Machine Learning Assignment 4

Name: Akhila Boddu

Student ID: 700742171

GitHub link: https://github.com/AkhilaBoddu/ML-Assignment4.git

Video Link:

https://drive.google.com/file/d/1cuMqA3UU0P9ug2aJ9CdXvPlsLMENwUl6/view?usp=share_link

Question1

Pandas

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.

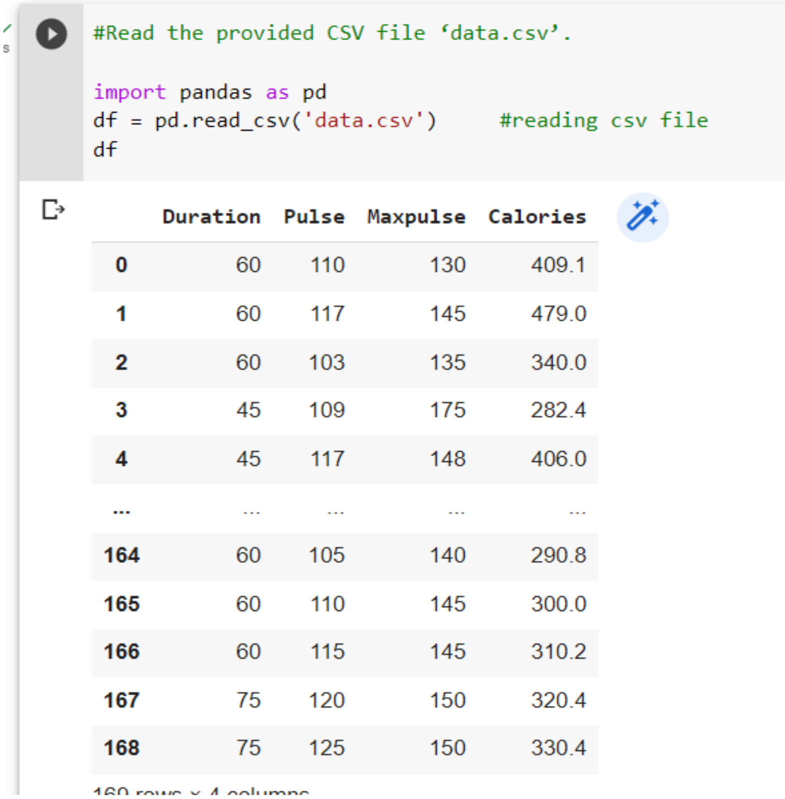
10. Using pandas create a scatter plot for the two columns (Duration and Calories).

<u>Source Code:</u>

2. Pandas

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

Using import keyword, I imported pandas module. read_csv() reads CSV files from the system.



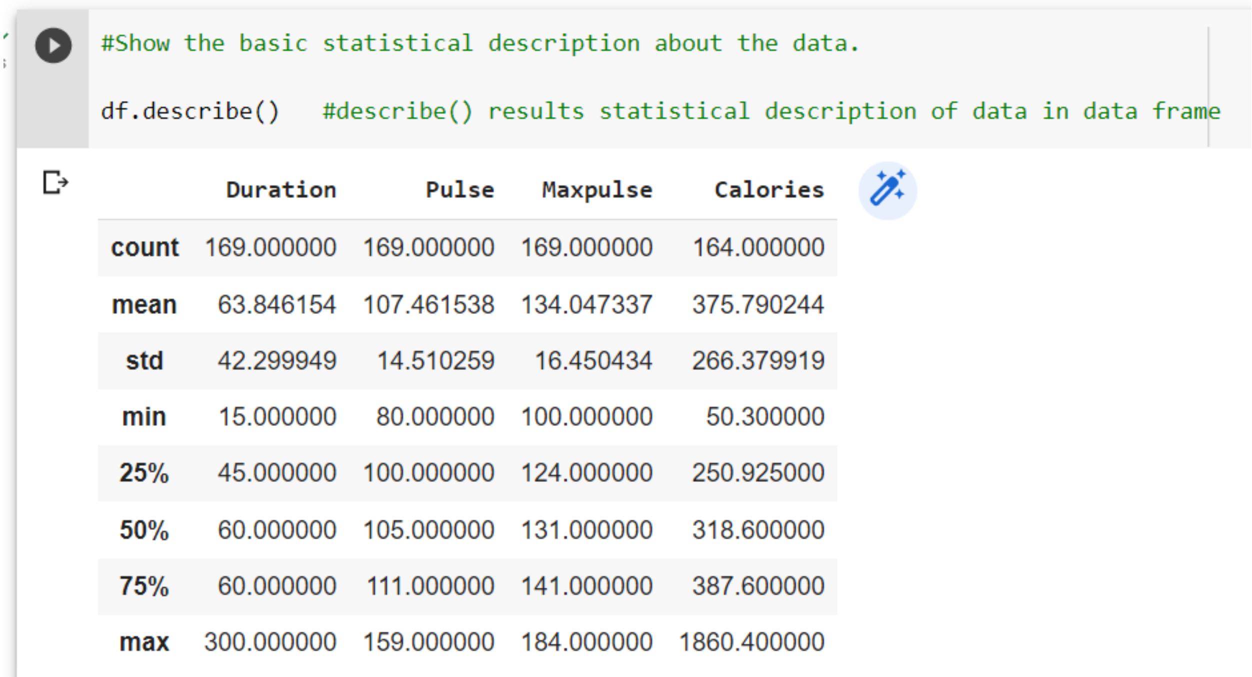
```python
#Read the provided CSV file 'data.csv'.

import pandas as pd
df = pd.read_csv('data.csv')     #reading csv file
df
```

	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
...	...	...	...	...
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

169 rows × 4 columns

2. Show the basic statistical description about the data.

With describe () function from pandas module we get the statistical description of data which is present in data frame. Statistical description contains min, max, count, 1st quantile, mean, median, standard deviation values of columns.



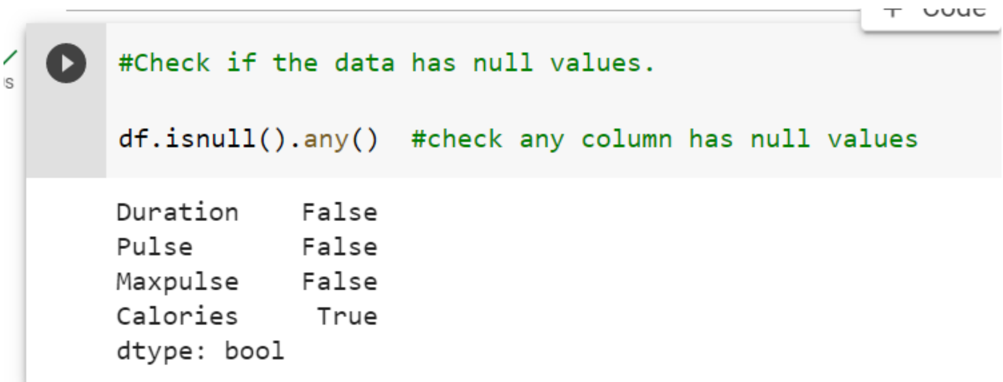
```
#Show the basic statistical description about the data.

df.describe()   #describe() results statistical description of data in data frame
```

	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

3. Check if the data has null values.

To check any null values present in data frame we need to use isnull() function which results a Boolean value. If null values present return true otherwise false.

In data frame we imported contains null values only in ‘Calories’ column.



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

df.isnull().any()  #check any column has null values
```

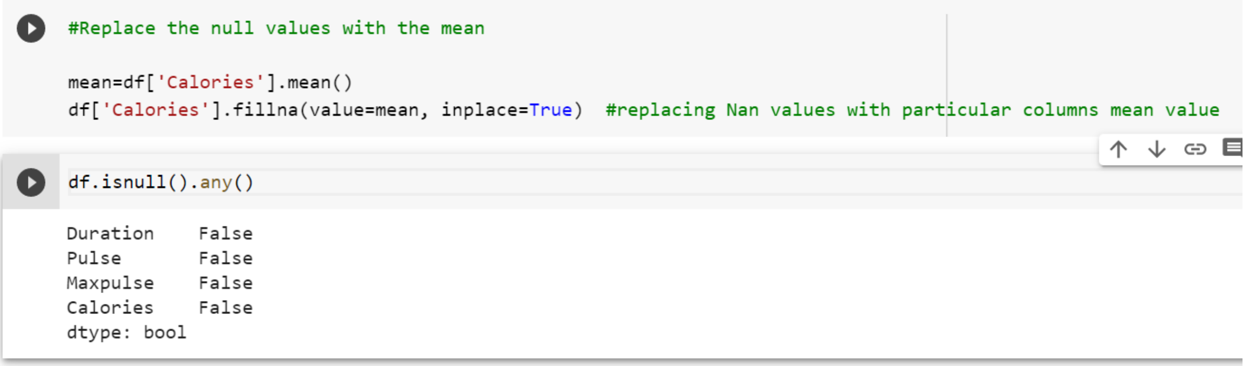
```
Duration     False
Pulse        False
Maxpulse     False
Calories      True
dtype: bool
```

a. Replace the null values with the mean

With fillna() function we can replace null values in a data frame. Null values present in only calories column, so we need to replace those null values with calories column mean value.

Mean() function gives mean value. Using fillna() we can replace null values.

After replacing null values with mean of column, we can see that there are no null values in our data frame.



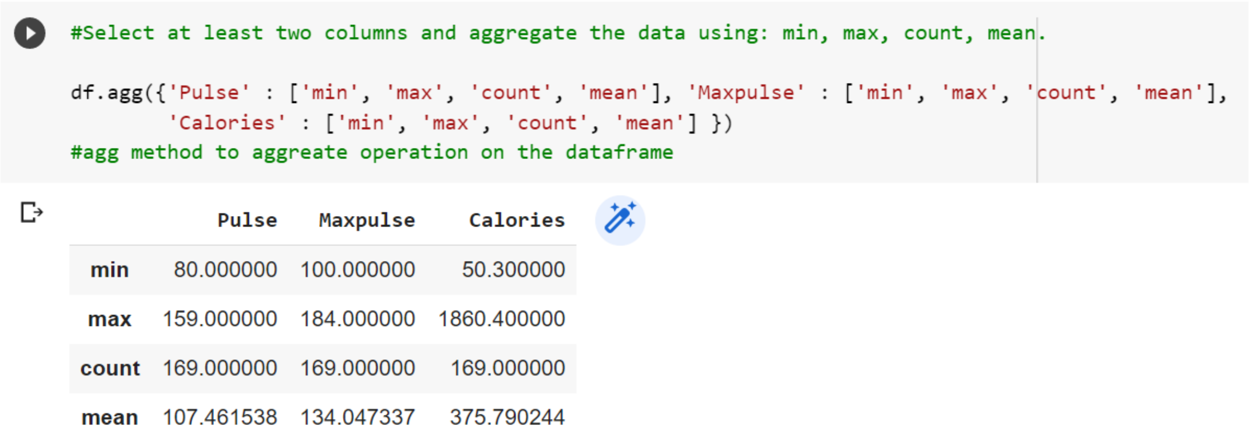
```
#Replace the null values with the mean

mean=df['Calories'].mean()
df['Calories'].fillna(value=mean, inplace=True)  #replacing Nan values with particular columns mean value
```

```
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.

Using agg() method we can apply certain operation on data. Here I applied aggregate functions on three columns like Pulse, Maxpulse and Calories.

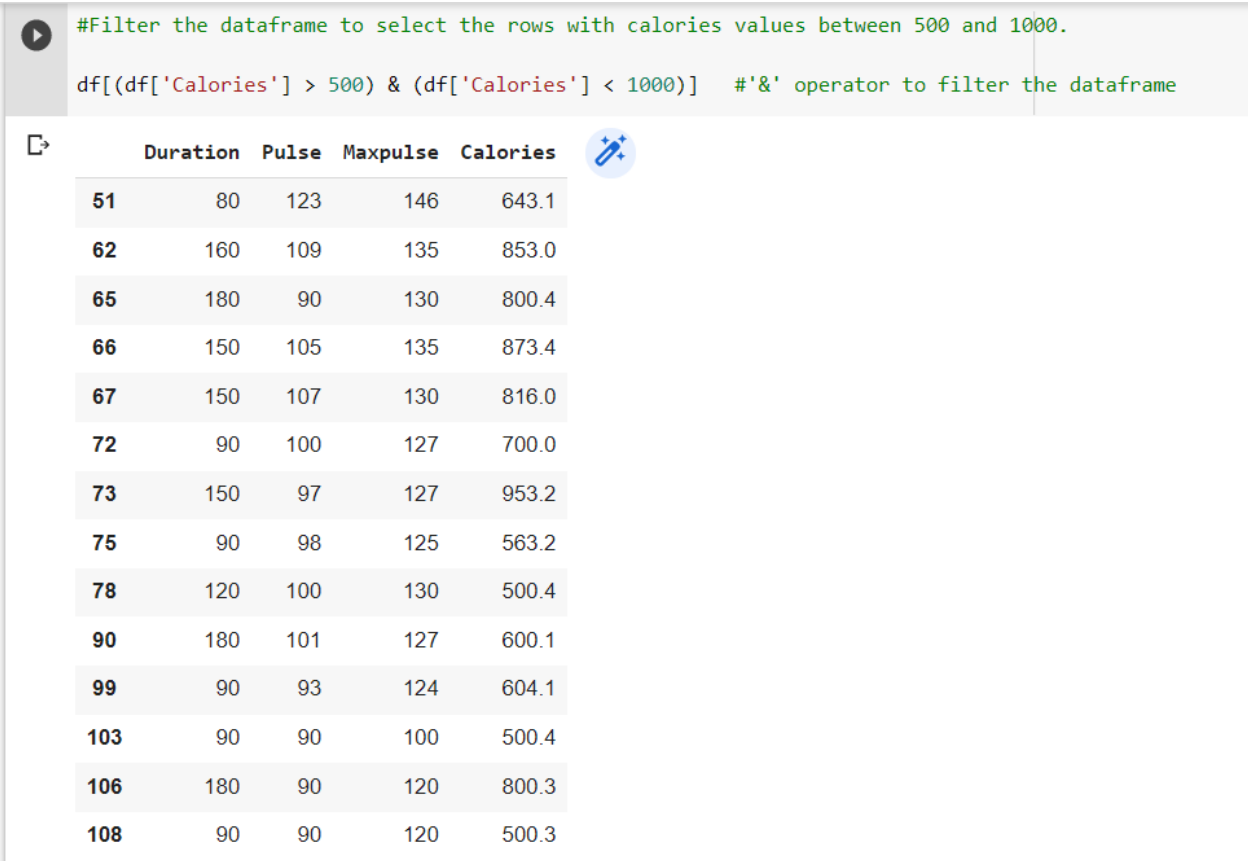


```
#Select at least two columns and aggregate the data using: min, max, count, mean.

df.agg({'Pulse' : ['min', 'max', 'count', 'mean'], 'Maxpulse' : ['min', 'max', 'count', 'mean'],
        'Calories' : ['min', 'max', 'count', 'mean'] })
#agg method to aggreate operation on the dataframe
```

	Pulse	Maxpulse	Calories
min	80.000000	100.000000	50.300000
max	159.000000	184.000000	1860.400000
count	169.000000	169.000000	169.000000
mean	107.461538	134.047337	375.790244

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

Using ‘& ‘operator we can filter the data based on the conditions given. Here I applied ‘&‘ operator on calories column whose values are between 500 and 1000.

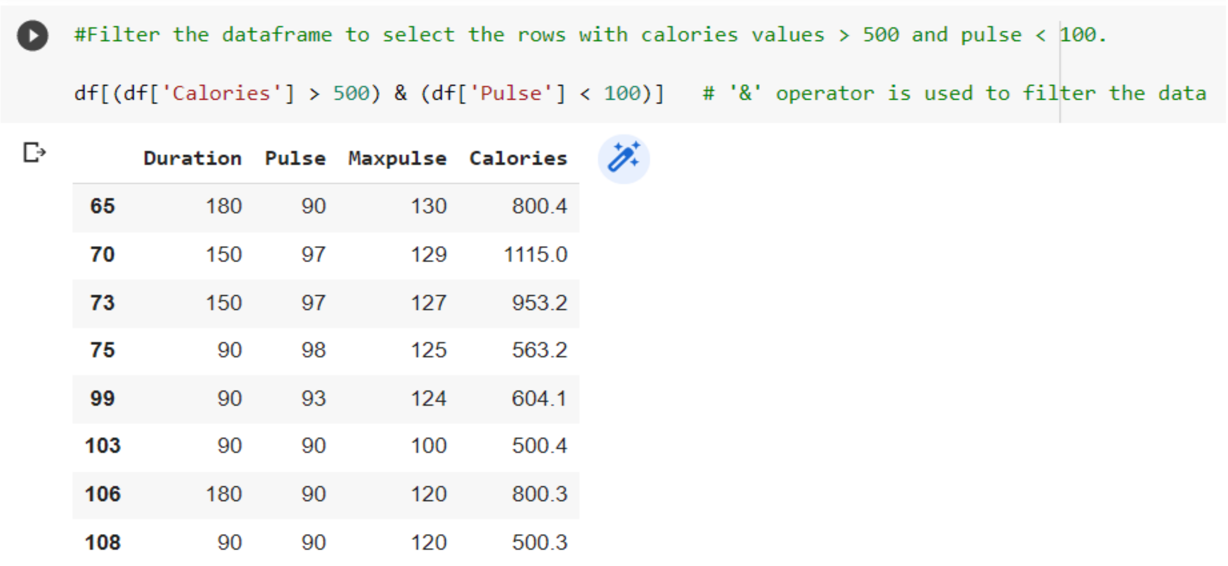


```
#Filter the dataframe to select the rows with calories values between 500 and 1000.

df[(df['Calories'] > 500) & (df['Calories'] < 1000)]   #'&' operator to filter the dataframe
```

```
     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
```

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

Using ‘& ‘operator we can filter the data based on the conditions given. Here I applied ‘&‘ operator on calories column whose values are greater than 500 and in pulse column whose values are less than 100.



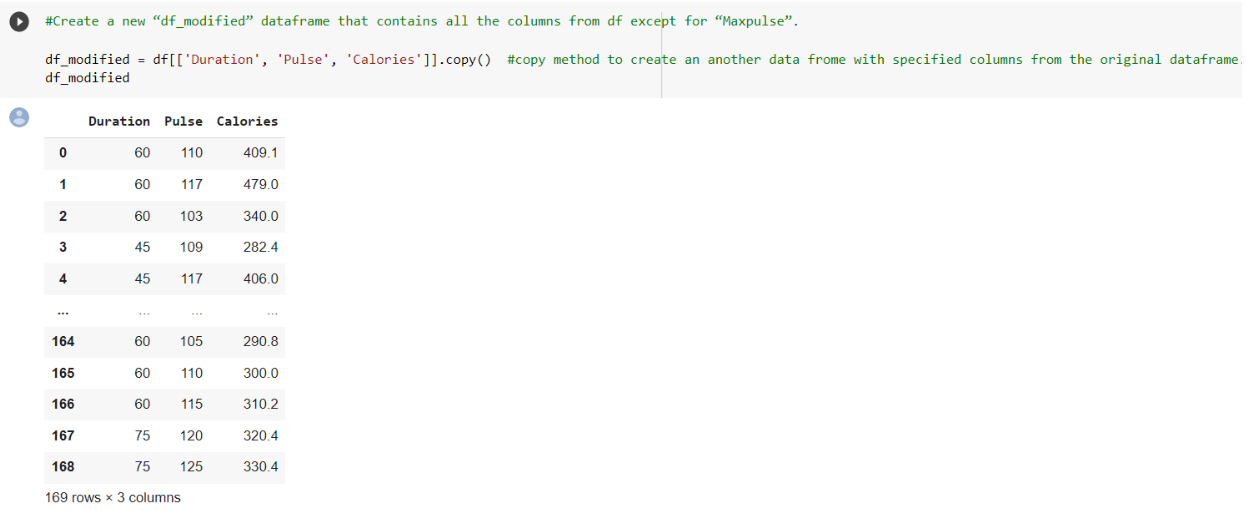
```
#Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

df[(df['Calories'] > 500) & (df['Pulse'] < 100)]   # '&' operator is used to filter the data
```

```
     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”.

Using copy() method we can copy the data from the original data frame to another data frame.

Here I copied the data excluding the data of Maxpulse column.

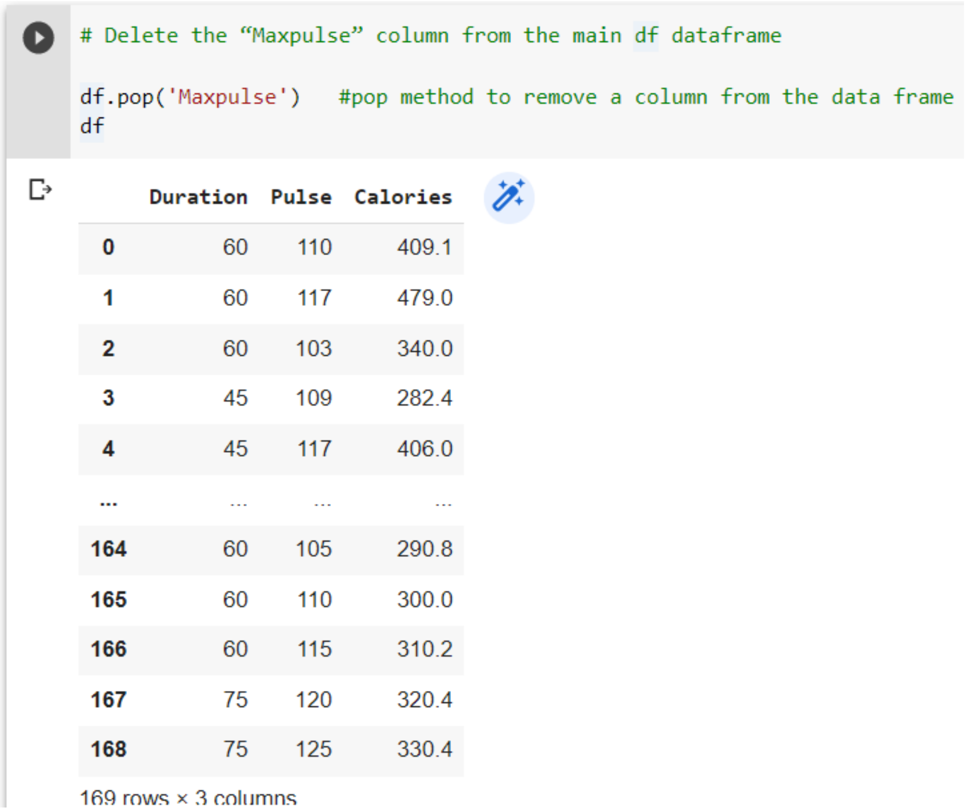


```
#Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.

df_modified = df[['Duration', 'Pulse', 'Calories']].copy()  #copy method to create an another data frome with specified columns from the original dataframe.
df_modified
```

```
     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
```

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

Pop() method can be used to remove a particular column from the data frame. Here pop() is applied on the original data frame to remove Maxpulse column.



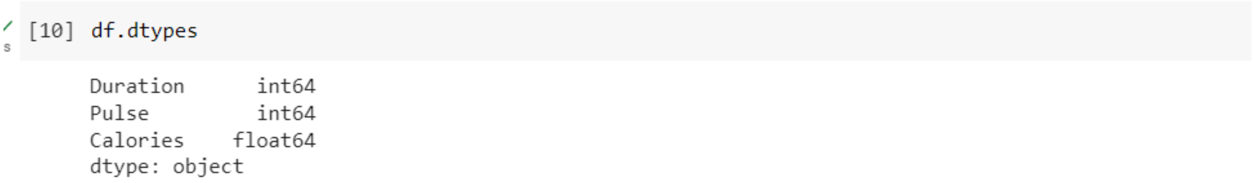
```
# Delete the “Maxpulse” column from the main df dataframe

df.pop('Maxpulse')   #pop method to remove a column from the data frame
df
```

	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

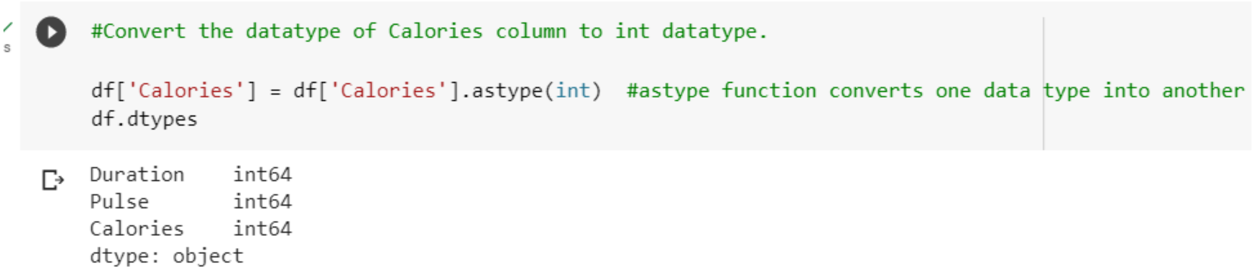
169 rows × 3 columns

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

astype() method to convert one data type to other. Here we can see Calories is of float type and it is being converted to int data type using astype() function.



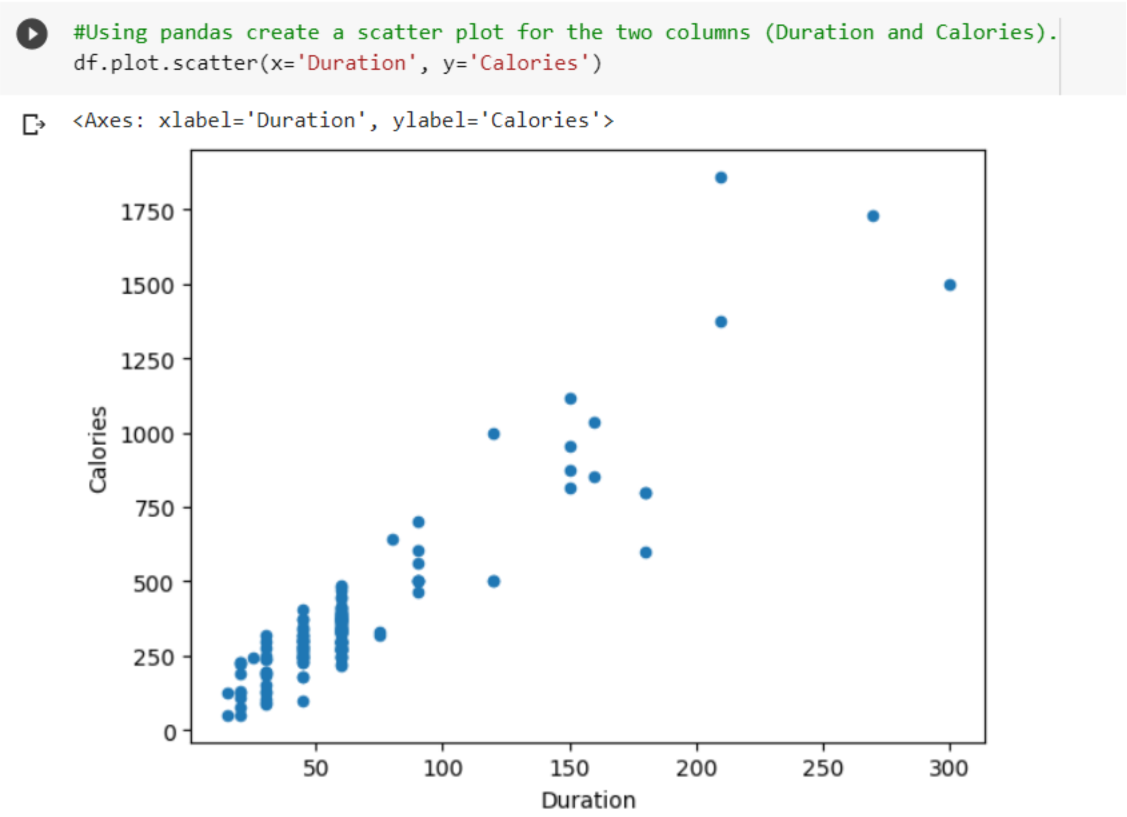
```
[10] df.dtypes

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



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

df['Calories'] = df['Calories'].astype(int)  #astype function converts one data type into another
df.dtypes

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

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

pandas module contains functions to represent the data in visual format. Plot.scatter() method to represent data in scatter plot where duration values lie on x-axis and Calories values on y-axis.

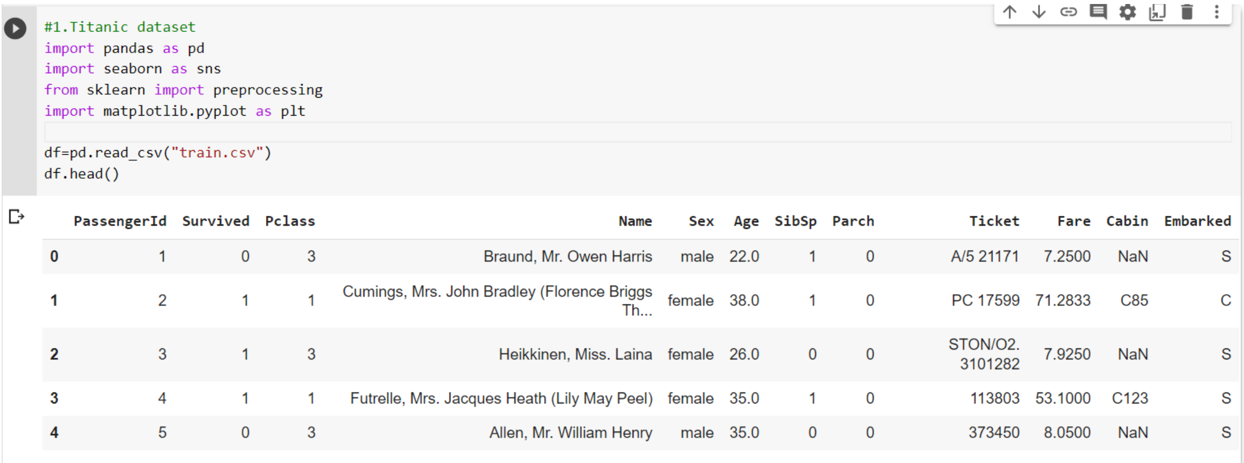


```
#Using pandas create a scatter plot for the two columns (Duration and Calories).
df.plot.scatter(x='Duration', y='Calories')
```

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

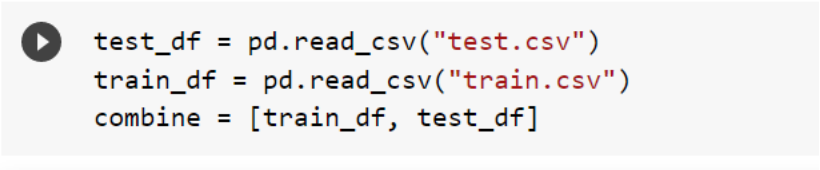
1. (Titanic Dataset)

Using Python NumPy and Pandas libraries, I imported test and train data and combined them into a single dataset.



```python
#1.Titanic dataset
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt

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

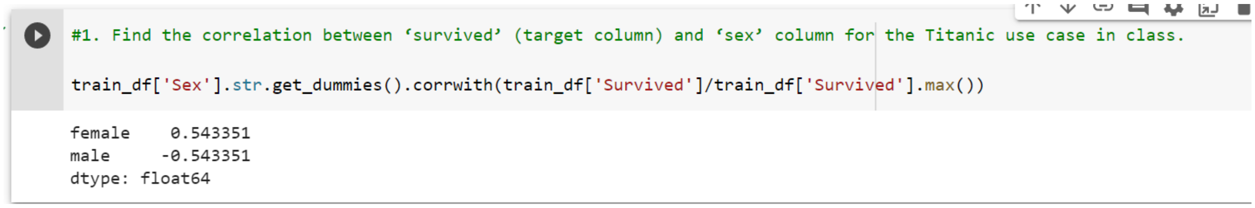
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S



```python
test_df = pd.read_csv("test.csv")
train_df = pd.read_csv("train.csv")
combine = [train_df, test_df]
```

1. Find the correlation between ‘survived’ (target column) and ‘sex’ column for the Titanic use case in class.

As sex column contains string object so we cannot find the correlation between sex column and survived column. So, first we need to convert into type of objects with which we are comparing and find the correlation with survived column.



```python
#1. Find the correlation between ‘survived’ (target column) and ‘sex’ column for the Titanic use case in class.

train_df['Sex'].str.get_dummies().corrwith(train_df['Survived']/train_df['Survived'].max())
```

```
female    0.543351
male     -0.543351
dtype: float64
```

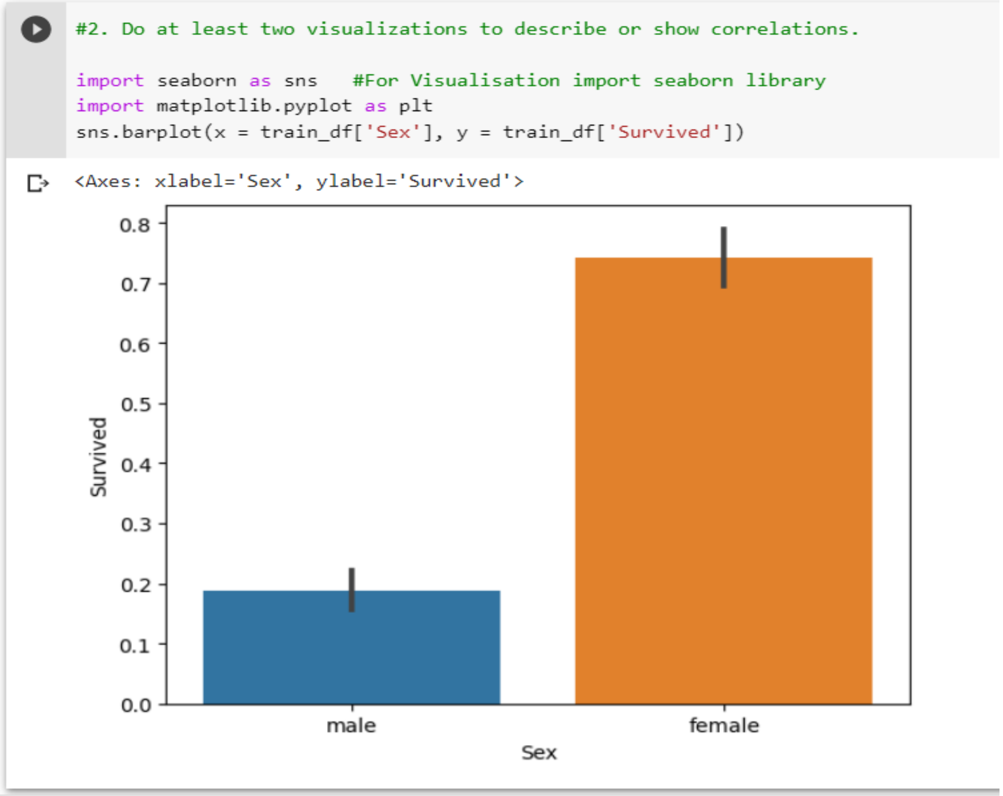
a. Do you think we should keep this feature?

As correlation results shows that males were strongly negatively correlated, and females were Strongly positively correlated with their survival. Males are inversely proportional, and females are directly proportional to their survival. So, we need this feature to analysis.

2. Do at least two visualizations to describe or show correlations.

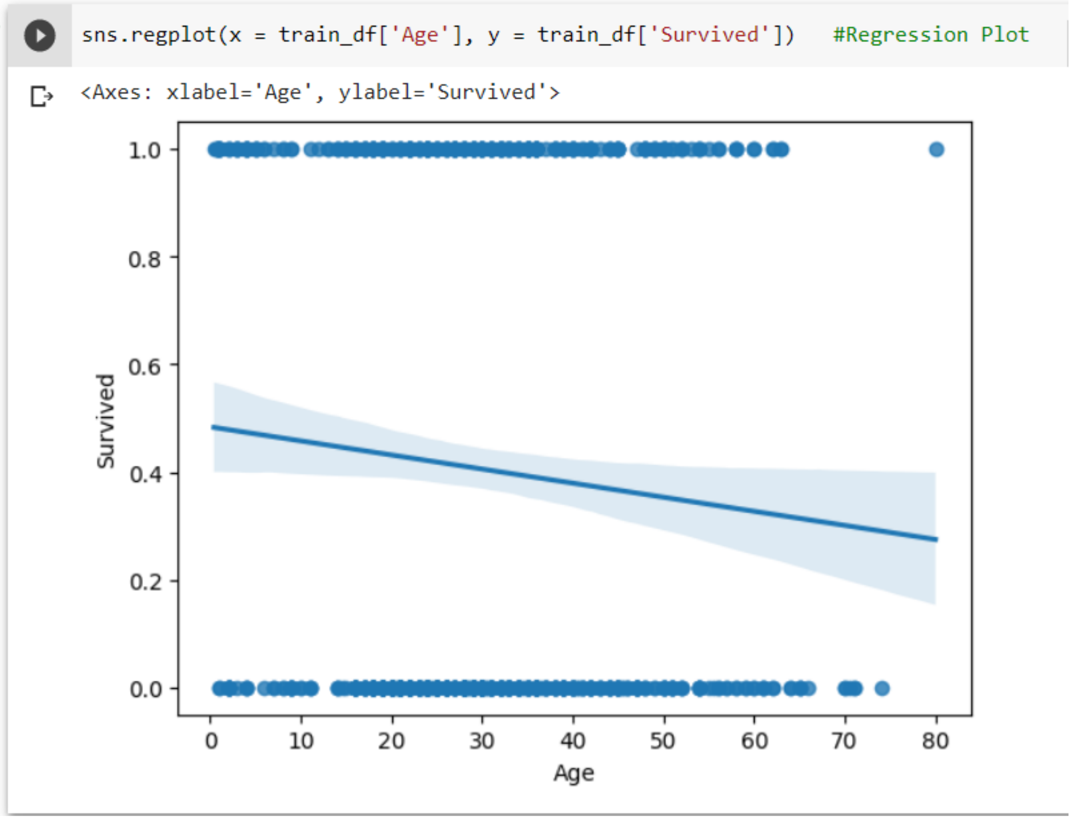
Seaborn library is used to visually show the correlations between the columns data. Here, I am representing correlation between Sex and Survived column using bar plot were Sex on x-axis and survived on y-axis.

Similarly, Regression plot for Age and survived columns and Multi plot grids for Survived and sex columns.



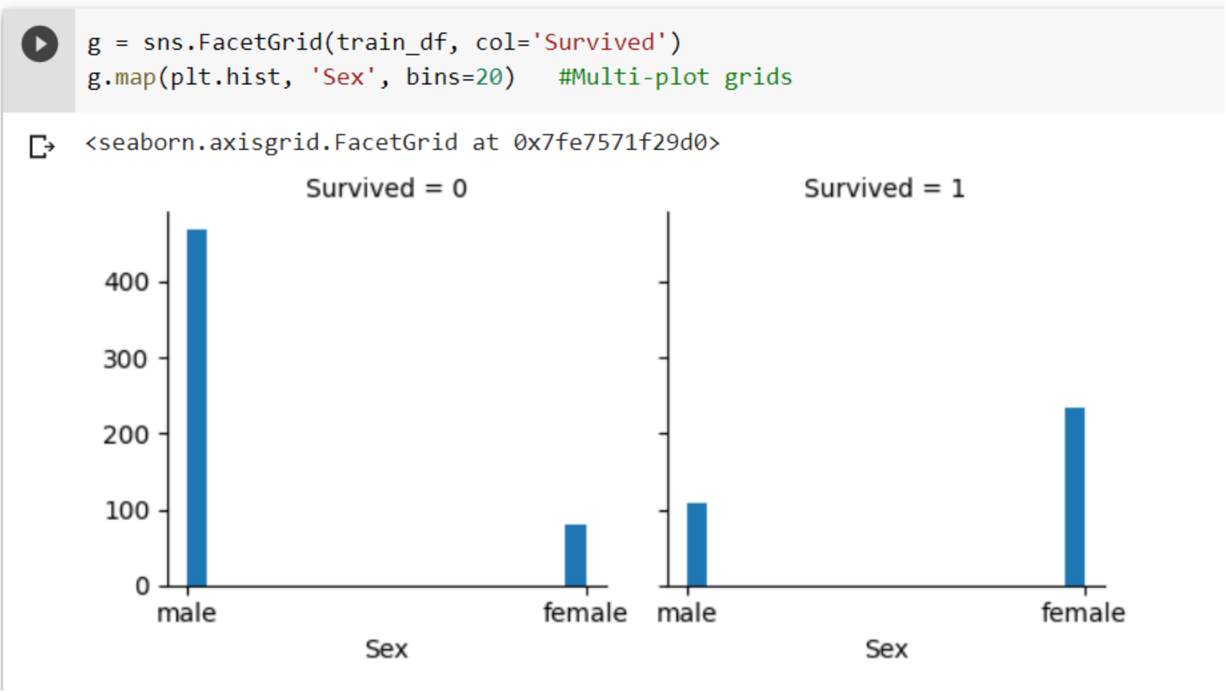
```
#2. Do at least two visualizations to describe or show correlations.

import seaborn as sns   #For Visualisation import seaborn library
import matplotlib.pyplot as plt
sns.barplot(x = train_df['Sex'], y = train_df['Survived'])
```

```
<Axes: xlabel='Sex', ylabel='Survived'>
```



```
sns.regplot(x = train_df['Age'], y = train_df['Survived'])   #Regression Plot
```

```
<Axes: xlabel='Age', ylabel='Survived'>
```



```
g = sns.FacetGrid(train_df, col='Survived')
g.map(plt.hist, 'Sex', bins=20)   #Multi-plot grids
```

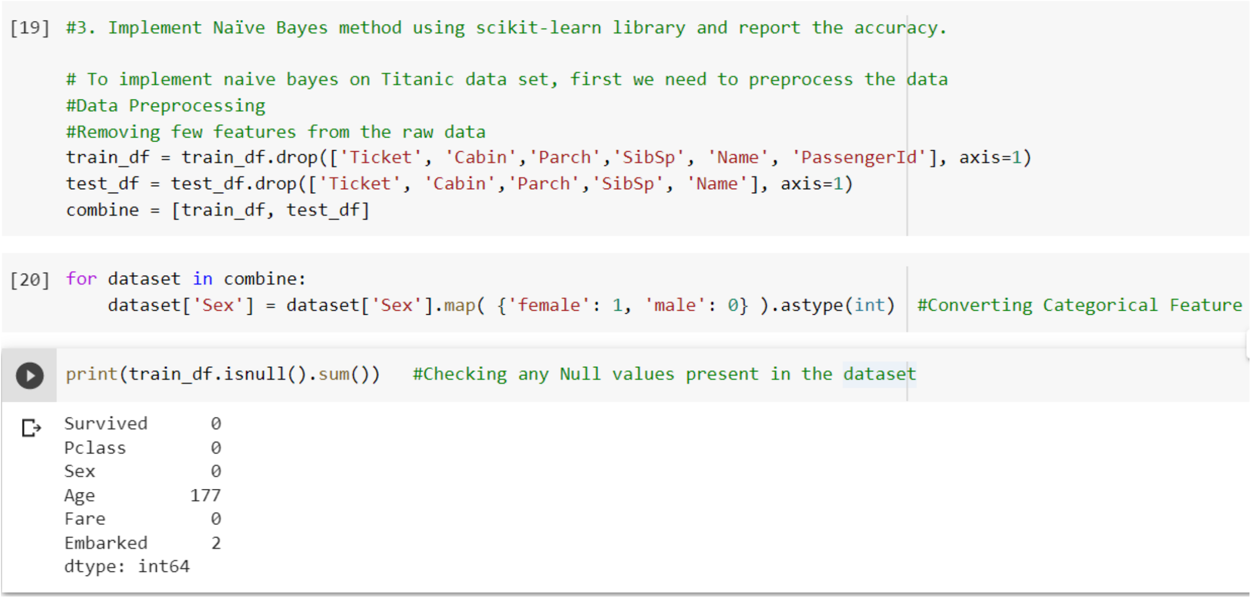
```
<seaborn.axisgrid.FacetGrid at 0x7fe7571f29d0>
```

3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

Before applying machine learning algorithms on data, first we need to preprocess the data to replace null values and to remove any inconsistency present in data.

Here, we converted sex columns features into integers i.e., replacing female with 1 and male as 0. In the age column, there are few missing values and those are replaced with age columns mean value. Similarly, in the fare column missing values are replaced with median value.

In embarked column, I replaced null values with 'S' and those features are converted into integers i.e., S with 0, C with 1, and Q with 2.After completing the preprocessing then we need to apply machine learning algorithms over the data.

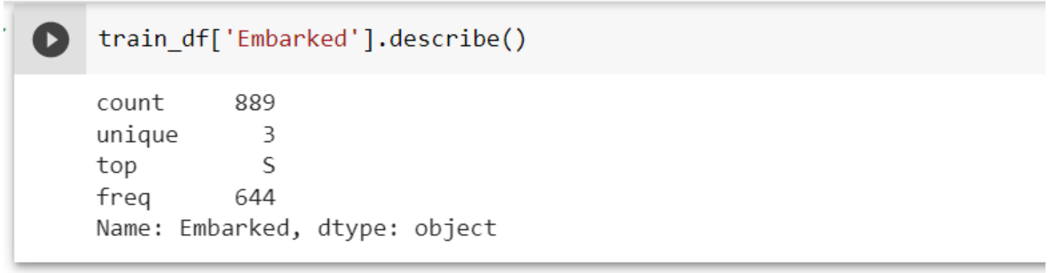


```
[19] #3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

     # To implement naive bayes on Titanic data set, first we need to preprocess the data
     #Data Preprocessing
     #Removing few features from the raw data
     train_df = train_df.drop(['Ticket', 'Cabin','Parch','SibSp', 'Name', 'PassengerId'], axis=1)
     test_df = test_df.drop(['Ticket', 'Cabin','Parch','SibSp', 'Name'], axis=1)
     combine = [train_df, test_df]
```

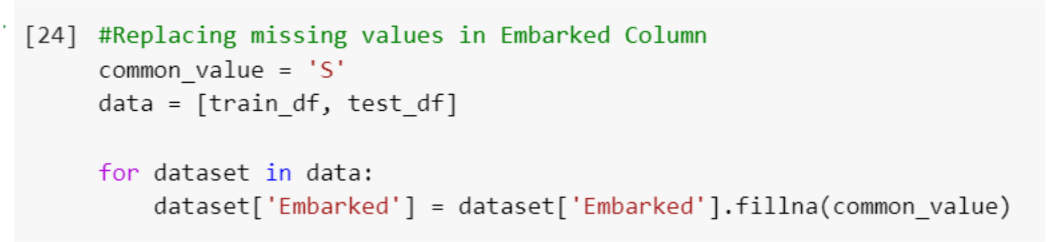
```
[20] for dataset in combine:
         dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(int)  #Converting Categorical Feature
```

```
print(train_df.isnull().sum())   #Checking any Null values present in the dataset
```

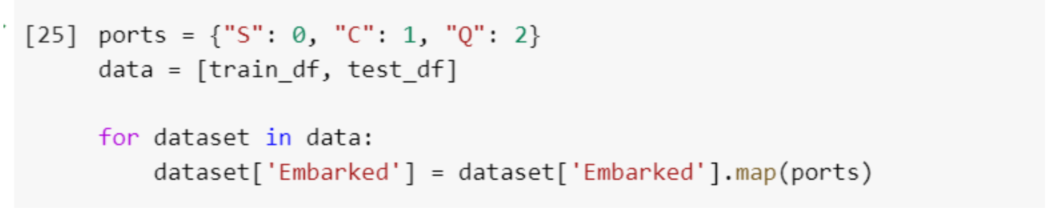
```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
Embarked      2
dtype: int64
```



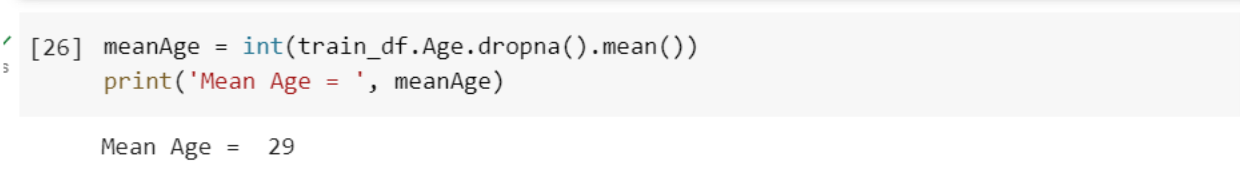
```
train_df['Embarked'].describe()

count      889
unique       3
top          S
freq       644
Name: Embarked, dtype: object
```



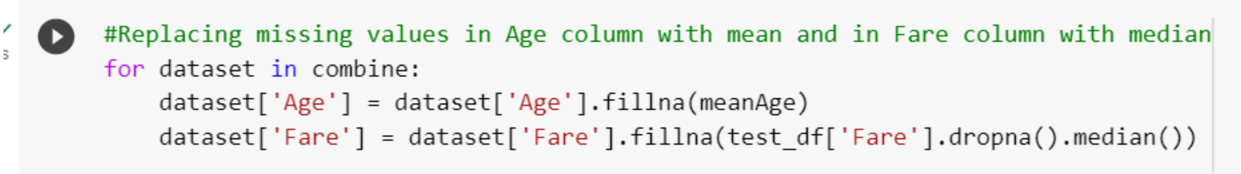
```
[24] #Replacing missing values in Embarked Column
     common_value = 'S'
     data = [train_df, test_df]

     for dataset in data:
         dataset['Embarked'] = dataset['Embarked'].fillna(common_value)
```



```
[25] ports = {"S": 0, "C": 1, "Q": 2}
     data = [train_df, test_df]

     for dataset in data:
         dataset['Embarked'] = dataset['Embarked'].map(ports)
```



```
[26] meanAge = int(train_df.Age.dropna().mean())
     print('Mean Age = ', meanAge)

     Mean Age =  29
```



```
#Replacing missing values in Age column with mean and in Fare column with median
for dataset in combine:
    dataset['Age'] = dataset['Age'].fillna(meanAge)
    dataset['Fare'] = dataset['Fare'].fillna(test_df['Fare'].dropna().median())
```

Here, I applied Naïve Bayes Algorithm on the preprocessed data using sklearn library. Python sklearn library contains many machine learning algorithms to analyze the data.

In the given data, there are no labels present in the test data set to compare with our predicted data. So, we need to use the training data set to compare with our predicted data set using Naïve bayes algorithm.

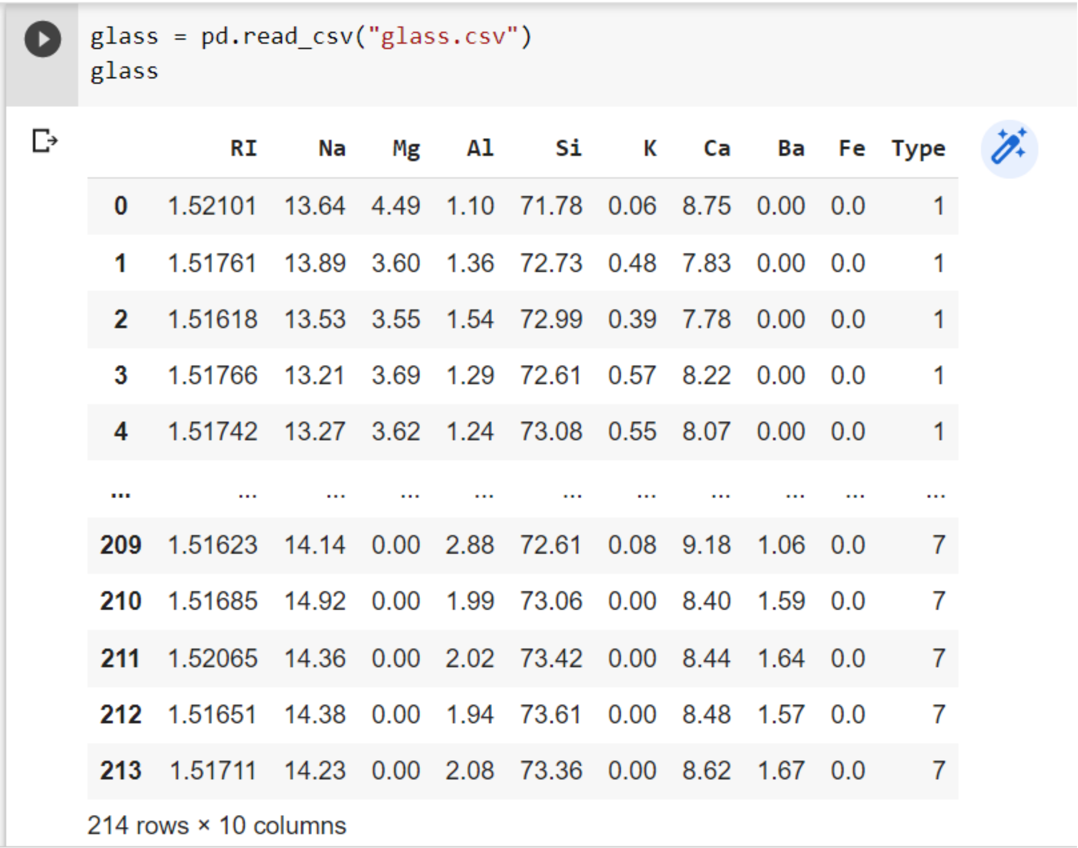
Using accuracy, we can compare with other machine learning algorithms to find which method is performing better on this data set.

2. (Glass Dataset)

1. Implement Naïve Bayes method using scikit-learn library.

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

Using read_csv method from pandas module I imported glass data set.



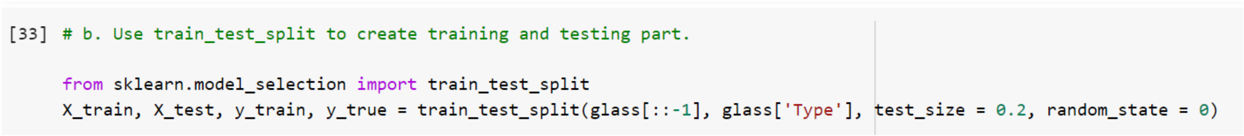
```
glass = pd.read_csv("glass.csv")
glass
```

	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	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

b. Use train_test_split to create the training and testing part.

sklearn module contains train_test_split method to split our data set into training and testing data sets. In this data set Type column can be used for labels. In this method, test_size defines how much proportion of data to be in the test data set. When we test_size value whole analysis results will change.

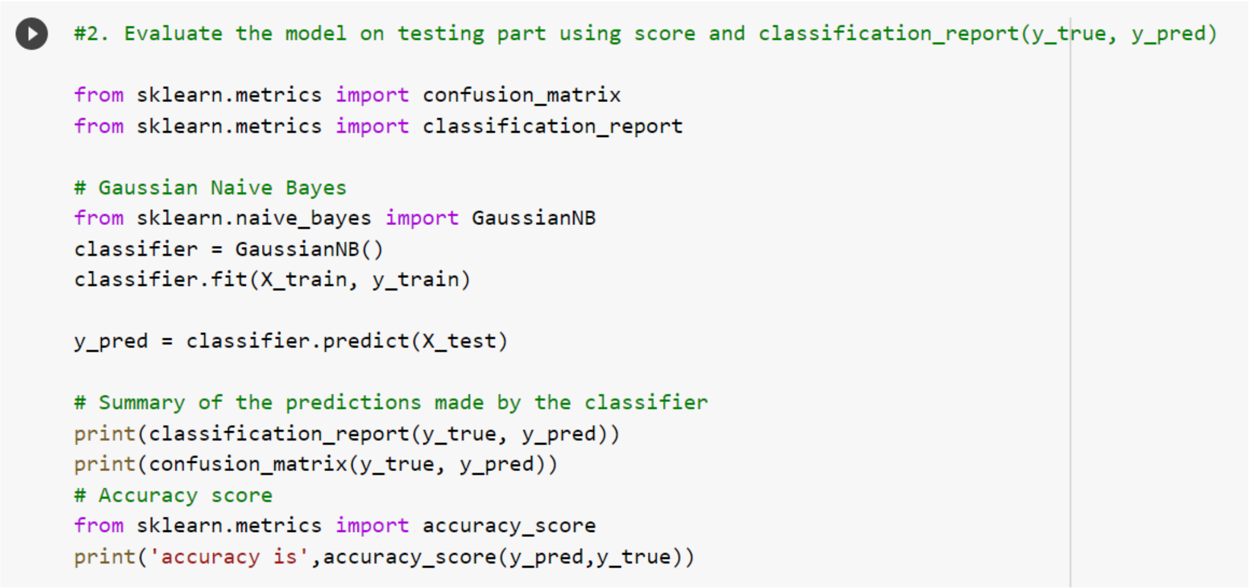


```
[33] # b. Use train_test_split to create training and testing part.

     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_true = train_test_split(glass[::-1], glass['Type'], test_size = 0.2, random_state = 0)
```

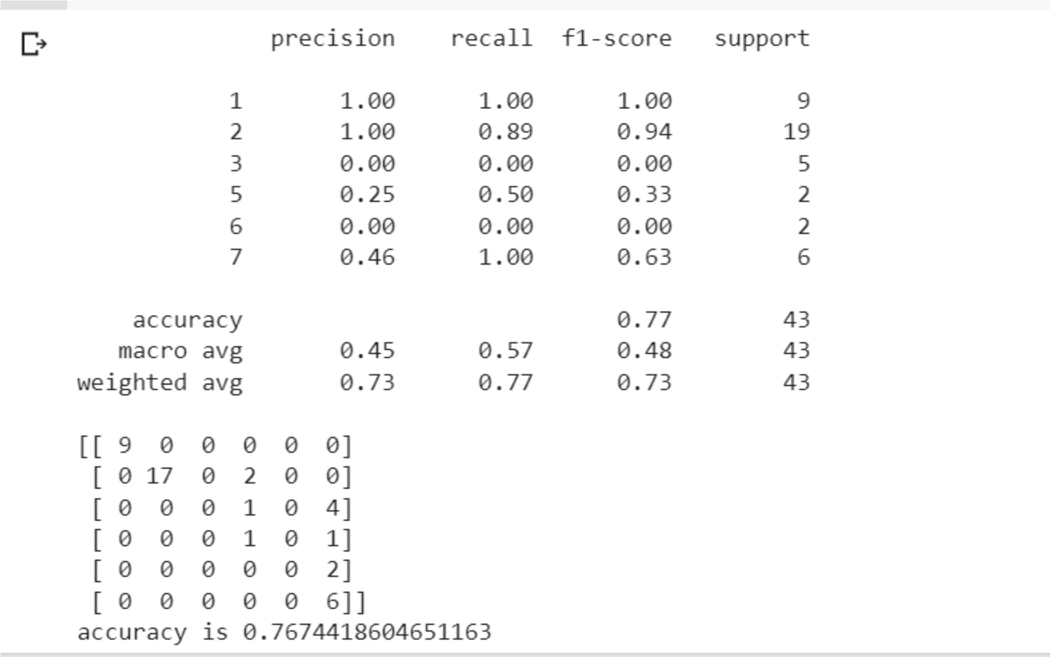
2. Evaluate the model on testing part using score and classification_report(y_true, y_pred)

In the given data, there are no missing values present in it. So, we can directly apply the machine learning algorithms on the data.

sklearn module is imported to analyze the data using different algorithms. Classification_report and confusion_matrix methods to result the summary of the predictions made using the specific algorithm. These summaries can be used to compare with another algorithms to define which algorithm is better. Naïve bayes on this data set results 77% of accuracy.



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

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_true, y_pred))
print(confusion_matrix(y_true, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_true))
```

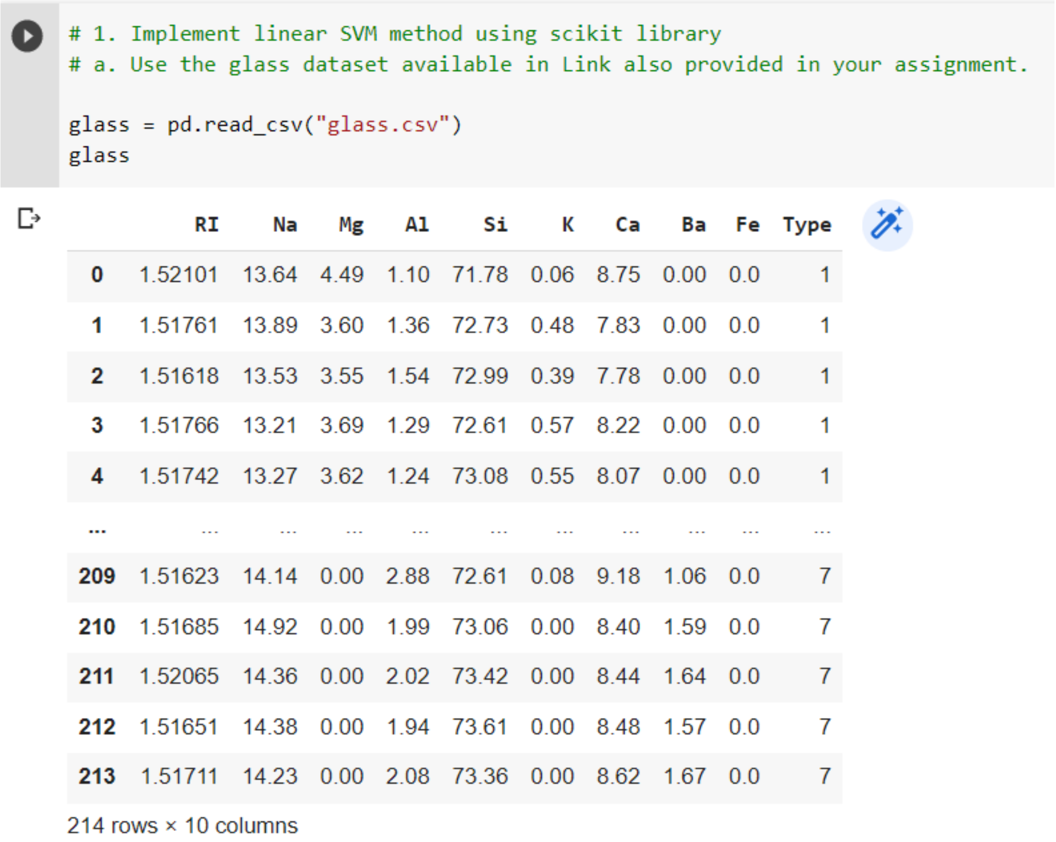


```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         9
           2       1.00      0.89      0.94        19
           3       0.00      0.00      0.00         5
           5       0.25      0.50      0.33         2
           6       0.00      0.00      0.00         2
           7       0.46      1.00      0.63         6

    accuracy                           0.77        43
   macro avg       0.45      0.57      0.48        43
weighted avg       0.73      0.77      0.73        43

[[ 9  0  0  0  0  0]
 [ 0 17  0  2  0  0]
 [ 0  0  0  1  0  4]
 [ 0  0  0  1  0  1]
 [ 0  0  0  0  0  2]
 [ 0  0  0  0  0  6]]
accuracy is 0.7674418604651163
```

1. Implement linear SVM method using scikit library

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

Using read_csv method from pandas module I imported glass data set.



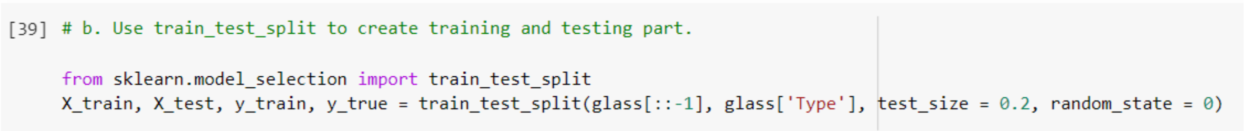
```
# 1. Implement linear SVM method using scikit library
# a. Use the glass dataset available in Link also provided in your assignment.

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

	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	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

b. Use train_test_split to create the training and testing part.

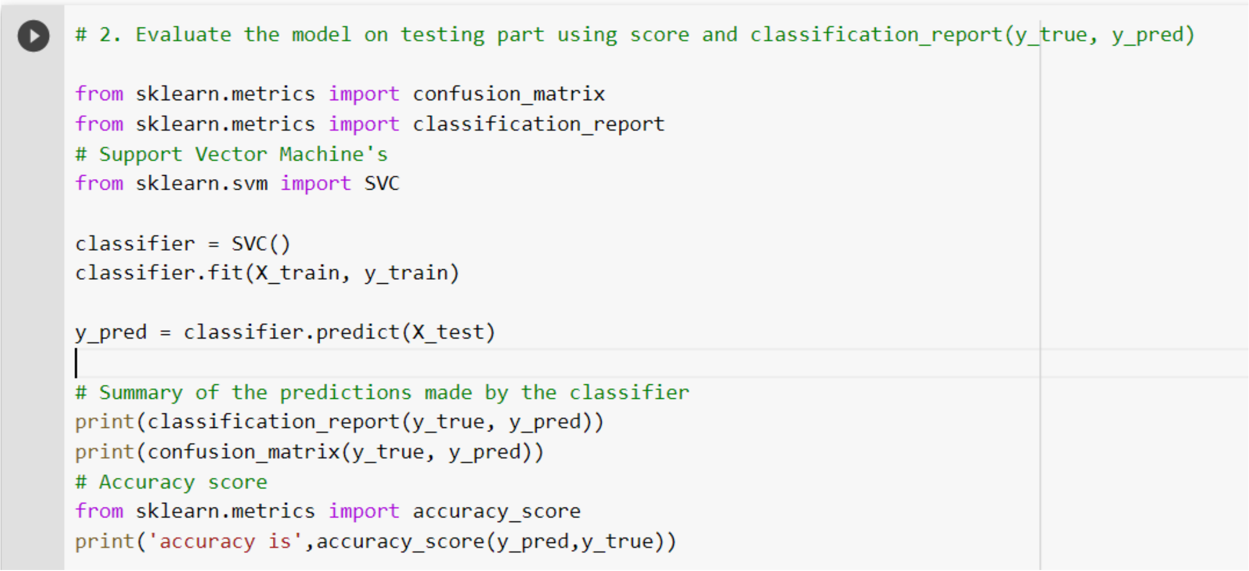
sklearn module contains train_test_split method to split our data set into training and testing data sets. In this data set Type column can be used for labels. In this method, test_size defines how much proportion of data to be in the test data set. When we test_size value whole analysis results will change.



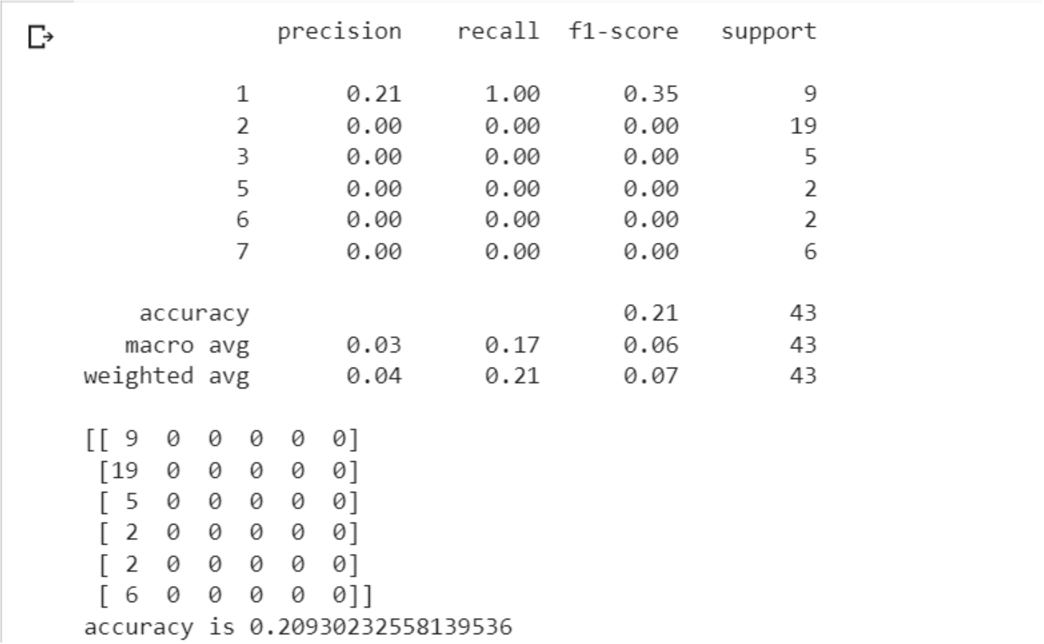
```
[39] # b. Use train_test_split to create training and testing part.

     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_true = train_test_split(glass[::-1], glass['Type'], test_size = 0.2, random_state = 0)
```

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

Support vector machine algorithm is applied to this data set using sklearn module. We got an accuracy of 21% using SVM.



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

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# Support Vector Machine's
from sklearn.svm import SVC

classifier = SVC()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_true, y_pred))
print(confusion_matrix(y_true, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_true))
```

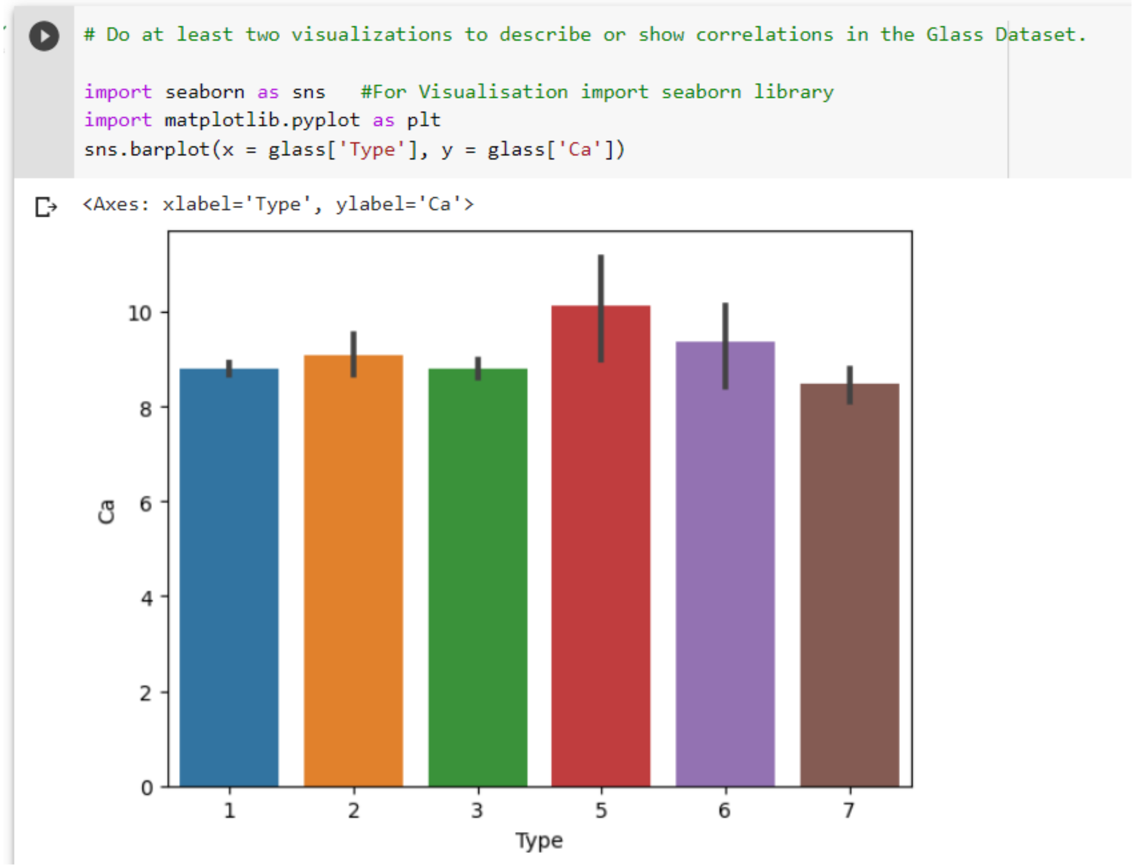


```
              precision    recall  f1-score   support

           1       0.21      1.00      0.35         9
           2       0.00      0.00      0.00        19
           3       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         2
           7       0.00      0.00      0.00         6

    accuracy                           0.21        43
   macro avg       0.03      0.17      0.06        43
weighted avg       0.04      0.21      0.07        43

[[ 9  0  0  0  0  0]
 [19  0  0  0  0  0]
 [ 5  0  0  0  0  0]
 [ 2  0  0  0  0  0]
 [ 2  0  0  0  0  0]
 [ 6  0  0  0  0  0]]
accuracy is 0.20930232558139536
```

Do at least two visualizations to describe or show correlations in the Glass Dataset.

Seaborn library is used to visually show the correlations between the columns data. Here, I am representing correlation between Type and Ca column using bar plot where Type on x-axis and Ca on y-axis.

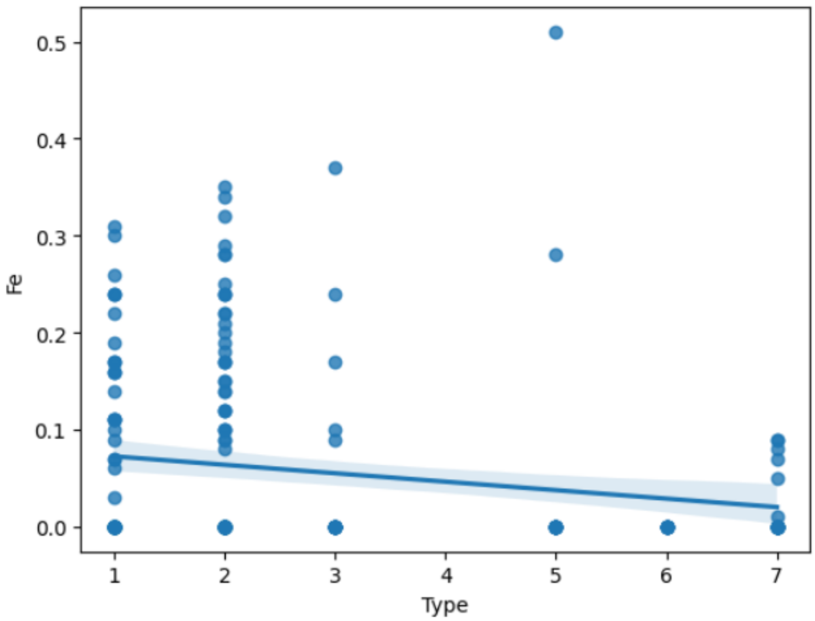
Similarly, Regression plot for Type and Fe columns and categorized plot for Type and K columns.



```
# Do at least two visualizations to describe or show correlations in the Glass Dataset.

import seaborn as sns   #For Visualisation import seaborn library
import matplotlib.pyplot as plt
sns.barplot(x = glass['Type'], y = glass['Ca'])
```

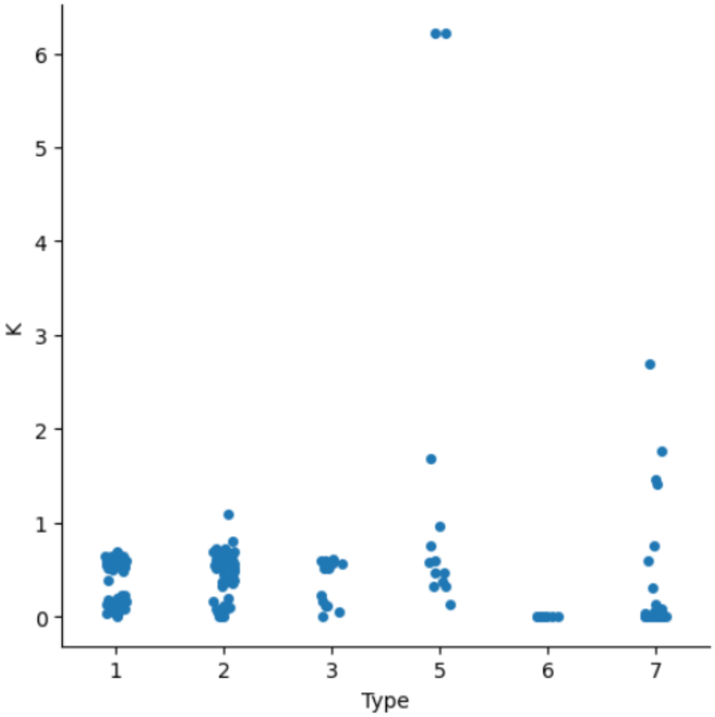
```
<Axes: xlabel='Type', ylabel='Ca'>
```

```
sns.regplot(x="Type", y="Fe", data=glass);
```



```
sns.catplot(data=glass, x="Type", y="K")
```

```
<seaborn.axisgrid.FacetGrid at 0x7fe755fa03a0>
```



Which algorithm got better accuracy? Can you justify why?

Among Naïve Bayes and Support vector machine algorithms, naïve bayes got better accuracy than the SVM. Naïve Bayes gives better results than SVM for this data set. we may get better results using SVM than naïve bayes when we work with another data set. In this glass data set, types of glass are independent predictors. When there are any independent predictors present in the data set naïve bayes perform better than other models.