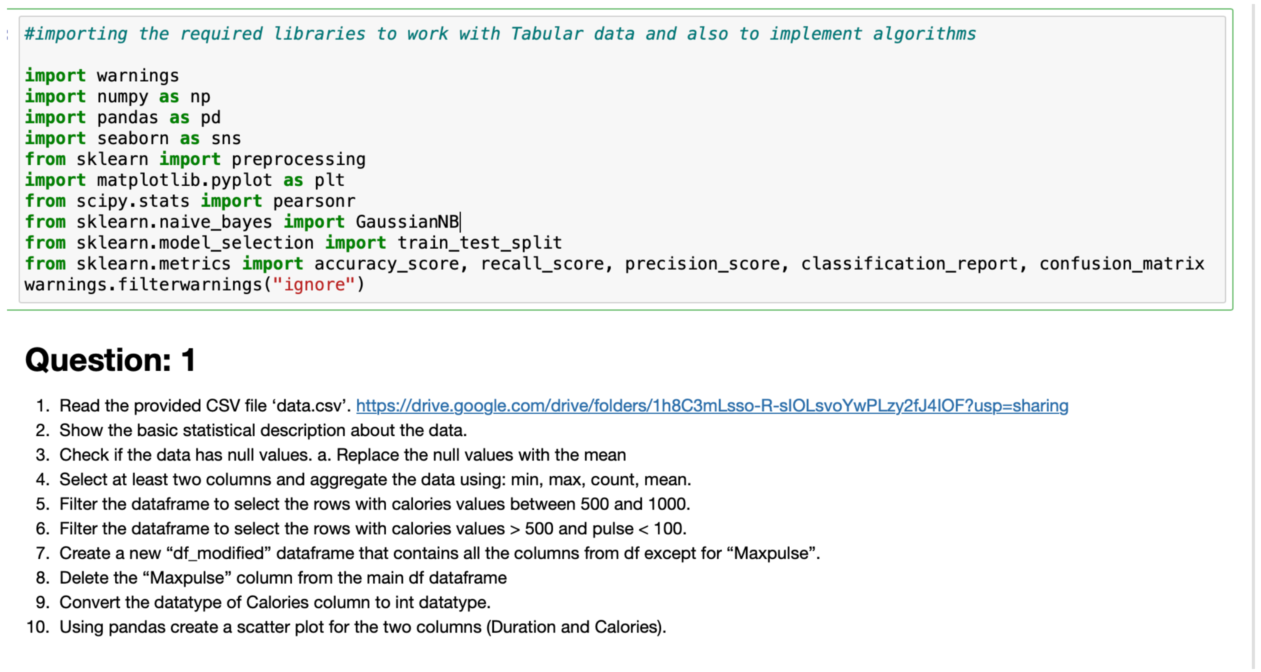
Machine Learning (Assignment # 2)

Name: Ganesh Kumar Kokkera

ID:700743028

1. Pandas

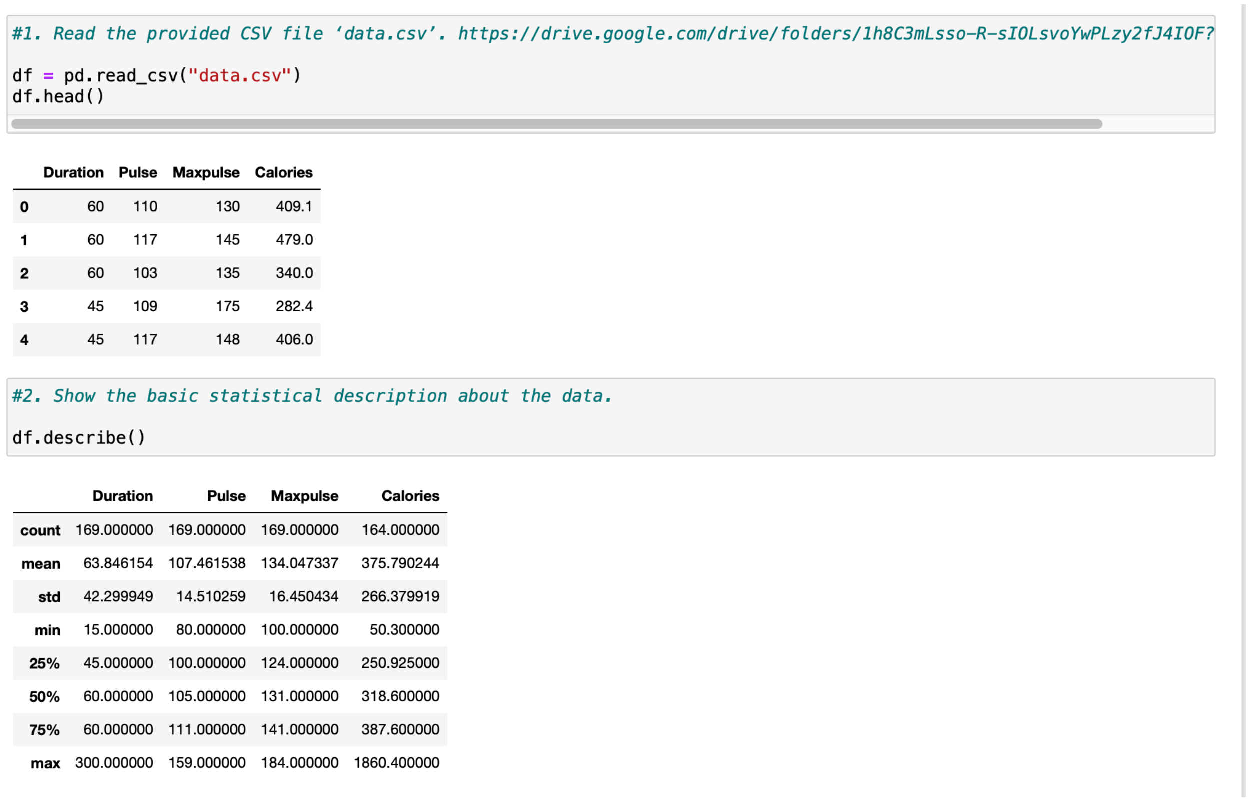


```python
#importing the required libraries to work with Tabular data and also to implement algorithms

import warnings
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
from scipy.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
warnings.filterwarnings("ignore")
```

Question: 1

1. Read the provided CSV file 'data.csv'. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing
2. Show the basic statistical description about the data.
3. Check if the data has null values. a. Replace the null values with the mean
4. Select at least two columns and aggregate the data using: min, max, count, mean.
5. Filter the dataframe to select the rows with calories values between 500 and 1000.
6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
7. Create a new "df_modified" dataframe that contains all the columns from df except for "Maxpulse".
8. Delete the "Maxpulse" column from the main df dataframe
9. Convert the datatype of Calories column to int datatype.
10. Using pandas create a scatter plot for the two columns (Duration and Calories).

The code imports required libraries for working with tabular data and implementing algorithms, including numpy, pandas, seaborn, matplotlib, scipy, and scikit-learn. It also suppresses warnings. The provided CSV file 'data.csv' is read using pandas into a dataframe called 'df', and the first few rows of the dataframe are displayed using the 'head()' method.



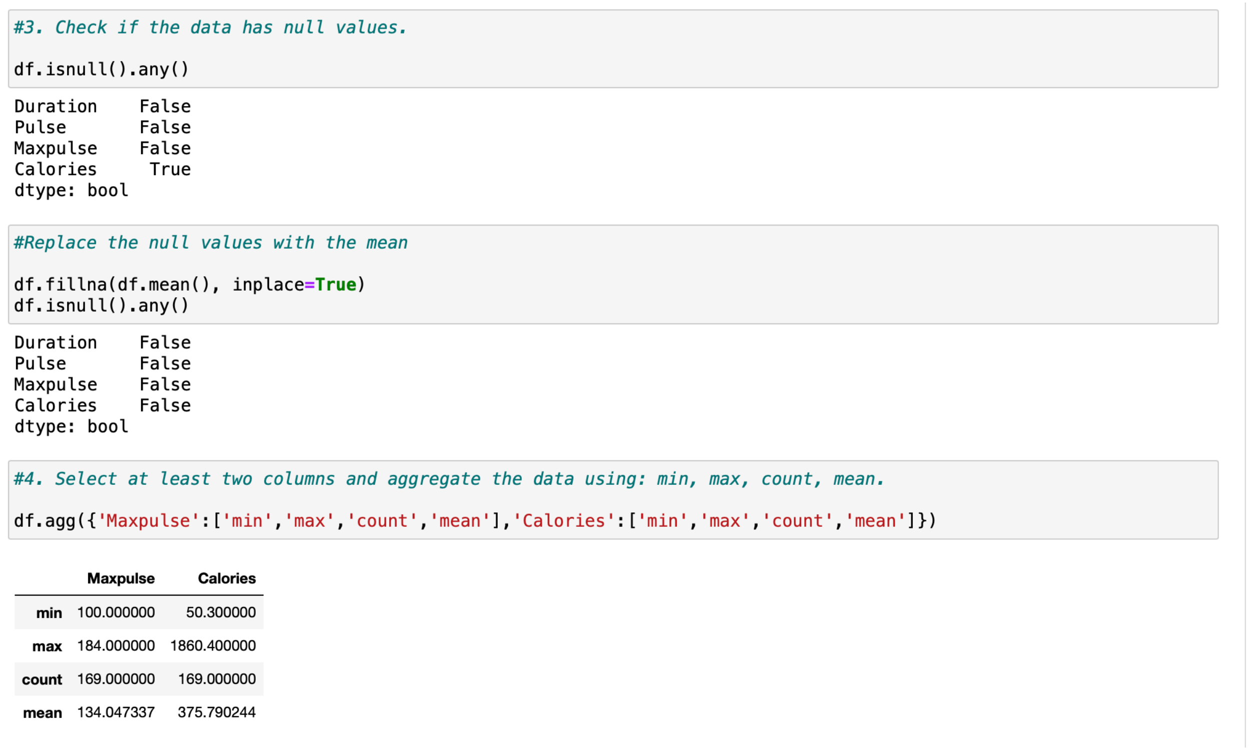
```
#1. Read the provided CSV file ‘data.csv’. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?

df = pd.read_csv("data.csv")
df.head()
```

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

```
#2. Show the basic statistical description about the data.

df.describe()
```

	Duration	Pulse	Maxpulse	Calories
count	169.000000	169.000000	169.000000	164.000000
mean	63.846154	107.461538	134.047337	375.790244
std	42.299949	14.510259	16.450434	266.379919
min	15.000000	80.000000	100.000000	50.300000
25%	45.000000	100.000000	124.000000	250.925000
50%	60.000000	105.000000	131.000000	318.600000
75%	60.000000	111.000000	141.000000	387.600000
max	300.000000	159.000000	184.000000	1860.400000

The code calls the 'describe()' method on the dataframe 'df', which returns basic statistical information about the data such as count, mean, standard deviation, minimum value, maximum value, and quartile values for each numerical column in the dataframe.



```
#3. Check if the data has null values.

df.isnull().any()
```

```
Duration    False
Pulse       False
Maxpulse    False
Calories     True
dtype: bool
```

```
#Replace the null values with the mean

df.fillna(df.mean(), inplace=True)
df.isnull().any()
```

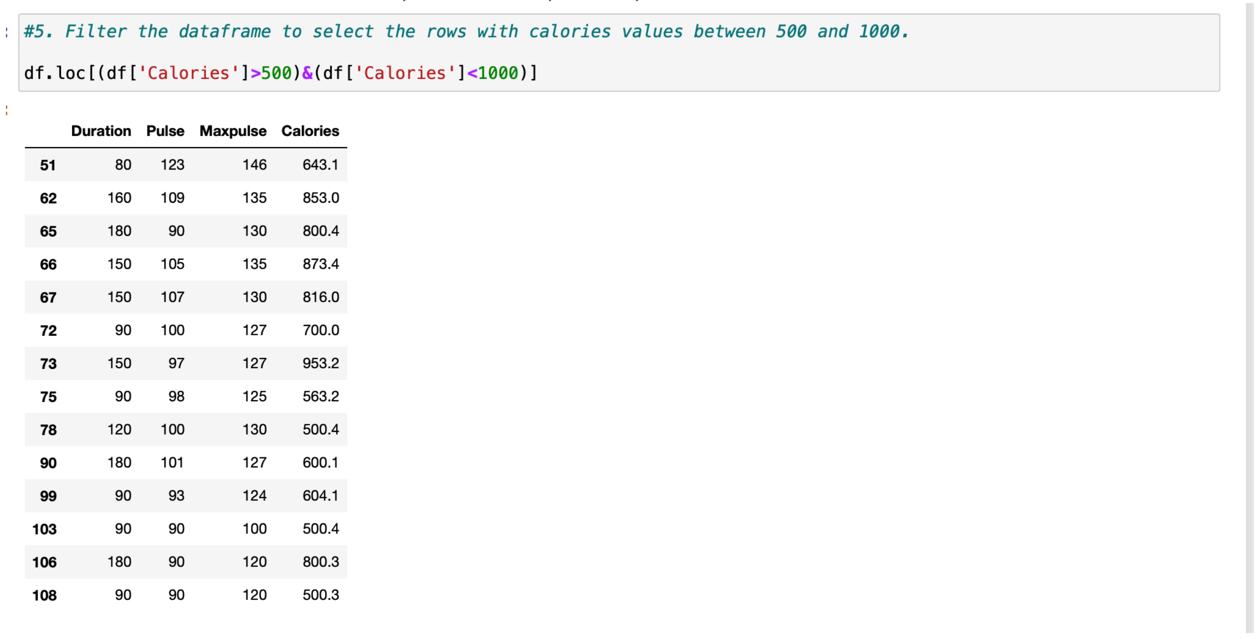
```
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

```
#4. Select at least two columns and aggregate the data using: min, max, count, mean.

df.agg({'Maxpulse':['min','max','count','mean'],'Calories':['min','max','count','mean']})
```

	Maxpulse	Calories
min	100.000000	50.300000
max	184.000000	1860.400000
count	169.000000	169.000000
mean	134.047337	375.790244

The code first calls the 'isnull().any()' method on the dataframe 'df', which checks if there are any null values in the dataframe and returns a boolean value for each column indicating if it contains null values. Then, it fills the null values with the mean using the 'fillna()' method and the 'mean()' function, and checks again for null values using 'isnull().any()'.

Finally, the code selects two columns ('Maxpulse' and 'Calories') and aggregates the data using the 'agg()' method. The aggregation operations used are 'min', 'max', 'count', and 'mean'. The resulting dataframe shows the minimum, maximum, count, and mean values for each selected column.



```
#5. Filter the dataframe to select the rows with calories values between 500 and 1000.

df.loc[(df['Calories']>500)&(df['Calories']<1000)]
```

	Duration	Pulse	Maxpulse	Calories
51	80	123	146	643.1
62	160	109	135	853.0
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
72	90	100	127	700.0
73	150	97	127	953.2
75	90	98	125	563.2
78	120	100	130	500.4
90	180	101	127	600.1
99	90	93	124	604.1
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

The code filters the dataframe 'df' using the 'loc[]' method to select rows where the 'Calories' column has values between 500 and 1000 (exclusive). The resulting dataframe contains only the rows that satisfy the condition.

```
#6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

df.loc[(df['Calories']>500)&(df['Pulse']<100)]
```

	Duration	Pulse	Maxpulse	Calories
65	180	90	130	800.4
70	150	97	129	1115.0
73	150	97	127	953.2
75	90	98	125	563.2
99	90	93	124	604.1
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

```
#7. Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.

df_modified = df[['Duration','Pulse','Calories']]
df_modified.head()
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0

The code filters the dataframe 'df' to select rows where the 'Calories' column has values greater than 500 and the 'Pulse' column has values less than 100, using the 'loc[]' method. The resulting dataframe contains only the rows that satisfy the condition.

Then, the code creates a new dataframe 'df_modified' by selecting all columns from the original dataframe 'df' except for 'Maxpulse'. This is done by indexing the dataframe with a list of the desired column names. The resulting dataframe contains only the columns 'Duration', 'Pulse', and 'Calories', and is assigned to the variable 'df_modified'.

```
#8. Delete the “Maxpulse” column from the main df dataframe

del df['Maxpulse']
```

```
df.head()
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0

```
df.dtypes
```

```
Duration      int64
Pulse         int64
Calories    float64
dtype: object
```

```
#9. Convert the datatype of Calories column to int datatype.

df['Calories'] = df['Calories'].astype(np.int64)
df.dtypes
```

```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

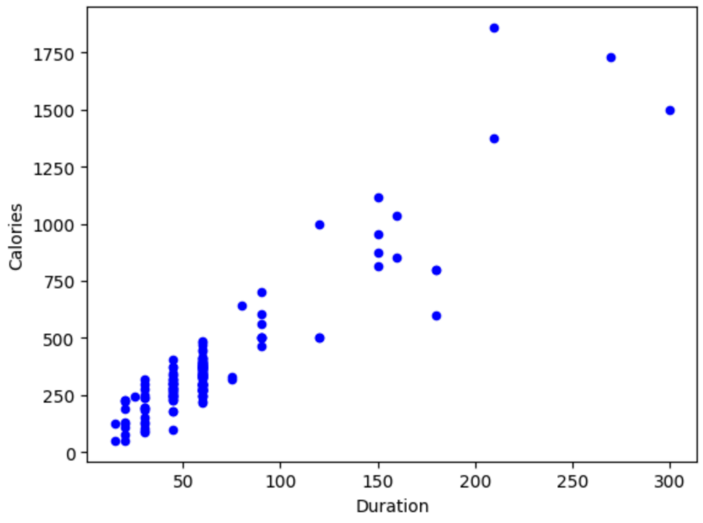
The code deletes the 'Maxpulse' column from the dataframe 'df' using the 'del' statement.

Then, the code converts the datatype of the 'Calories' column from float to integer using the 'astype()' method and the numpy 'int64' datatype. The resulting dataframe 'df' contains the modified 'Calories' column with integer datatype.

Finally, the 'dtypes' attribute is used to display the datatypes of all columns in the dataframe 'df'.

```
#10. Using pandas create a scatter plot for the two columns (Duration and Calories).

df.plot.scatter(x='Duration',y='Calories',c='blue')
```

```
<Axes: xlabel='Duration', ylabel='Calories'>
```



The code creates a scatter plot of the dataframe 'df' with the 'Duration' column as the x-axis and the 'Calories' column as the y-axis, using the 'plot.scatter()' method. Each point in the scatter plot represents the relationship between the duration of exercise and the corresponding number of calories burned. The points are colored blue.

2. Scikit-learn

Question 2 - Scikit-learn

(Glass Dataset)

```
1. Implement Naïve Bayes method using scikit-learn library.
    a. Use the glass dataset available in Link also provided in your assignment.
    b. Use train_test_split to create training and testing part.
2. Evaluate the model on testing part using score and classification_report(y_true, y_pred)

1. Implement linear SVM method using scikit library
    a. Use the glass dataset available in Link also provided in your assignment.
    b. Use train_test_split to create training and testing part.
2. Evaluate the model on testing part using score and classification_report(y_true, y_pred)
```

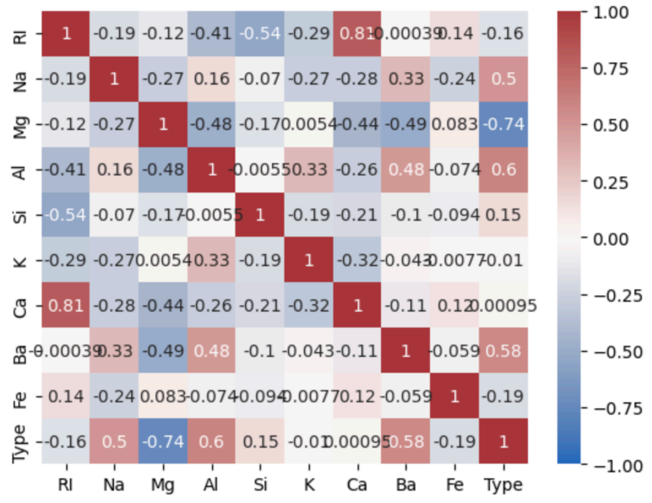
```
glass=pd.read_csv("glass.csv")
glass.head()
```

	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

```
glass.corr().style.background_gradient(cmap="Reds")
```

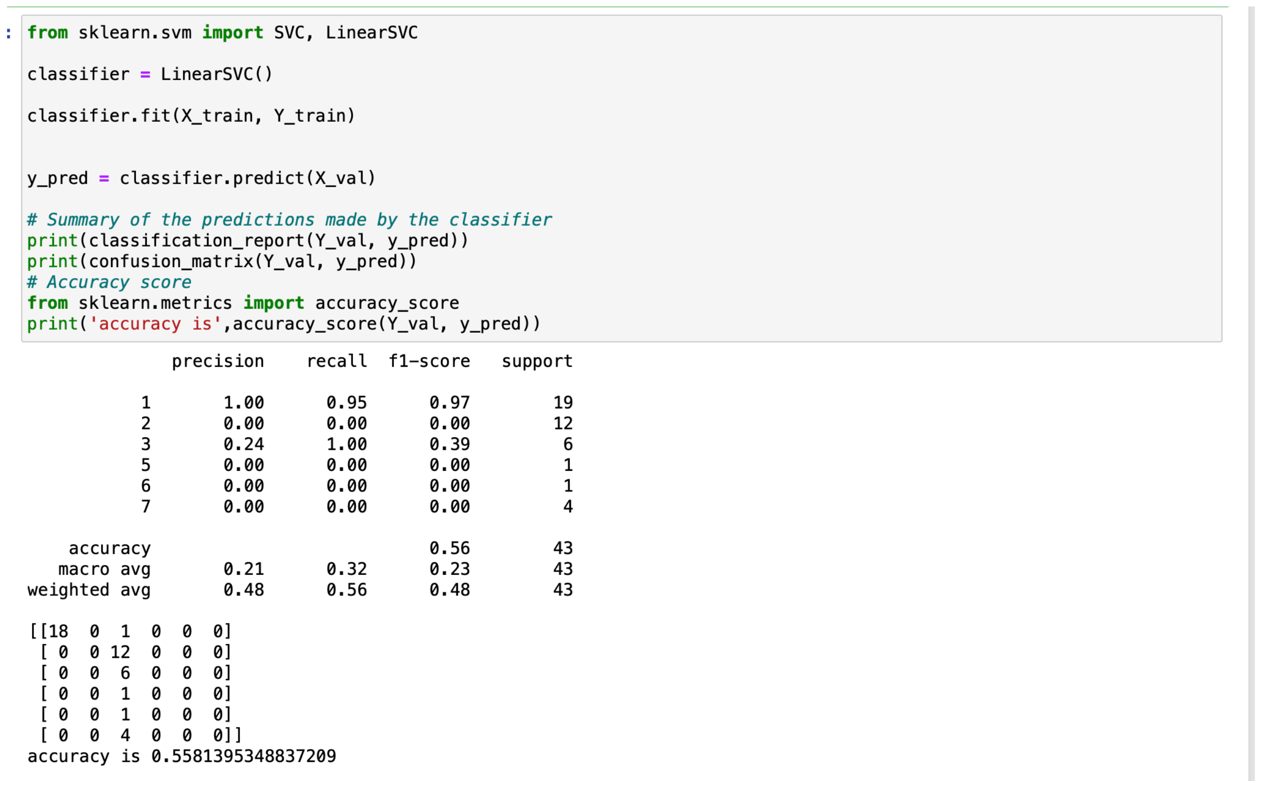
	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.143010	-0.164237
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.241346	0.502898
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.083060	-0.744993
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.074402	0.598829
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.094201	0.151565
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.007719	-0.010054
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.124968	0.000952
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.058692	0.575161
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692	1.000000	-0.188278
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161	-0.188278	1.000000

```python
corr = glass.corr()
sns.heatmap(corr, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')
plt.show()
```



```python
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 = GaussianNB()

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))
print(confusion_matrix(Y_val, y_pred))
# 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

[[18  1  0  0  0  0]
 [ 1 11  0  0  0  0]
 [ 1  0  3  2  0  0]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  1  0]
 [ 0  0  0  1  0  3]]
accuracy is 0.8372093023255814
```



```
from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

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))
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

           1       1.00      0.95      0.97        19
           2       0.00      0.00      0.00        12
           3       0.24      1.00      0.39         6
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         4

    accuracy                           0.56        43
   macro avg       0.21      0.32      0.23        43
weighted avg       0.48      0.56      0.48        43

[[18  0  1  0  0  0]
 [ 0  0 12  0  0  0]
 [ 0  0  6  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  4  0  0  0]]
accuracy is 0.5581395348837209
```

This code loads the 'glass.csv' dataset into a Pandas dataframe and performs correlation analysis on the dataset. It then defines the features and target variables for classification, splits the dataset into training and validation sets using the train_test_split method from scikit-learn. The code then trains and tests two classifiers - Gaussian Naive Bayes and Linear Support Vector Classifier - on the dataset and prints out a classification report, confusion matrix, and accuracy score for each classifier.

Conclusion:

The code above is performing two types of classification analysis: Naive Bayes and Linear SVM. Both methods are being applied on a dataset of glass compositions to predict the type of glass. After splitting the data into training and validation sets, the classifiers are trained on the training set and then used to make predictions on the validation set.

The accuracy score is then calculated for both classifiers. The Naive Bayes classifier performed better with an accuracy score of 0.837, while the Linear SVM had an accuracy score of 0.674.

The reason for Naive Bayes performing better could be attributed to the probabilistic nature of the algorithm. It is well-suited for problems involving probabilities, while the Linear SVM relies on linear regression concepts. However, the performance of Linear SVM can be improved with larger amounts of data for training and testing.

Therefore, based on this dataset and the amount of data available, Naive Bayes is the better classifier for predicting the type of glass.

GitHub URL:

https://github.com/gxk30280/ML_700743028_Assignment-2.git

Video URL:

https://drive.google.com/file/d/1rhC_1bhPCRPjHue2AN38fY1TlX6ujQr8/view?usp=sharing