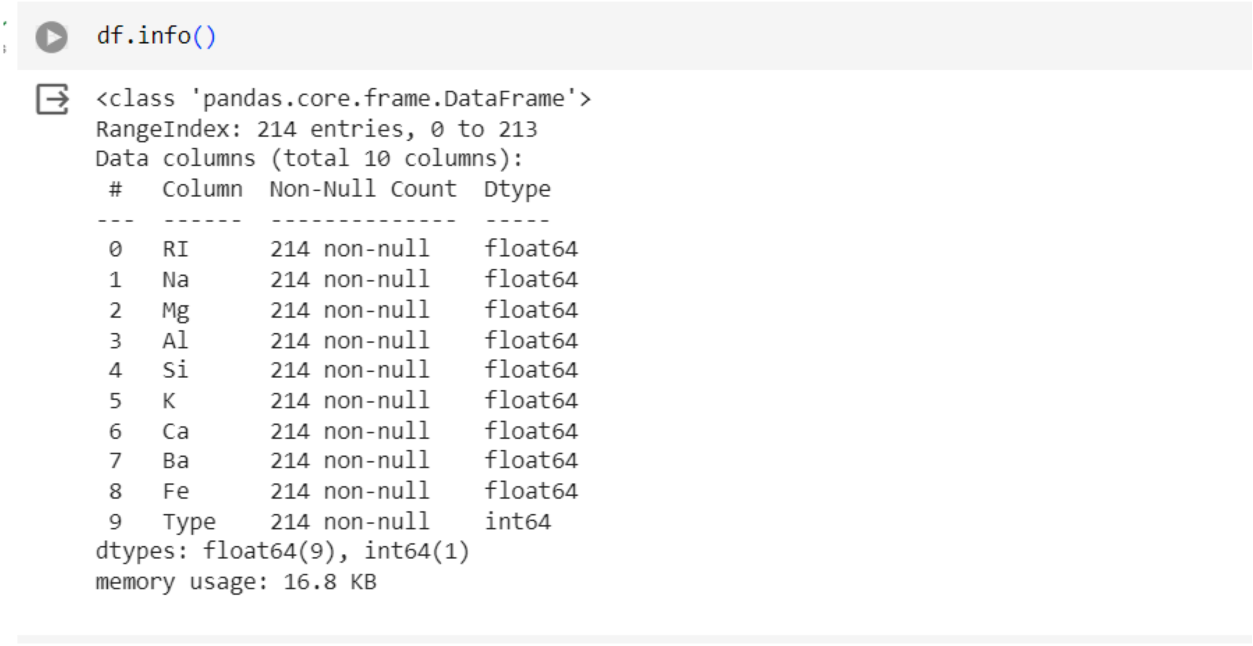


```
import pandas as pd
df=pd.read_csv('/content/glass.csv')
```



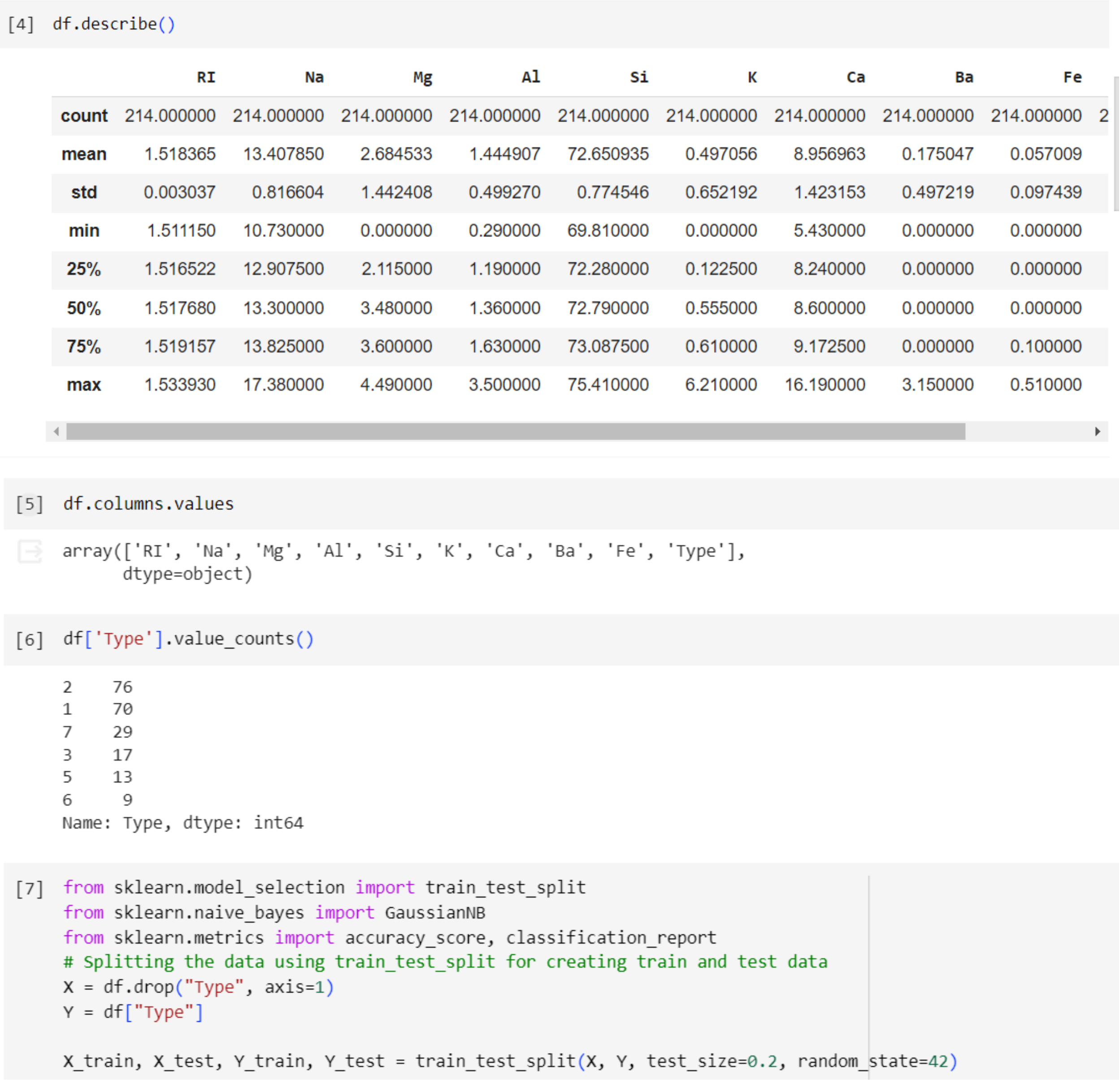
```
df.head()
```

	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()
```

```
<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
```



```
[4] df.describe()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	2
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	

```
[5] df.columns.values
```

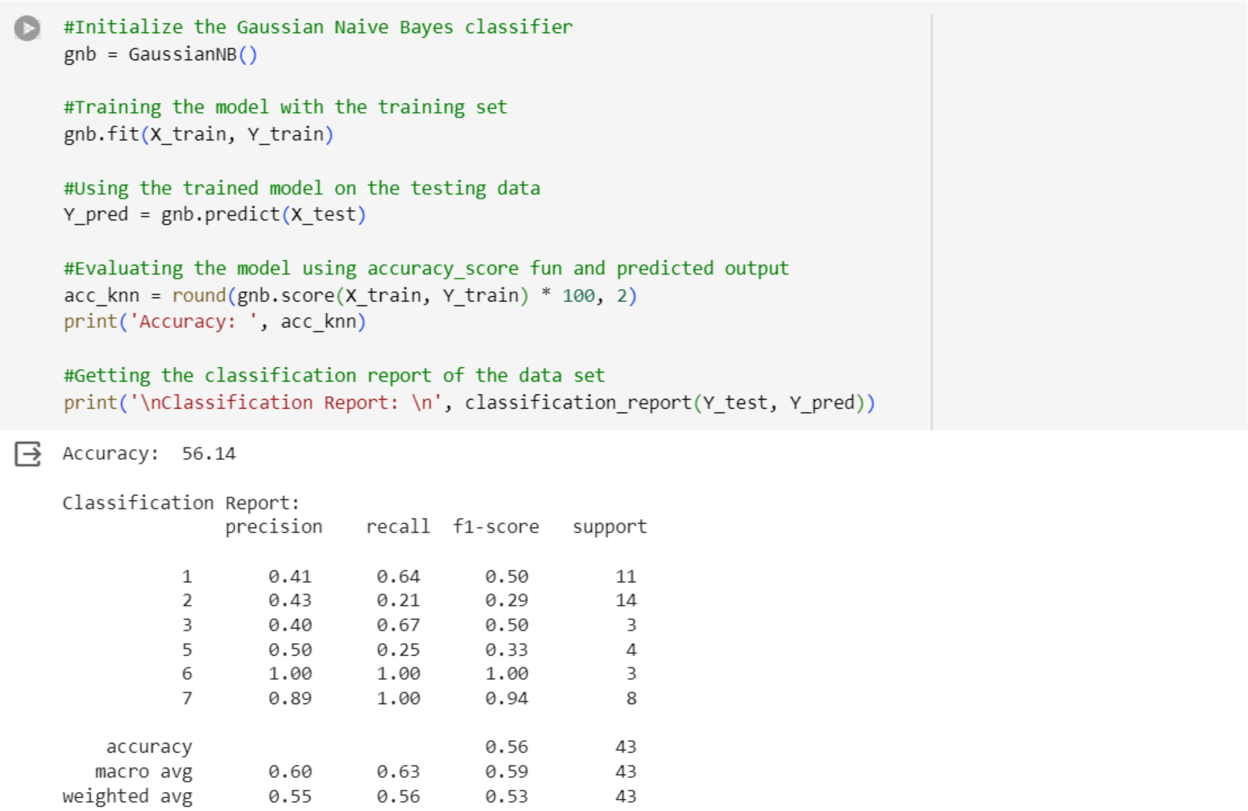
```
array(['RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe', 'Type'],
      dtype=object)
```

```
[6] df['Type'].value_counts()
```

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

```
[7] 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)
```



```
#Initialize the Gaussian Naive Bayes classifier
gnb = GaussianNB()

#Training the model with the training set
gnb.fit(X_train, Y_train)

#Using the trained model on the testing data
Y_pred = gnb.predict(X_test)

#Evaluating the model using accuracy_score fun and predicted output
acc_knn = round(gnb.score(X_train, Y_train) * 100, 2)
print('Accuracy: ', acc_knn)

#Getting the classification report of the data set
print('\nClassification Report: \n', classification_report(Y_test, Y_pred))
```

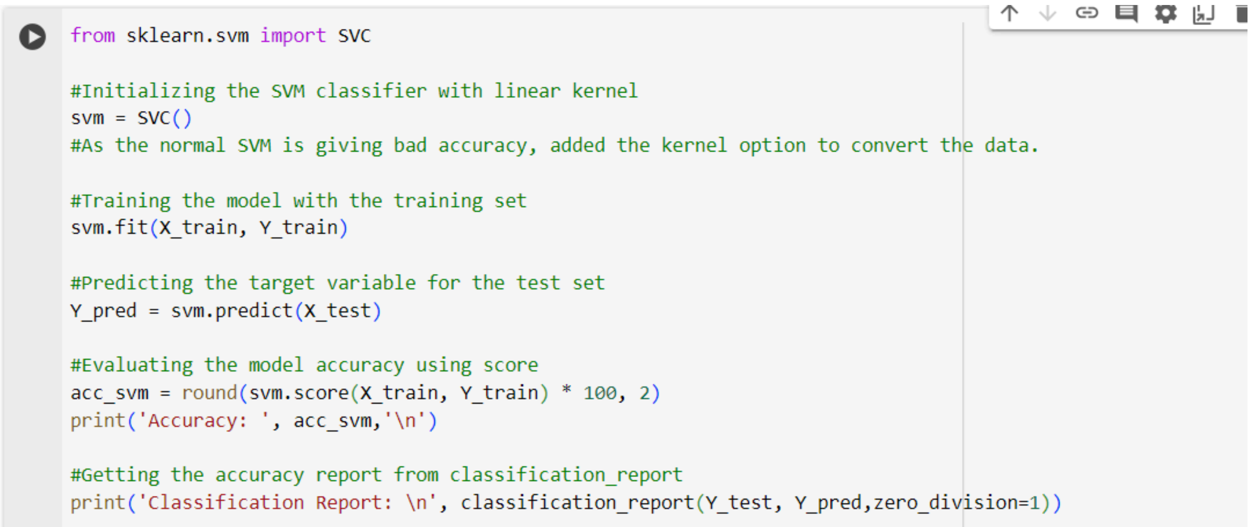
```
Accuracy:  56.14

Classification Report: 
               precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.43      0.21      0.29        14
           3       0.40      0.67      0.50         3
           5       0.50      0.25      0.33         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.56        43
   macro avg       0.60      0.63      0.59        43
weighted avg       0.55      0.56      0.53        43
```

2. Implement linear SVM method using scikit library
Use the same dataset above
Use train_test_split to create training and testing pd using scikit library Use the same dataset above Use train_test_split to create training and testing p



```
from sklearn.svm import SVC

#Initializing the SVM classifier with linear kernel
svm = SVC()
#As the normal SVM is giving bad accuracy, added the kernel option to convert the data.

#Training the model with the training set
svm.fit(X_train, Y_train)

#Predicting the target variable for the test set
Y_pred = svm.predict(X_test)

#Evaluating the model accuracy using score
acc_svm = round(svm.score(X_train, Y_train) * 100, 2)
print('Accuracy: ', acc_svm,'\n')

#Getting the accuracy report from classification_report
print('Classification Report: \n', classification_report(Y_test, Y_pred,zero_division=1))
```

```
Accuracy:  36.26

Classification Report:
              precision    recall  f1-score   support

           1       1.00      0.00      0.00        11
           2       0.33      1.00      0.49        14
           3       1.00      0.00      0.00         3
           5       1.00      0.00      0.00         4
           6       1.00      0.00      0.00         3
           7       1.00      0.00      0.00         8

    accuracy                           0.33        43
   macro avg       0.89      0.17      0.08        43
weighted avg       0.78      0.33      0.16        43
```