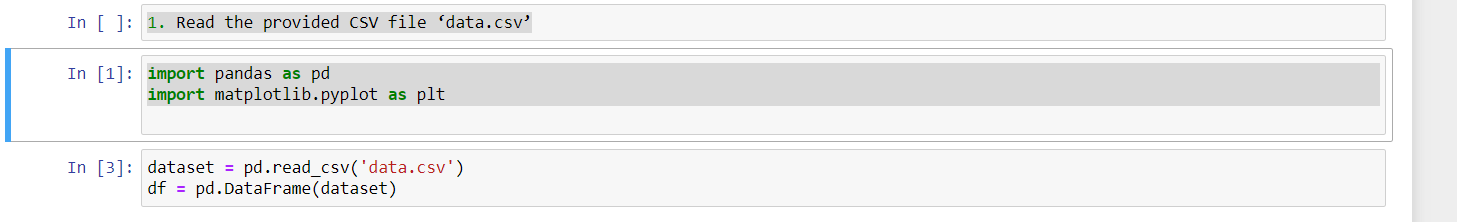
MACHINE LEARNING ASSIGNMENT-2

Video link-

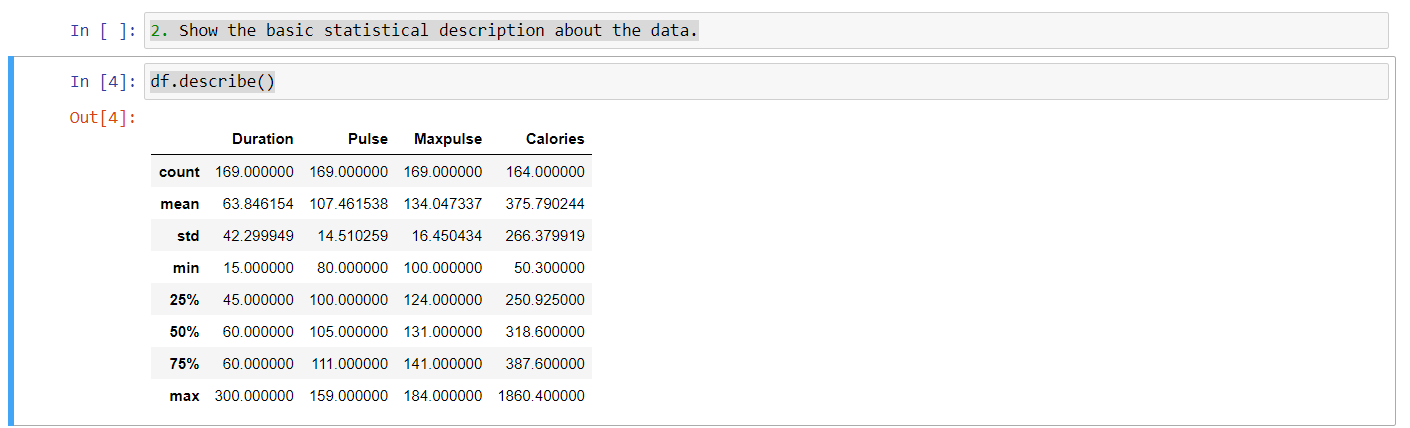
<https://drive.google.com/file/d/1D_sC2dF_X_Fjg4764Vhm9wD8sxhme2Rf/view?usp=sharing>

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



1. Show the basic statistical description about the data.

df.describe()



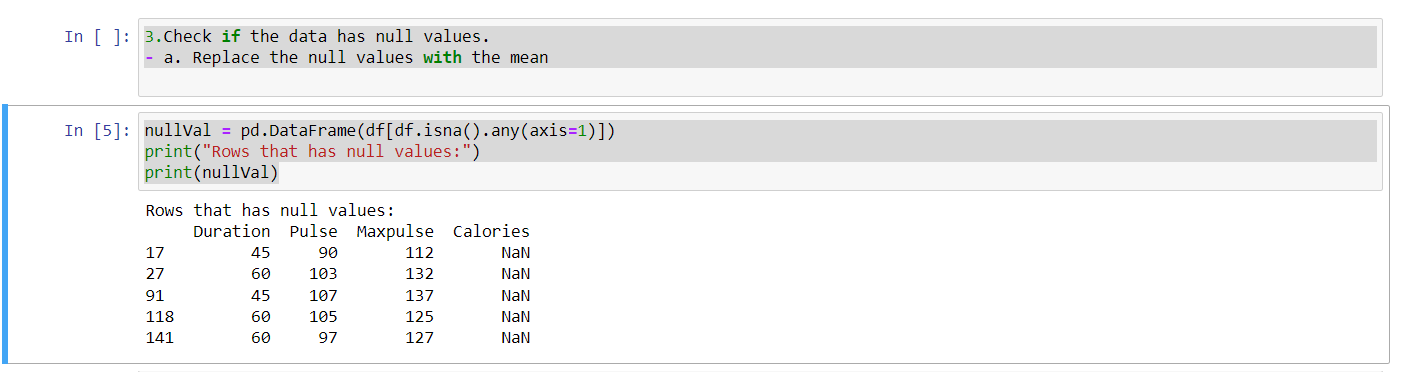
3.Check if the data has null values.

- a. Replace the null values with the mean

nullVal = pd.DataFrame(df[df.isna().any(axis=1)])

print("Rows that has null values:")

print(nullVal)



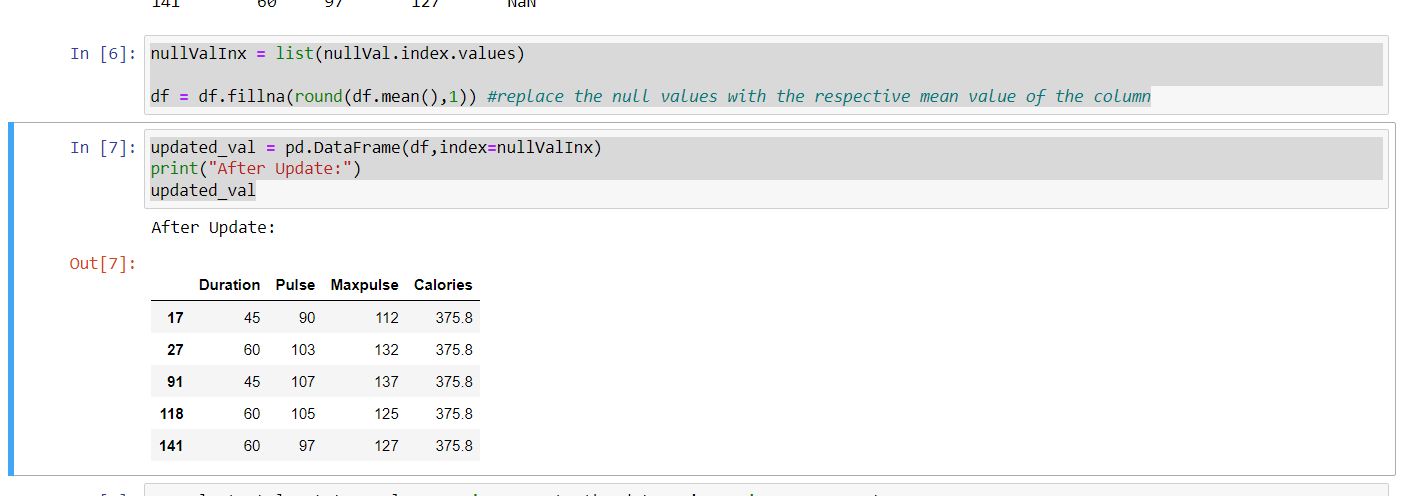
nullValInx = list(nullVal.index.values)

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

updated\_val = pd.DataFrame(df,index=nullValInx)

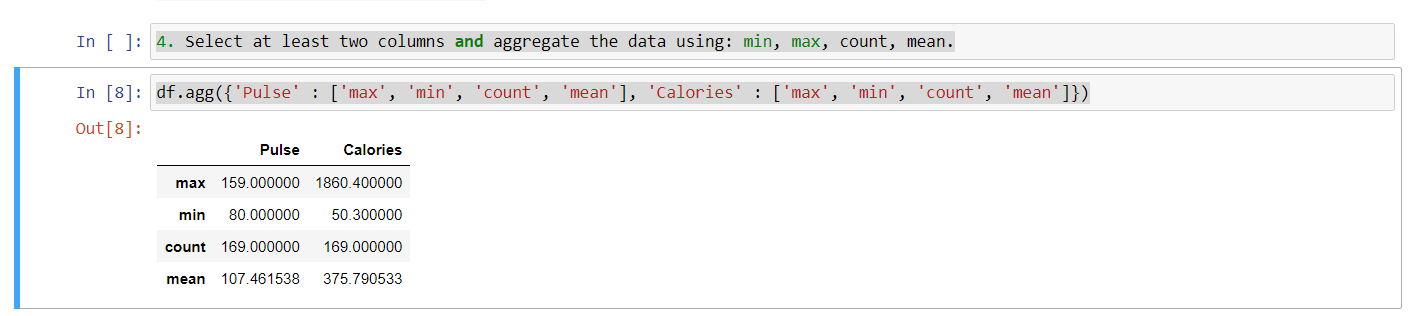
print("After Update:")

updated\_val



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

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

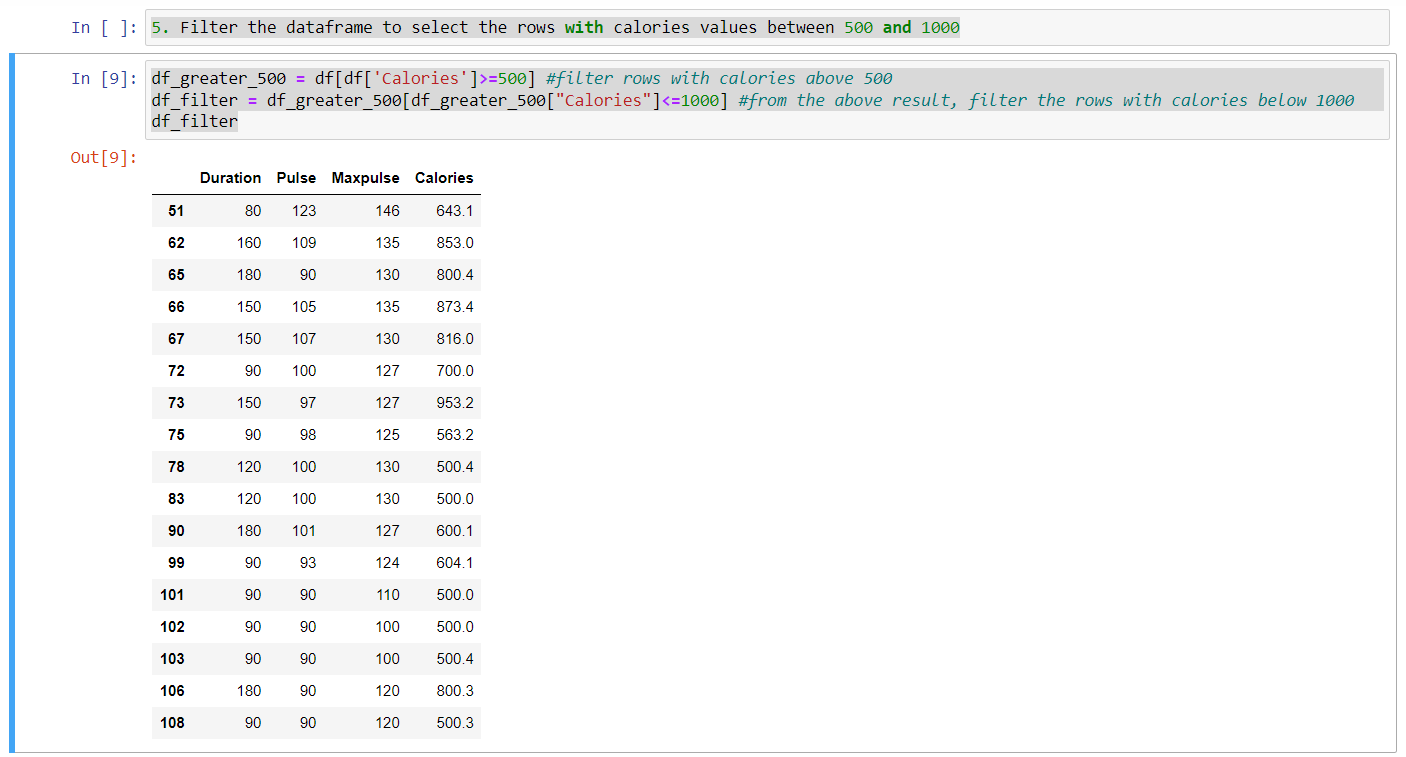


5. Filter the dataframe to select the rows with calories values between 500 and 1000

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 result, filter the rows with calories below 1000

df\_filter

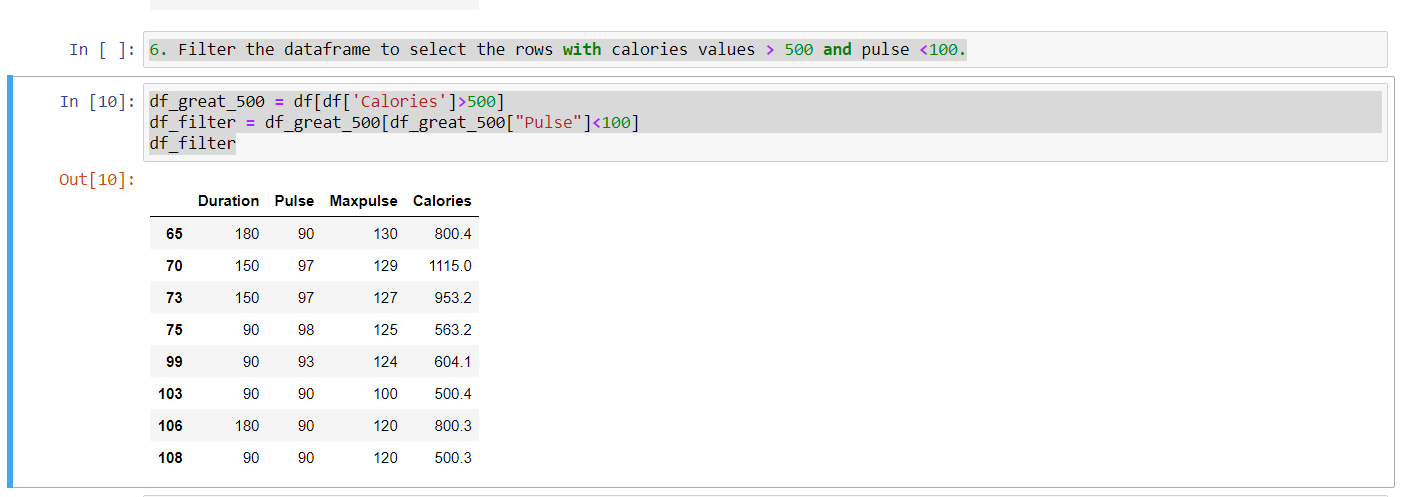


6. Filter the dataframe to select the rows with calories values > 500 and pulse <100.

df\_great\_500 = df[df['Calories']>500]

df\_filter = df\_great\_500[df\_great\_500["Pulse"]<100]

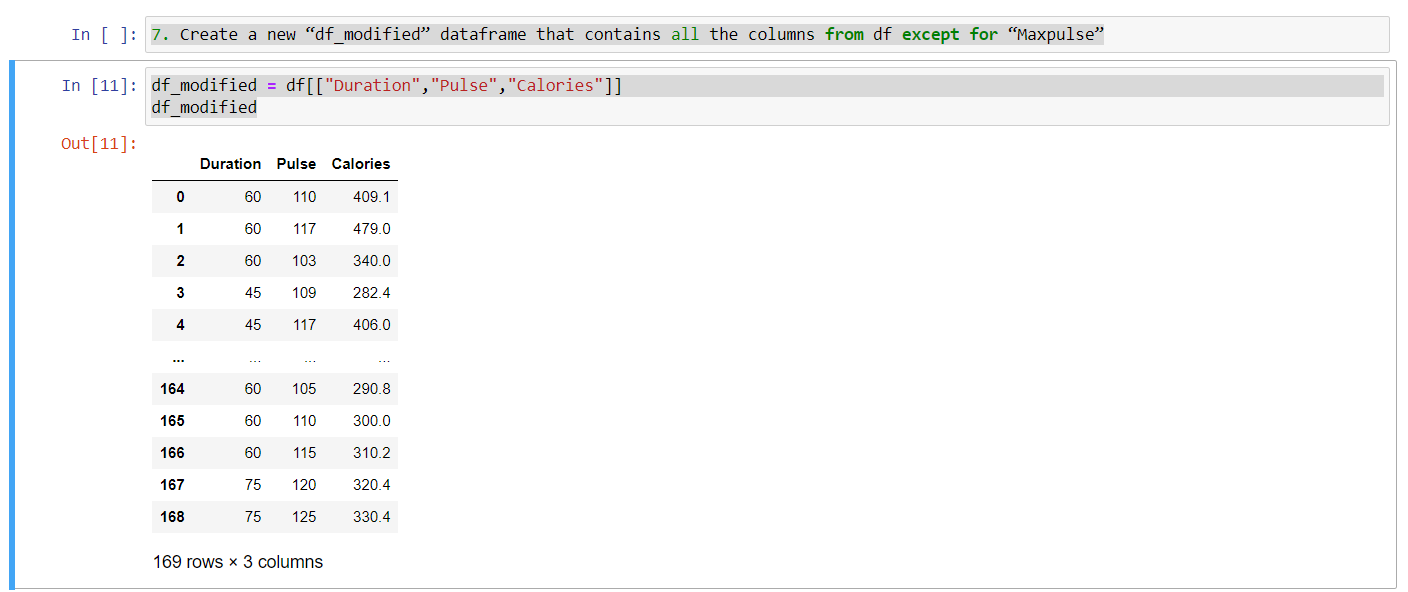
df\_filter



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

df\_modified = df[["Duration","Pulse","Calories"]]

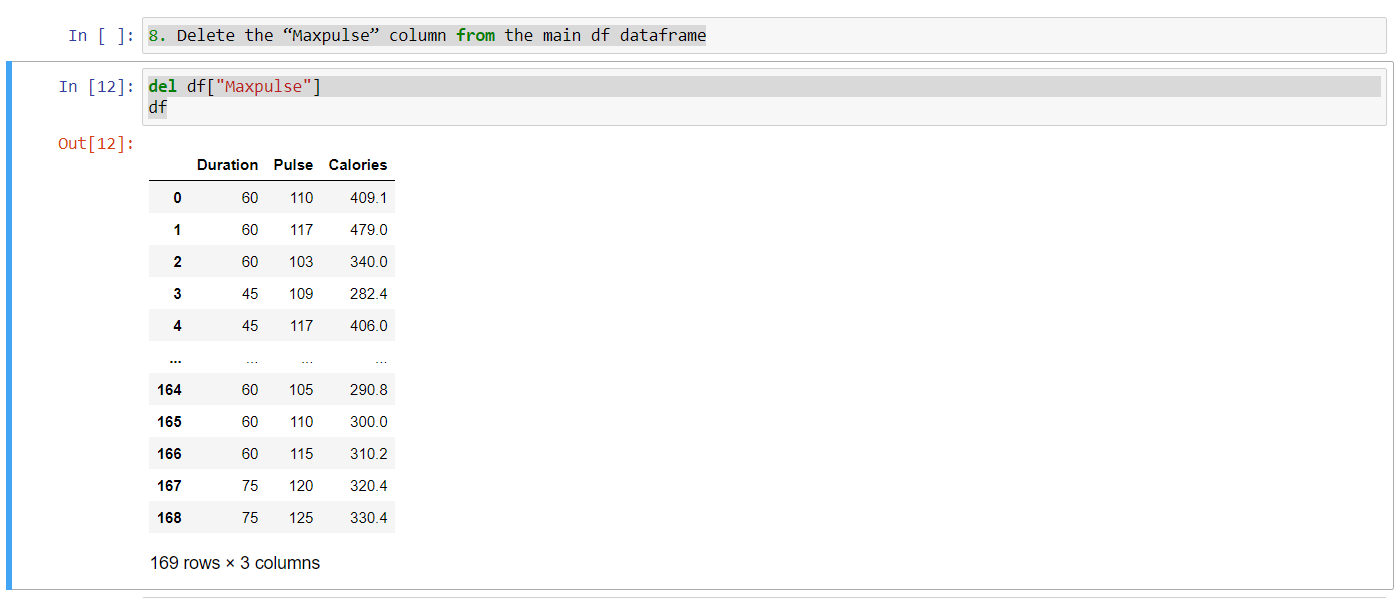
df\_modified



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

del df["Maxpulse"]

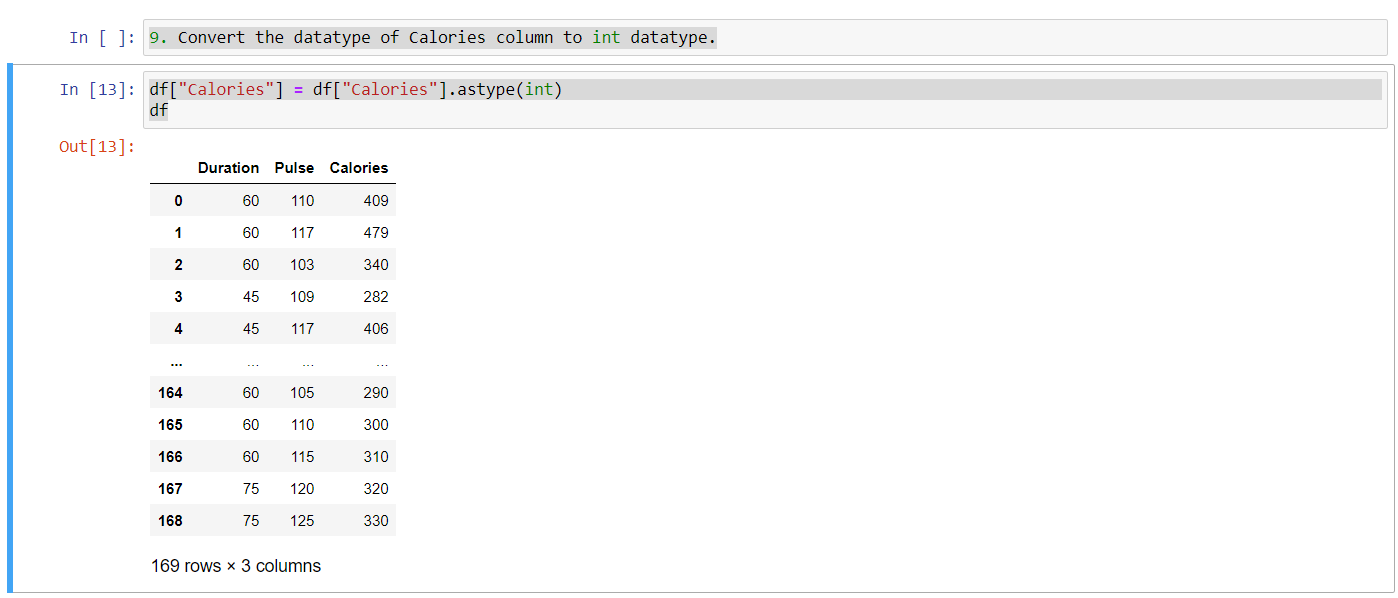
df



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

df["Calories"] = df["Calories"].astype(int)

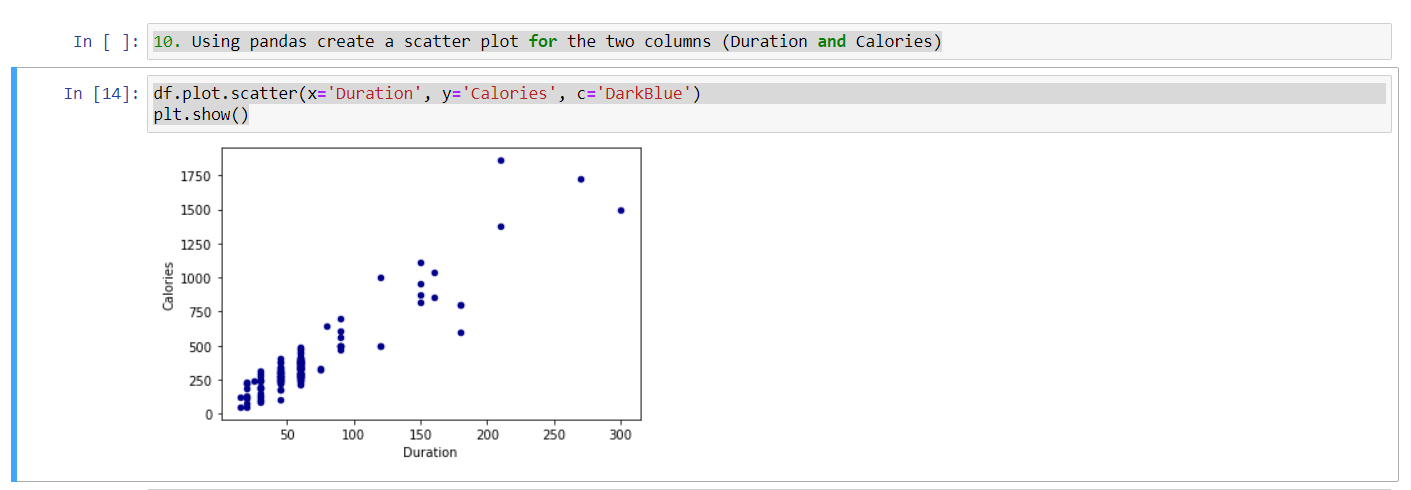
df



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

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

plt.show()



### 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\_state = 0)



1. Evaluate the model on testing part using score and classification\_report(y\_true, y\_pred)

# Naive Bayes

gnb = GaussianNB()

gnb.fit(X\_Train,Y\_Train)

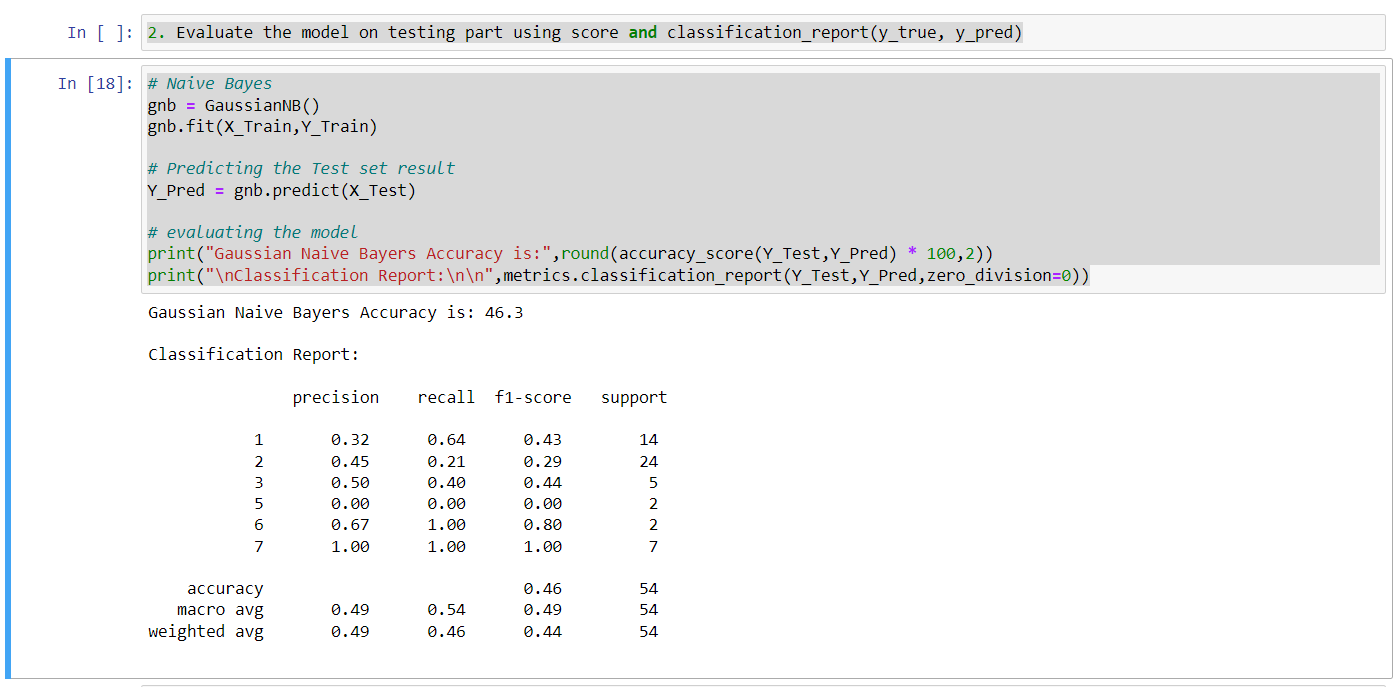
# Predicting the Test set result

Y\_Pred = gnb.predict(X\_Test)

# evaluating the model

print("Gaussian Naive Bayers Accuracy is:",round(accuracy\_score(Y\_Test,Y\_Pred) \* 100,2))

print("\nClassification Report:\n\n",metrics.classification\_report(Y\_Test,Y\_Pred,zero\_division=0))



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)

# 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,zero\_division=0))

