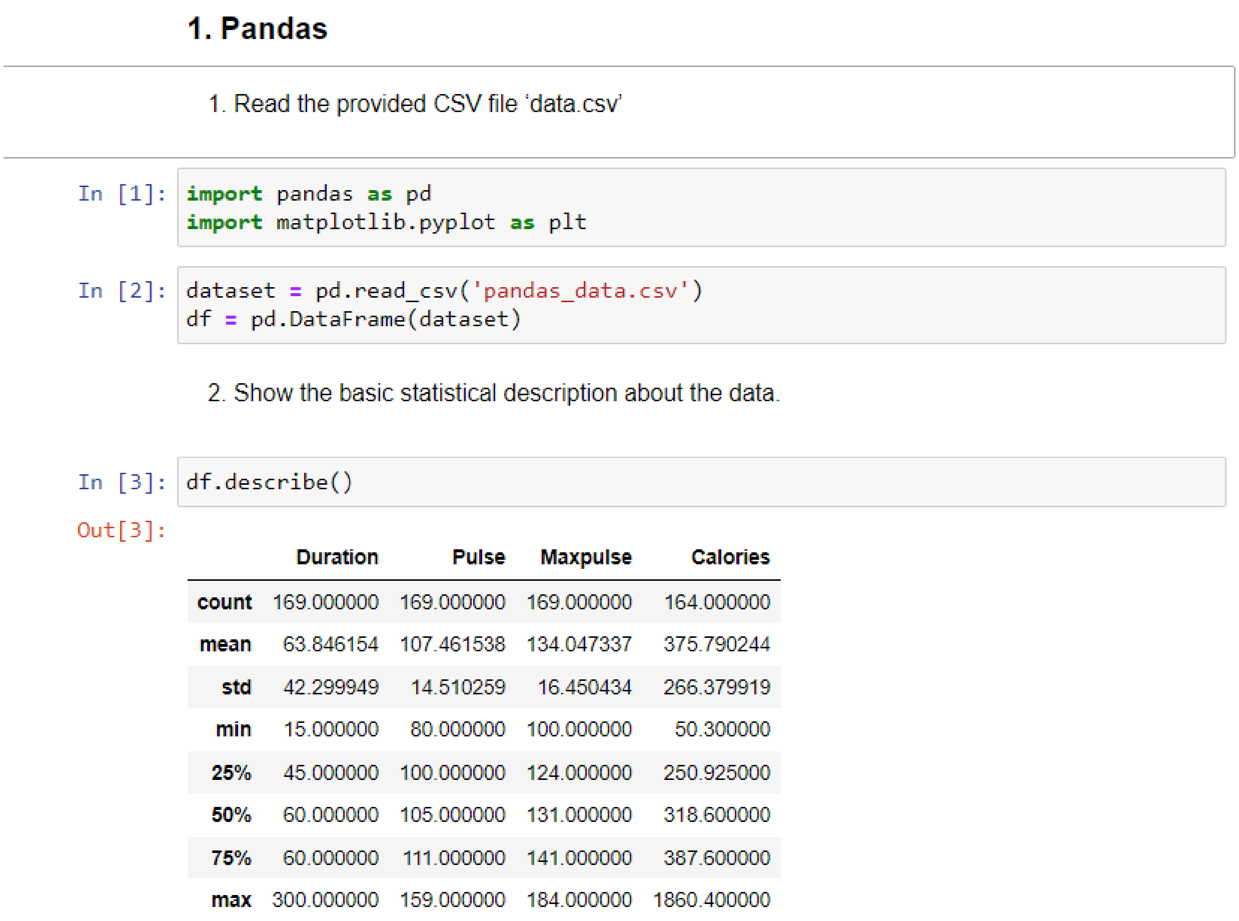
Name: Deepika Rani Gokavarapu
700#: 700740599
Programming Assignment 2

GitHub Link: https://github.com/GDeepikarani59/MachineLearning_Assignment2
Video Link:
https://drive.google.com/file/d/1pwfFRTxinMaEkBu8gJQnYa2Vv8ENXhOD/view?usp=drive_link

Question 1:



1. Pandas

1. Read the provided CSV file ‘data.csv’

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
```

```
In [2]: dataset = pd.read_csv('pandas_data.csv')
        df = pd.DataFrame(dataset)
```

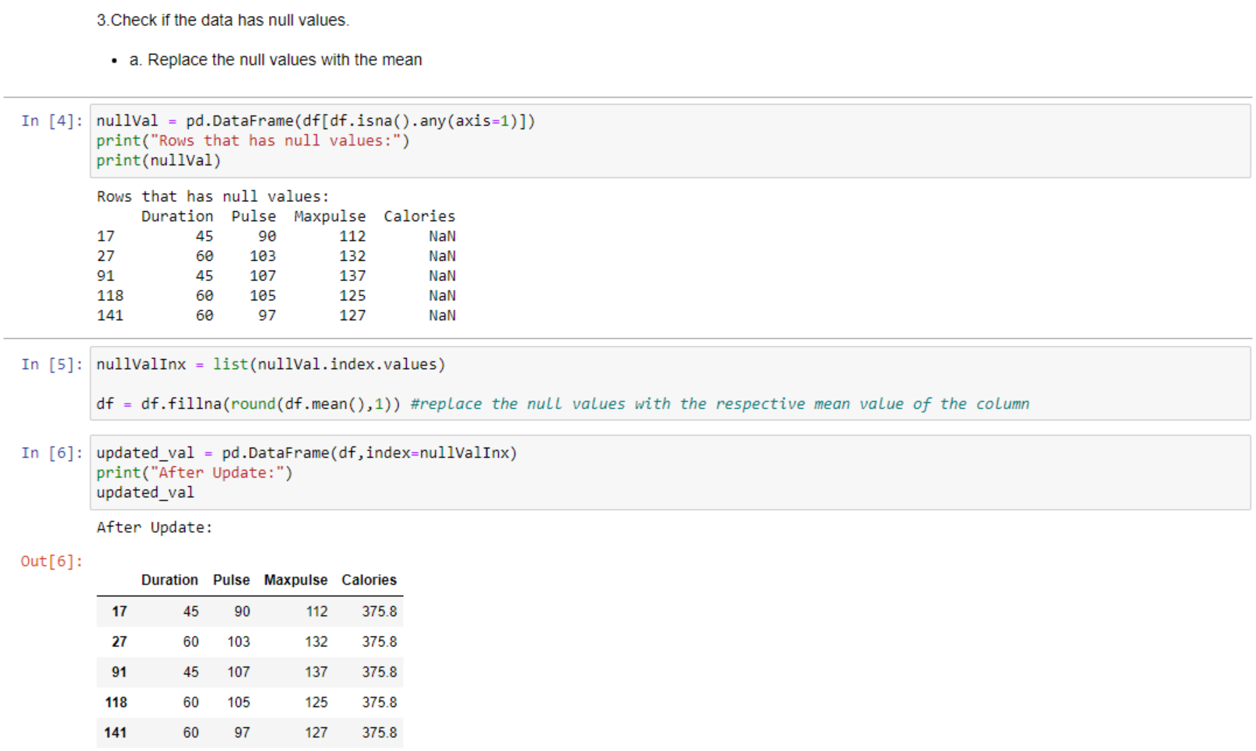
2. Show the basic statistical description about the data.

```
In [3]: df.describe()
```

Out[3]:

	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

Import the pandas libraries. Read the csv file containing the data sets and display the basic statistical descriptions of the dataset.



3.Check if the data has null values.

- a. Replace the null values with the mean

In [4]:
```
nullVal = pd.DataFrame(df[df.isna().any(axis=1)])
print("Rows that has null values:")
print(nullVal)
```

```
Rows that has null values:
     Duration  Pulse  Maxpulse  Calories
17         45     90       112       NaN
27         60    103       132       NaN
91         45    107       137       NaN
118        60    105       125       NaN
141        60     97       127       NaN
```

In [5]:
```
nullValInx = list(nullVal.index.values)

df = df.fillna(round(df.mean(),1)) #replace the null values with the respective mean value of the column
```

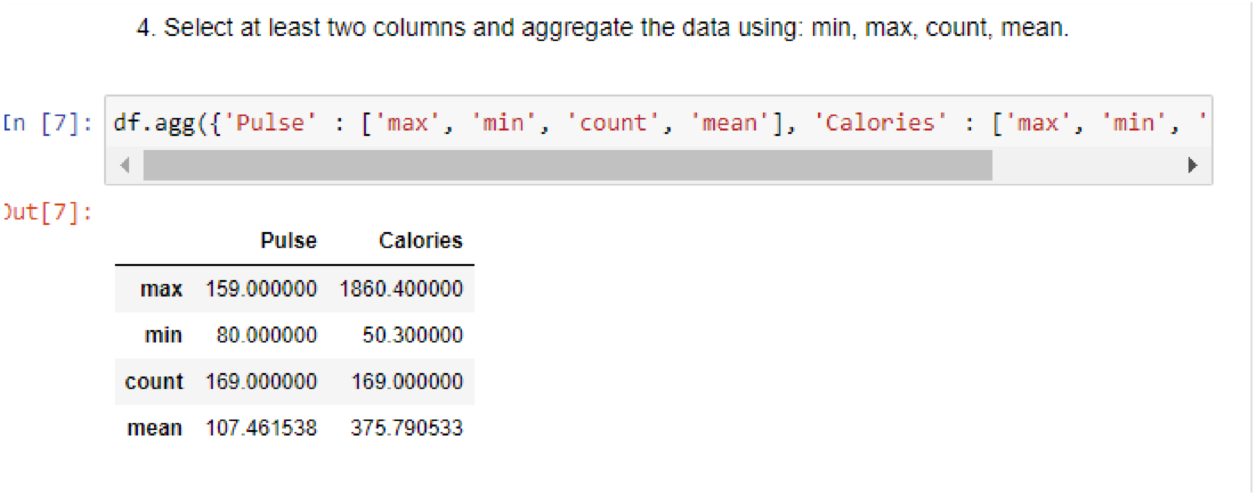
In [6]:
```
updated_val = pd.DataFrame(df,index=nullValInx)
print("After Update:")
updated_val
```

```
After Update:
```

Out[6]:

	Duration	Pulse	Maxpulse	Calories
17	45	90	112	375.8
27	60	103	132	375.8
91	45	107	137	375.8
118	60	105	125	375.8
141	60	97	127	375.8

Collect the rows that have any of their column values set to null, then copy the indexes of those rows into a list. To observe how our data will seem after updating the null values, we are saving the indexes here. Replace any null values with their corresponding means. We can now view the rows' updated appearances using the row indexes.



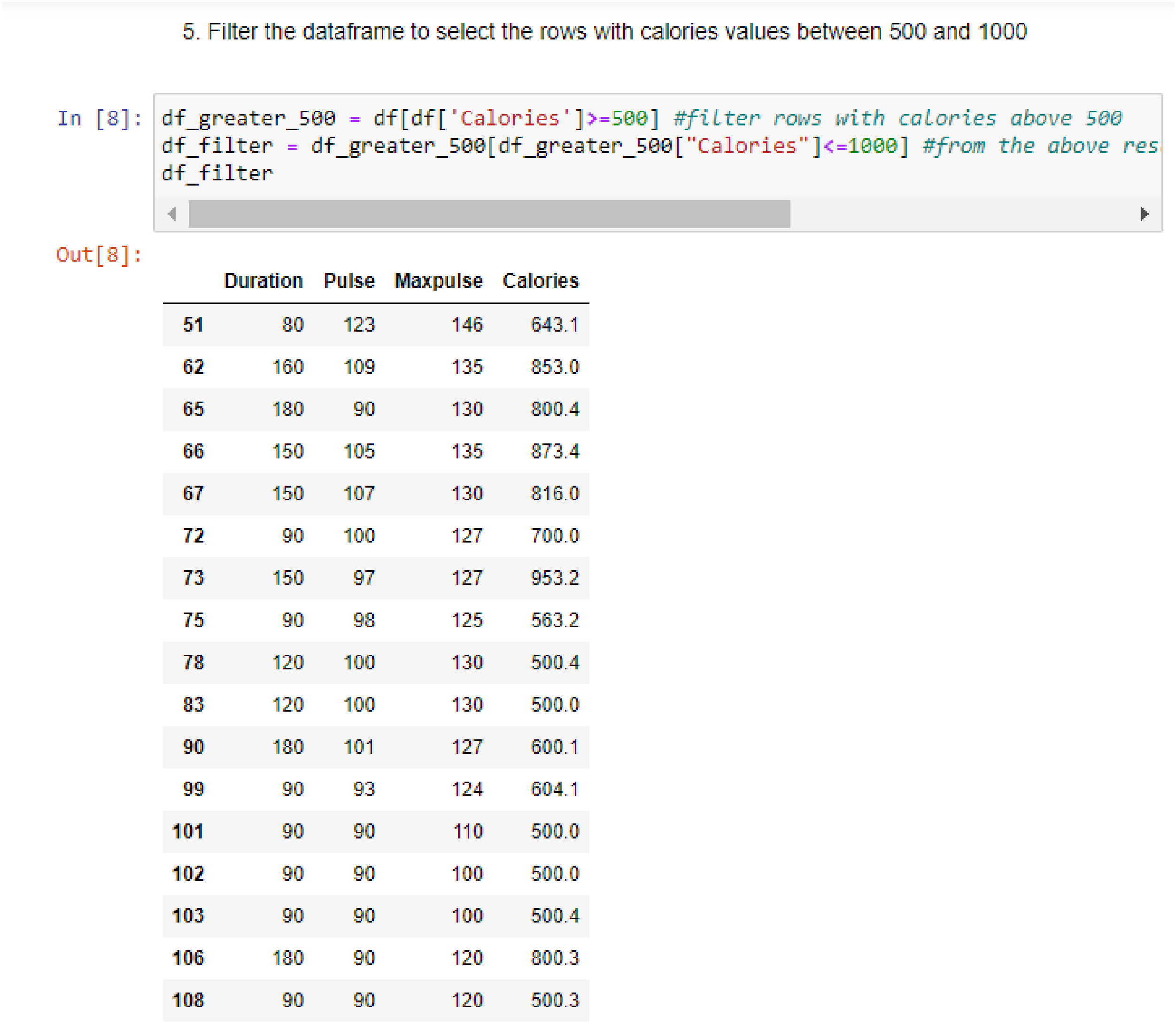
4. Select at least two columns and aggregate the data using: min, max, count, mean.

[n [7]:
```
df.agg({'Pulse' : ['max', 'min', 'count', 'mean'], 'Calories' : ['max', 'min', '
```

)ut[7]:

	Pulse	Calories
max	159.000000	1860.400000
min	80.000000	50.300000
count	169.000000	169.000000
mean	107.461538	375.790533

Here, the maximum value, minimum value, count, and mean of the two columns, pulse and calories, are combined and displayed.



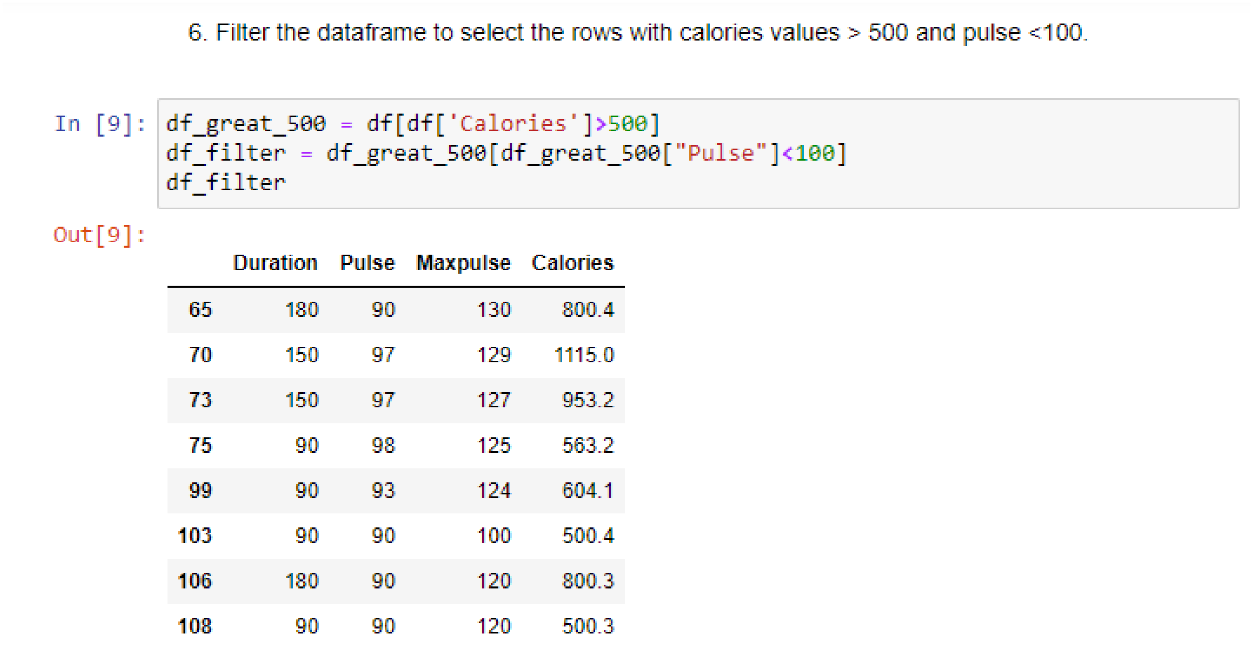
5. Filter the dataframe to select the rows with calories values between 500 and 1000

```
In [8]: df_greater_500 = df[df['Calories']>=500] #filter rows with calories above 500
        df_filter = df_greater_500[df_greater_500["Calories"]<=1000] #from the above res
        df_filter
```

Out[8]:

	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
83	120	100	130	500.0
90	180	101	127	600.1
99	90	93	124	604.1
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

The rows with Calories column values between 500 and 1000 are shown below. There are two steps to this. We first filter out numbers greater than 500 and record the results. After that, only the data with values less than 1000 will be kept.



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

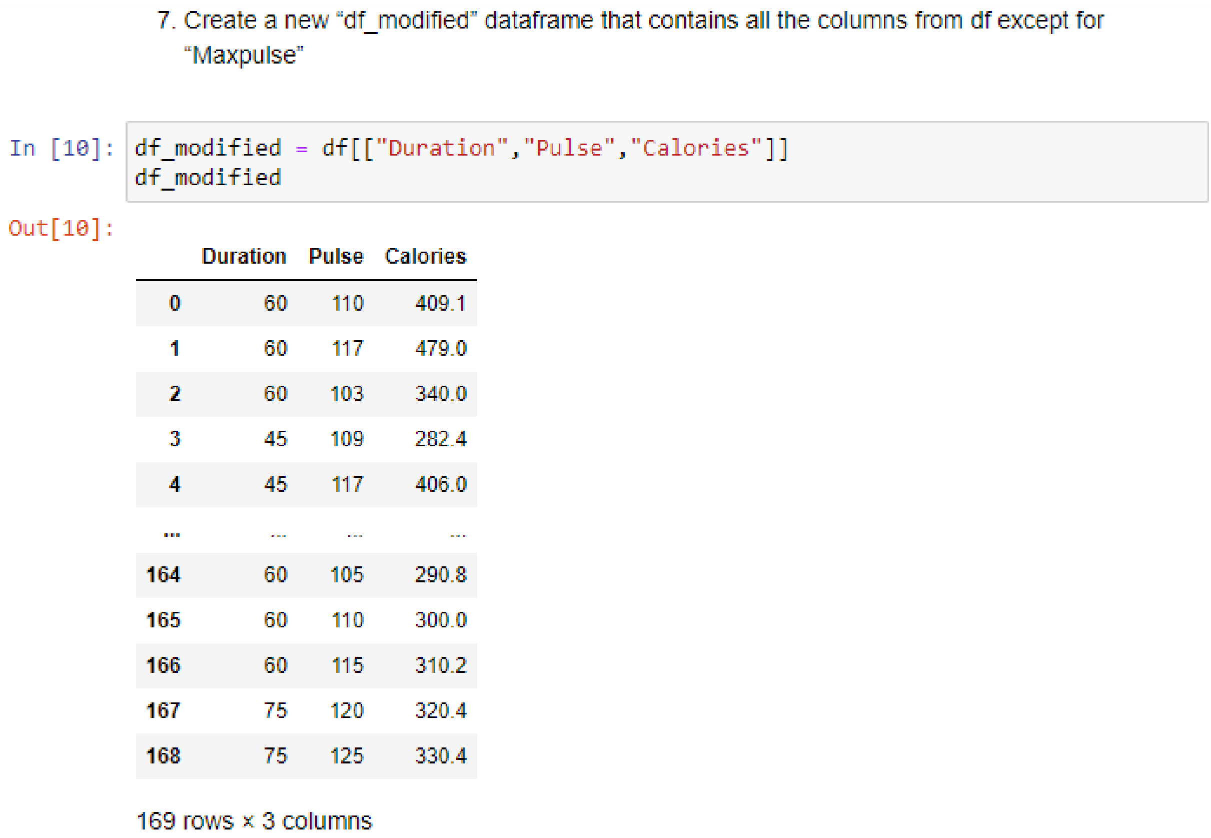
In [9]:

```
df_great_500 = df[df['Calories']>500]
df_filter = df_great_500[df_great_500["Pulse"]<100]
df_filter
```

Out[9]:

	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

The rows with Calories column values greater than 500 and Pulse column values less than 100 are shown here. There are two steps to this. In the beginning, we filter out calories with values more than 500 and store them. Then, filter the output data to exclude Pulse values below 1000.



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

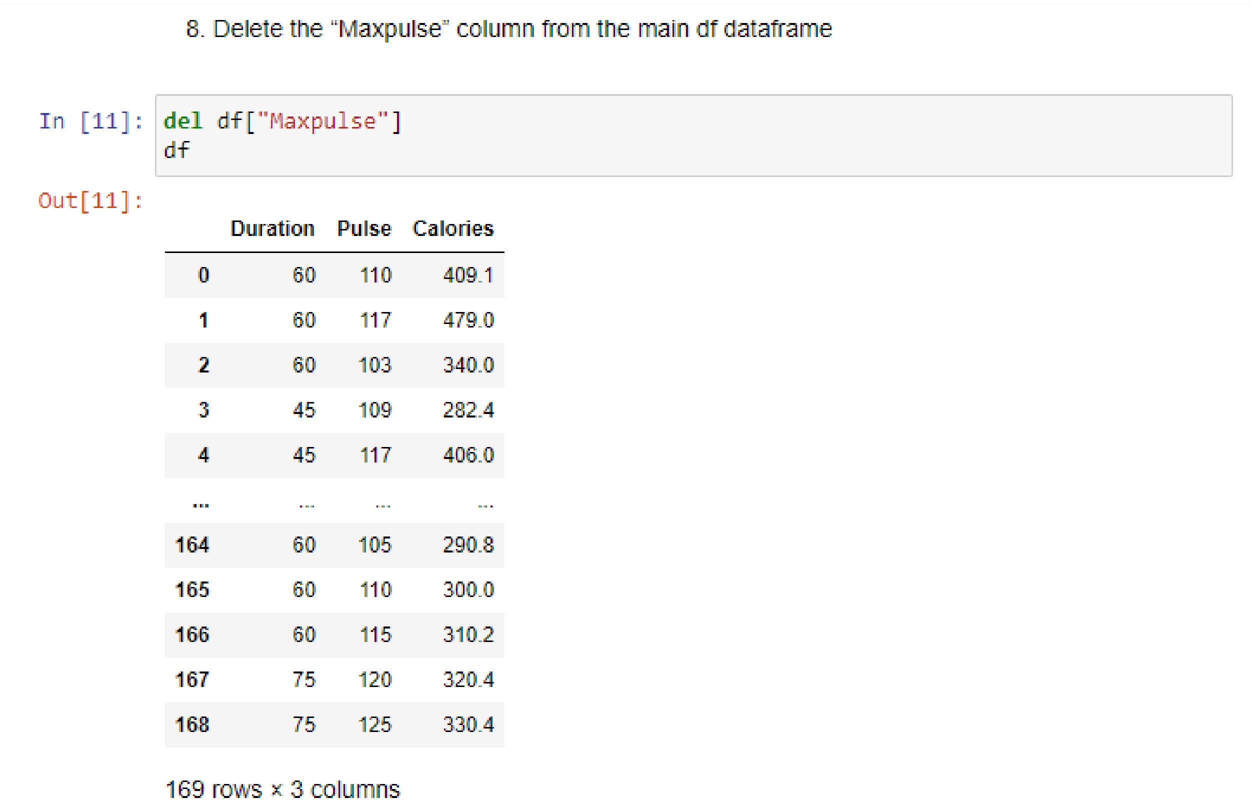
```
In [10]: df_modified = df[["Duration","Pulse","Calories"]]
         df_modified
```

Out[10]:

	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
...	...	...	...
164	60	105	290.8
165	60	110	300.0
166	60	115	310.2
167	75	120	320.4
168	75	125	330.4

169 rows × 3 columns

Creating a new data frame containing the all the columns except Maxpulse



8. Delete the "Maxpulse" column from the main df dataframe

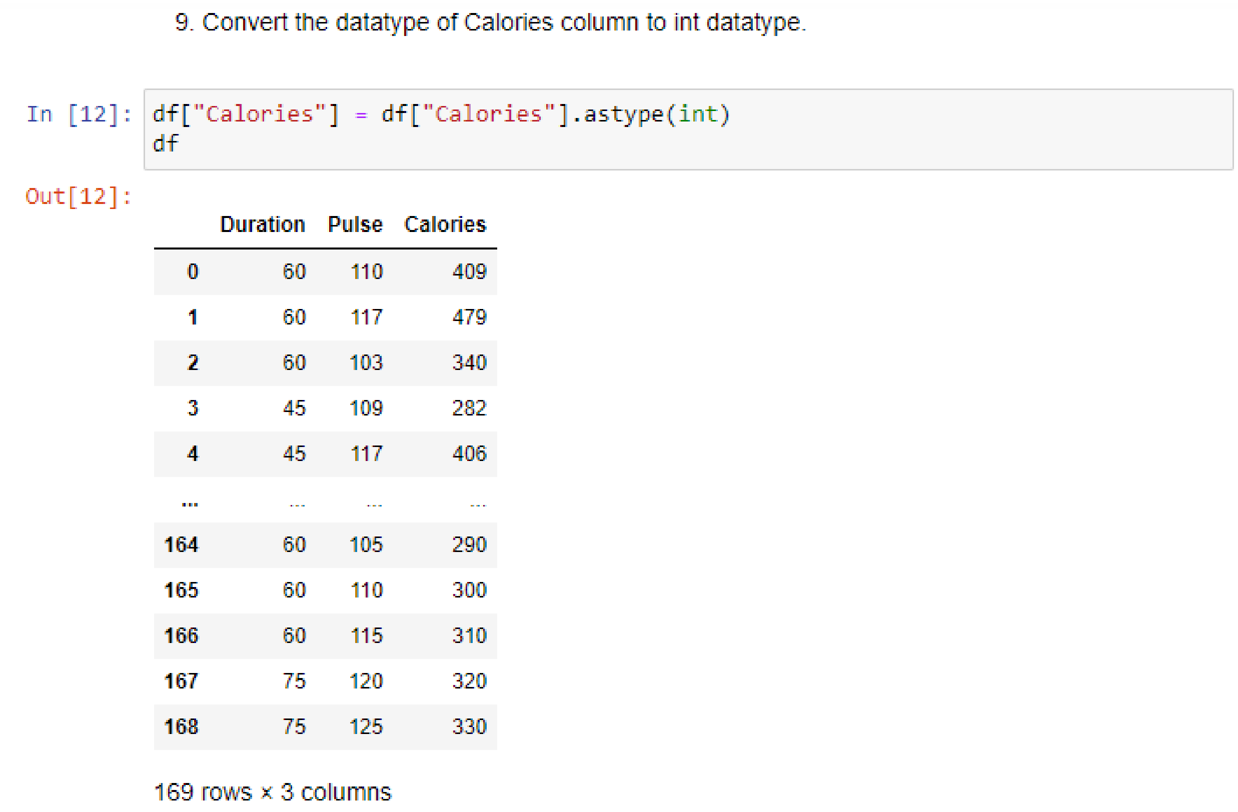
```
In [11]: del df["Maxpulse"]
         df
```

Out[11]:

	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
...	...	...	...
164	60	105	290.8
165	60	110	300.0
166	60	115	310.2
167	75	120	320.4
168	75	125	330.4

169 rows × 3 columns

From the main data frame, removed the Maxpulse column.



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

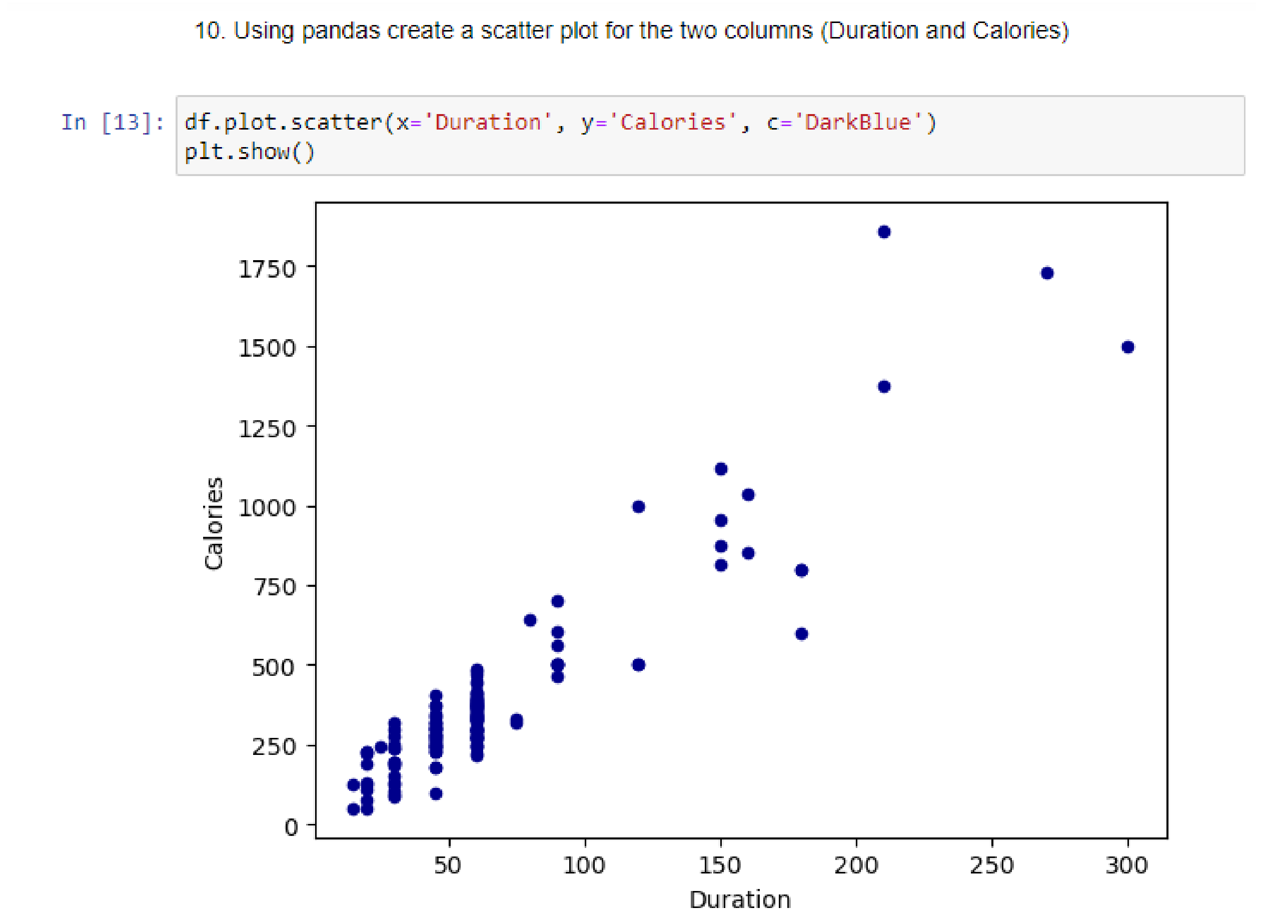
In [12]:
```
df["Calories"] = df["Calories"].astype(int)
df
```

Out[12]:

	Duration	Pulse	Calories
0	60	110	409
1	60	117	479
2	60	103	340
3	45	109	282
4	45	117	406
...	...	...	...
164	60	105	290
165	60	110	300
166	60	115	310
167	75	120	320
168	75	125	330

169 rows × 3 columns

changing the calorie column's float datatype to an int datatype.

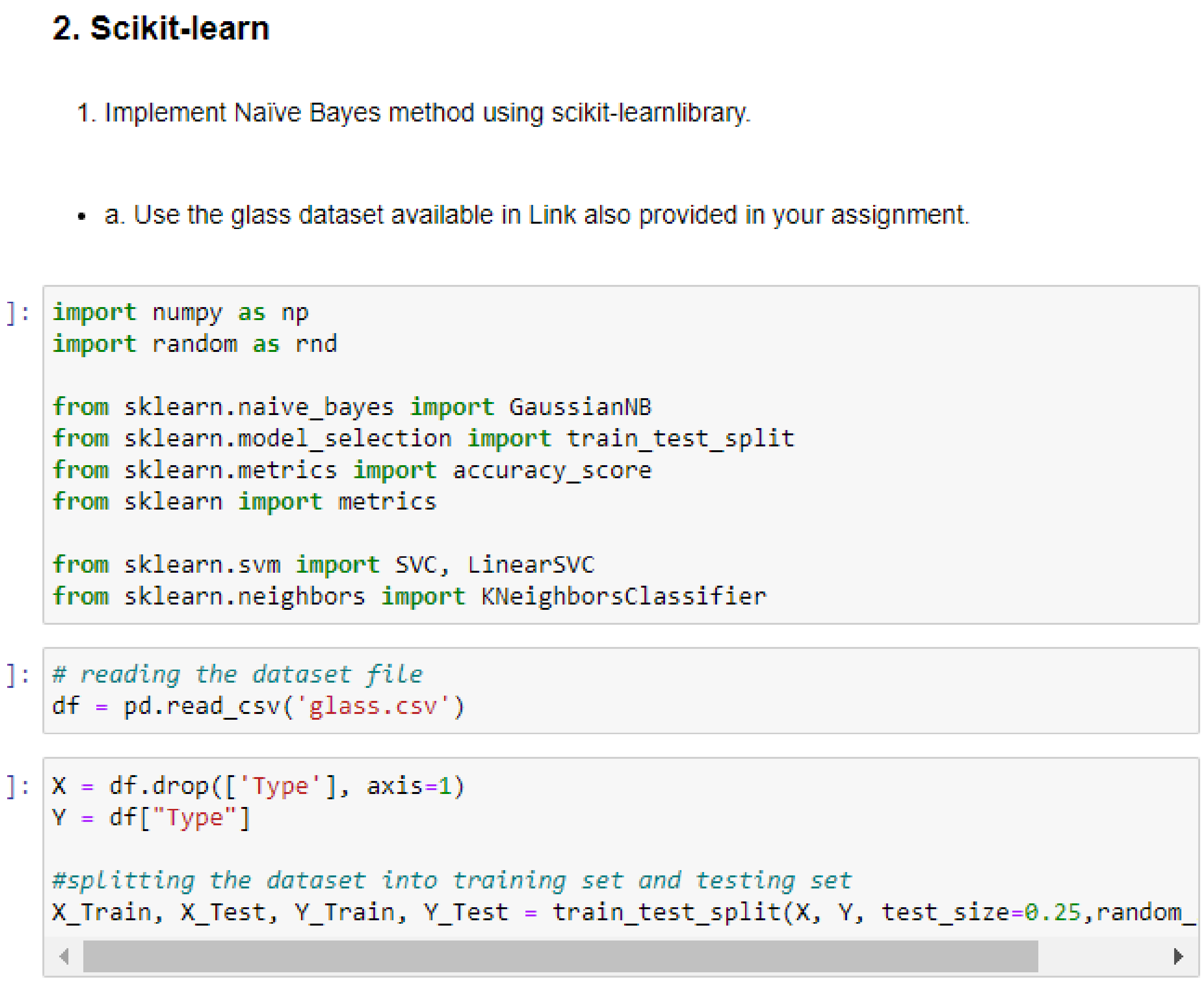


10. Using pandas create a scatter plot for the two columns (Duration and Calories)

```
In [13]: df.plot.scatter(x='Duration', y='Calories', c='DarkBlue')
         plt.show()
```

Created a scatter plot for Duration and Calories.

Question 2:



2. Scikit-learn

1. Implement Naïve Bayes method using scikit-learnlibrary.

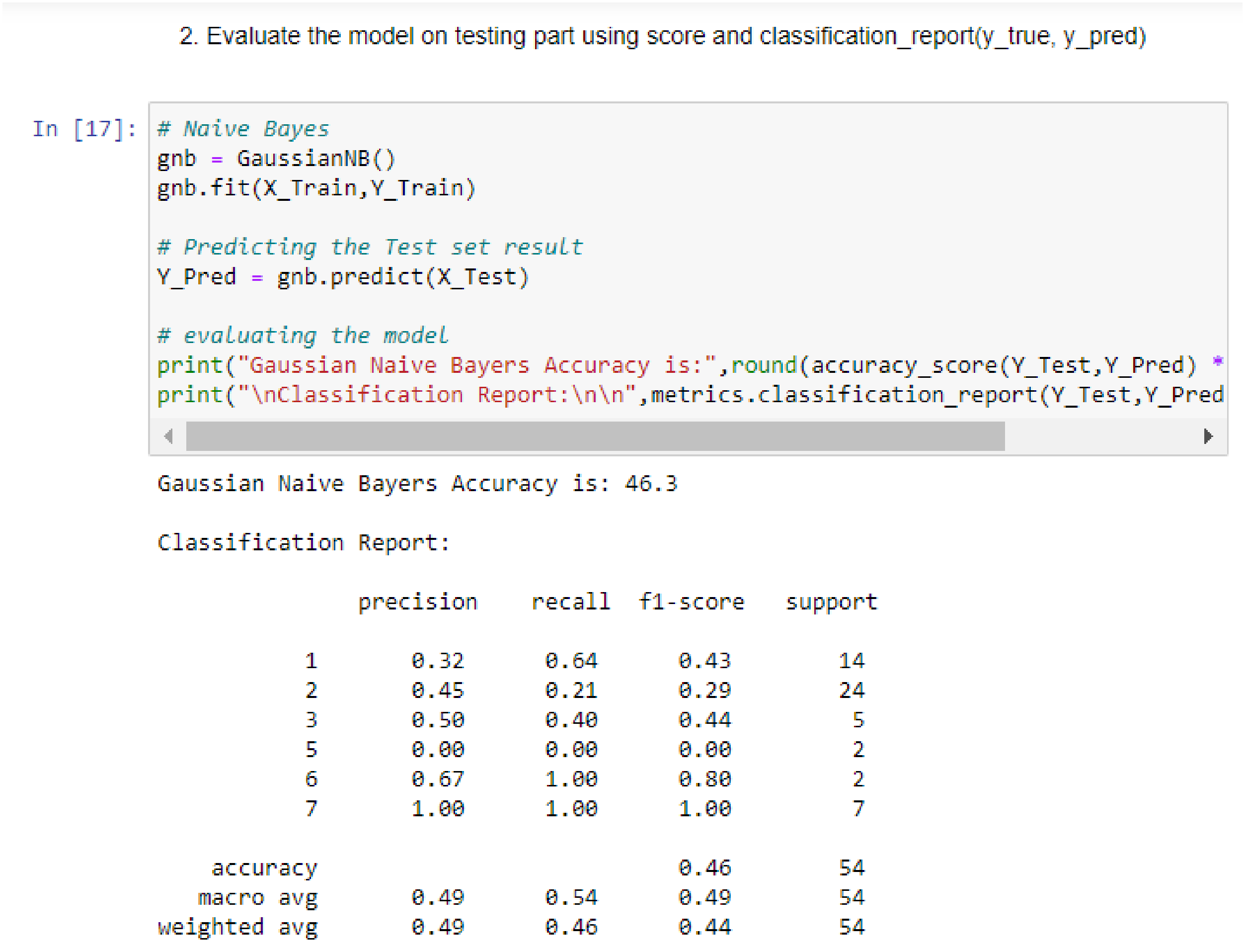
- a. Use the glass dataset available in Link also provided in your assignment.

```
]: import numpy as np
   import random as rnd

   from sklearn.naive_bayes import GaussianNB
   from sklearn.model_selection import train_test_split
   from sklearn.metrics import accuracy_score
   from sklearn import metrics

   from sklearn.svm import SVC, LinearSVC
   from sklearn.neighbors import KNeighborsClassifier
```

```
]: # reading the dataset file
   df = pd.read_csv('glass.csv')
```

```
]: X = df.drop(['Type'], axis=1)
   Y = df["Type"]

   #splitting the dataset into training set and testing set
   X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size=0.25,random_
```

Imported necessary libraries. Downloaded glass.csv file from the provided link and converted into a data frame using pandas and removed the column ‘type’. Splitted the dataset into training and testing dataset in 75 25 ratio respectively using train_test_split method.

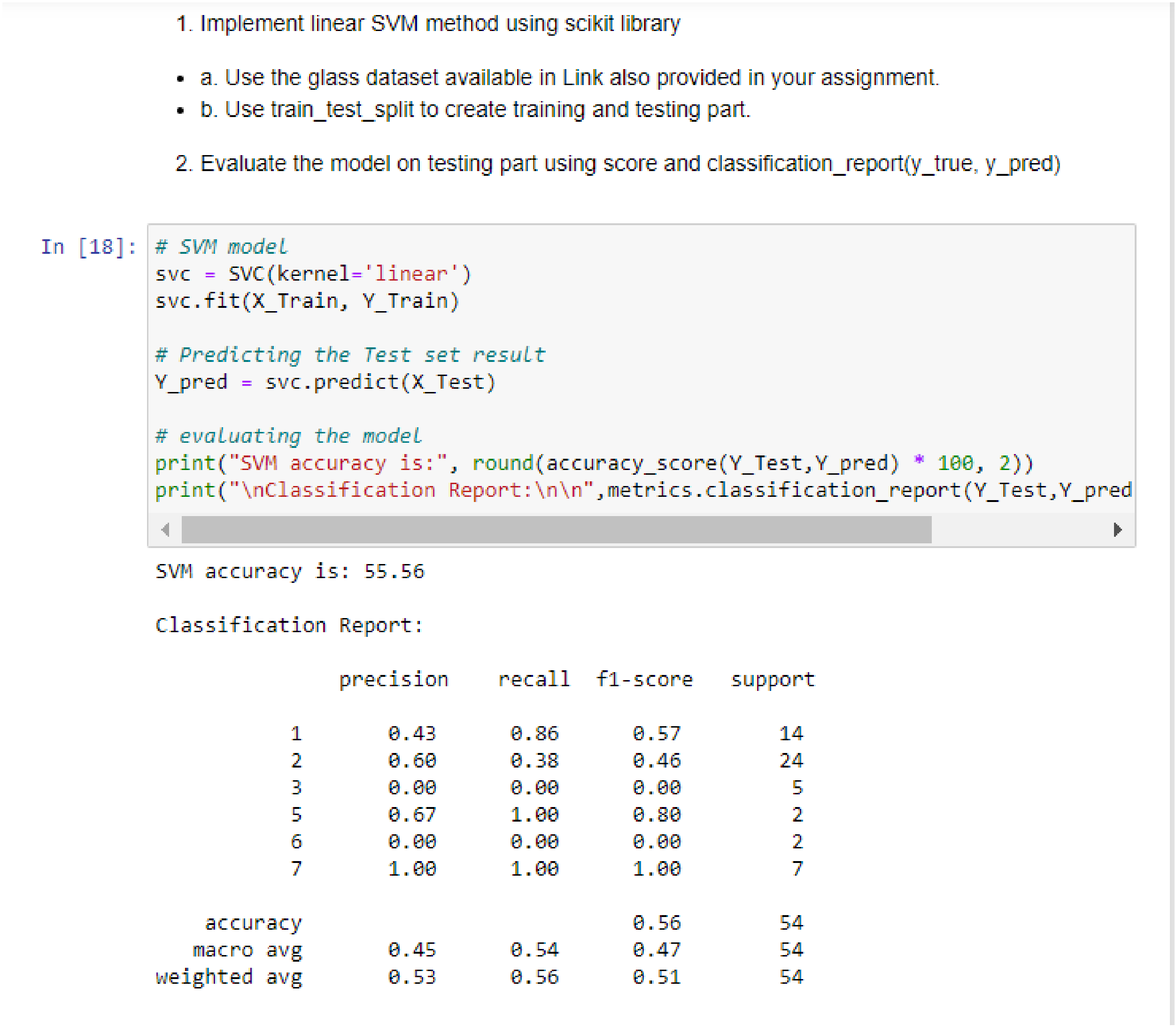


2. Evaluate the model on testing part using score and classification_report(y_true, y_pred)

```
In [17]: # Naive Bayes
         gnb = GaussianNB()
         gnb.fit(X_Train,Y_Train)

         # Predicting the Test set result
         Y_Pred = gnb.predict(X_Test)

         # evaluating the model
         print("Gaussian Naive Bayers Accuracy is:",round(accuracy_score(Y_Test,Y_Pred) *
         print("\nClassification Report:\n\n",metrics.classification_report(Y_Test,Y_Pred
```

```
Gaussian Naive Bayers Accuracy is: 46.3

Classification Report:

               precision    recall  f1-score   support

           1       0.32      0.64      0.43        14
           2       0.45      0.21      0.29        24
           3       0.50      0.40      0.44         5
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       1.00      1.00      1.00         7

    accuracy                           0.46        54
   macro avg       0.49      0.54      0.49        54
weighted avg       0.49      0.46      0.44        54
```

After splitting the data we have given training data to the naive bayes model. After that we predicted independent variable using test data and trained naive bayes model. Evaluated the model on testing part using score and classification_report.



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)

In [18]:

```
# SVM model
svc = SVC(kernel='linear')
svc.fit(X_Train, Y_Train)

# Predicting the Test set result
Y_pred = svc.predict(X_Test)

# evaluating the model
print("SVM accuracy is:", round(accuracy_score(Y_Test,Y_pred) * 100, 2))
print("\nClassification Report:\n\n",metrics.classification_report(Y_Test,Y_pred
```

```
SVM accuracy is: 55.56

Classification Report:

               precision    recall  f1-score   support

           1       0.43      0.86      0.57        14
           2       0.60      0.38      0.46        24
           3       0.00      0.00      0.00         5
           5       0.67      1.00      0.80         2
           6       0.00      0.00      0.00         2
           7       1.00      1.00      1.00         7

    accuracy                           0.56        54
   macro avg       0.45      0.54      0.47        54
weighted avg       0.53      0.56      0.51        54
```

After splitting the data we have given training data to the SVM model. After that we predicted independent variable using test data and trained the SVM model. Evaluated the model on the testing part using score and classification_report.