```
                                    **Lab Assignment-1**
```

Team Members:

Rahul Dhamerla: 8

Ramya Boyapati: 6

Youtube:

Introduction: This lab assignment focus on to make one familiar with python and machine learning algorithms. AI is a sort of man-made consciousness (AI) that furnishes PCs with the capacity to learn without being expressly modified. AI centers around the advancement of Computer Programs that can change when presented to new information.

Objective: To implement below concepts:

1. Lists, Tuples, Sets and Dictionaries
2. String Operations.
3. Inheritance
4. Multiple Regression
5. Naive Bayes, SVM and KNN implementation
6. K-Means Clustering.

Requirements:

1. PyCharm IDE
2. Python 3.7

Approaches/Methods:

1. Dynamic input method was used but not the static one’s.

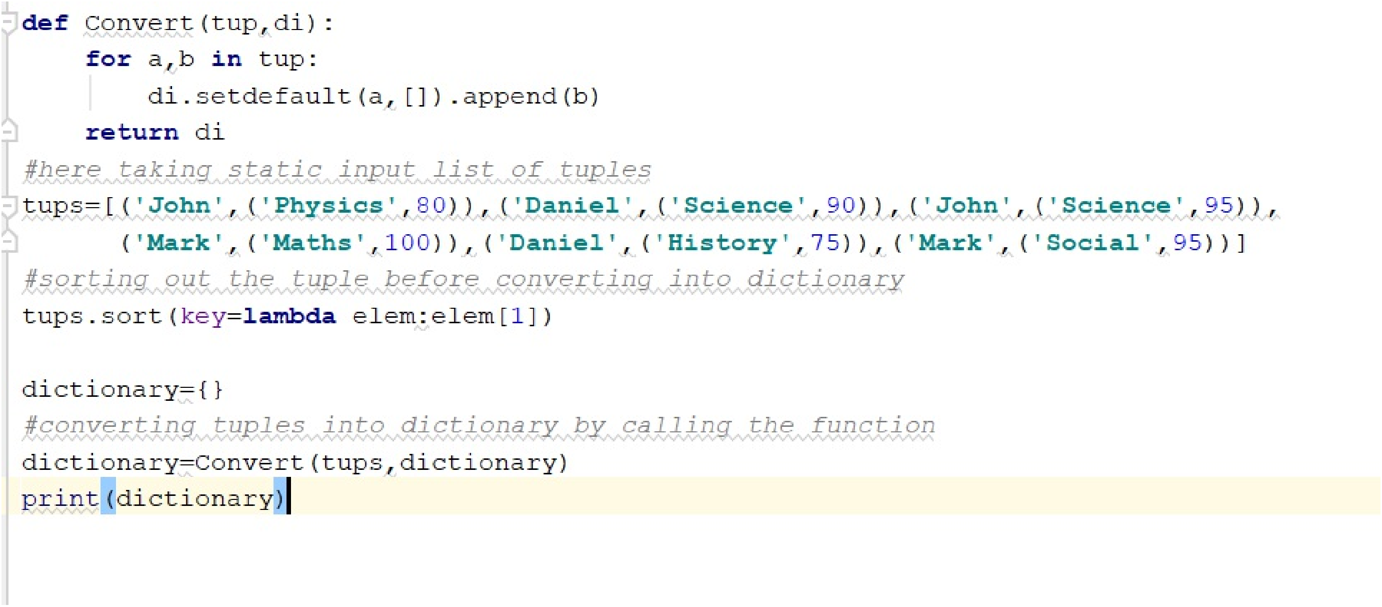
2. Used numpy and pandas dataframe for data cleaning.

3. Performed RMSE and R2 calculation for evaluating classification algorithms.

WorkFlow:

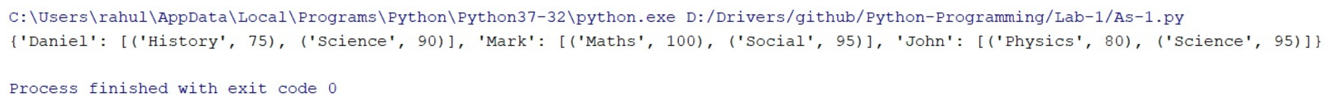
Task-1: Create a dictionary with keys as names and values as list of (subjects, marks)in sorted order with the given data.

Code:



```
def Convert(tup,di):
    for a,b in tup:
        di.setdefault(a,[]).append(b)
    return di
#here taking static input list of tuples
tups=[('John',('Physics',80)),('Daniel',('Science',90)),('John',('Science',95)),
      ('Mark',('Maths',100)),('Daniel',('History',75)),('Mark',('Social',95))]
#sorting out the tuple before converting into dictionary
tups.sort(key=lambda elem:elem[1])

dictionary={}
#converting tuples into dictionary by calling the function
dictionary=Convert(tups,dictionary)
print(dictionary)
```

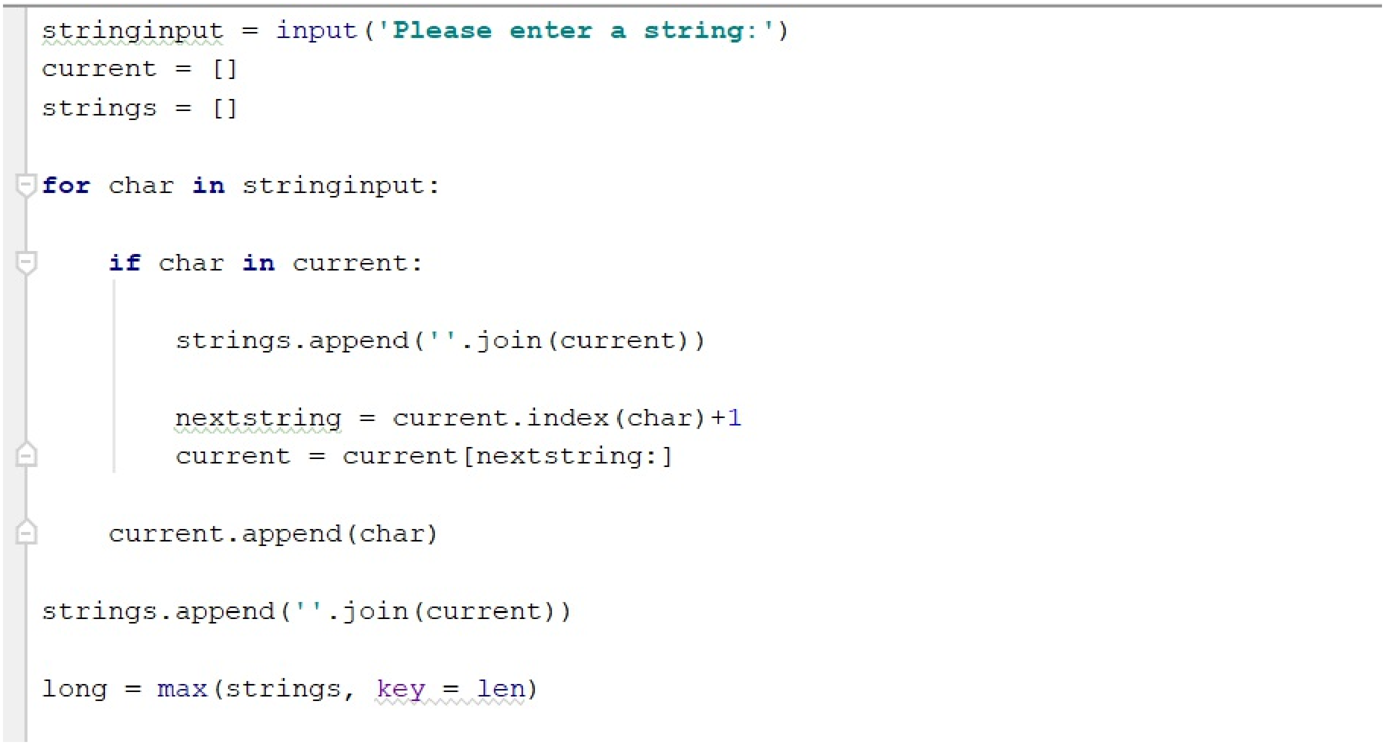
Output:



```
C:\Users\rahul\AppData\Local\Programs\Python\Python37-32\python.exe D:/Drivers/github/Python-Programming/Lab-1/As-1.py
{'Daniel': [('History', 75), ('Science', 90)], 'Mark': [('Maths', 100), ('Social', 95)], 'John': [('Physics', 80), ('Science', 95)]}

Process finished with exit code 0
```

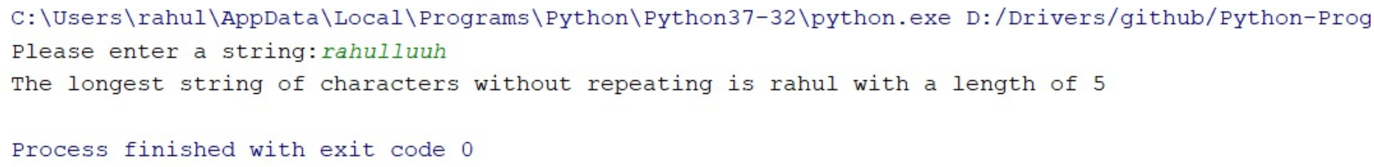
Task-2: Given a string, find the longest substrings without repeating characters along with the length as a tuple.

Code:



```python
stringinput = input('Please enter a string:')
current = []
strings = []

for char in stringinput:

    if char in current:

        strings.append(''.join(current))

        nextstring = current.index(char)+1
        current = current[nextstring:]

    current.append(char)

strings.append(''.join(current))

long = max(strings, key = len)
```

Output:



```
C:\Users\rahul\AppData\Local\Programs\Python\Python37-32\python.exe D:/Drivers/github/Python-Prog
Please enter a string:rahulluuh
The longest string of characters without repeating is rahul with a length of 5

Process finished with exit code 0
```

Task-3: Write a python program to create the following management systems.

1. Airline Booking Reservation System (e.g. classes Flight, Person, Employee, Passenger etc.)

Prerequisites:

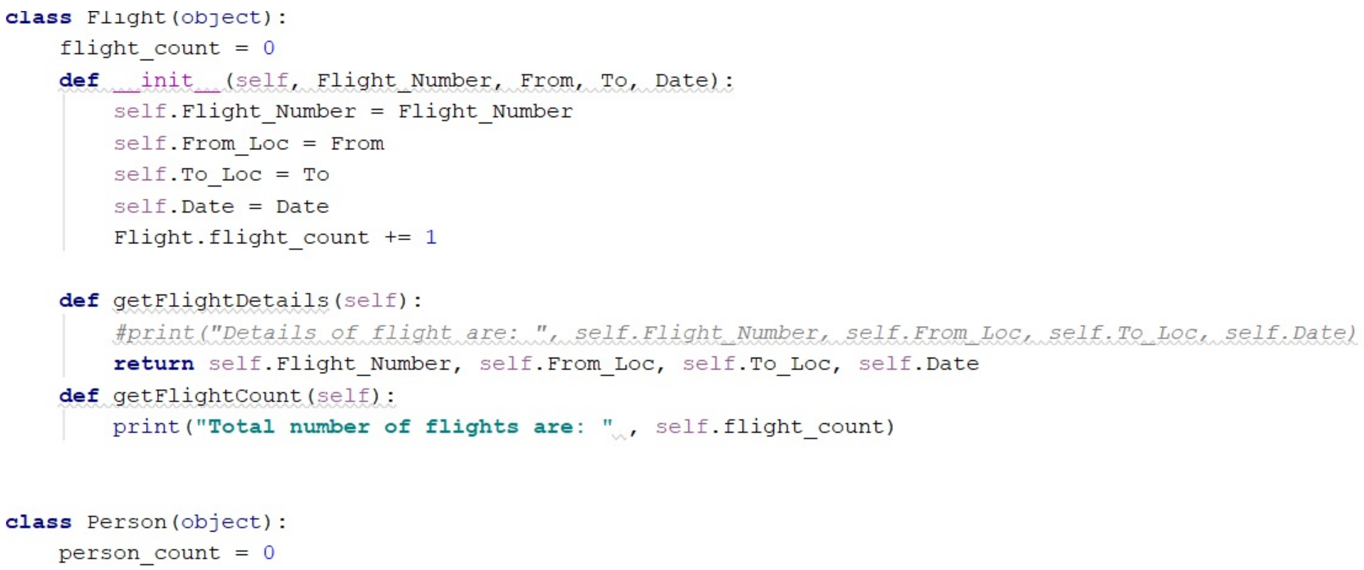
a. Your code should show inheritance at least once.

b. Your code should have one super call

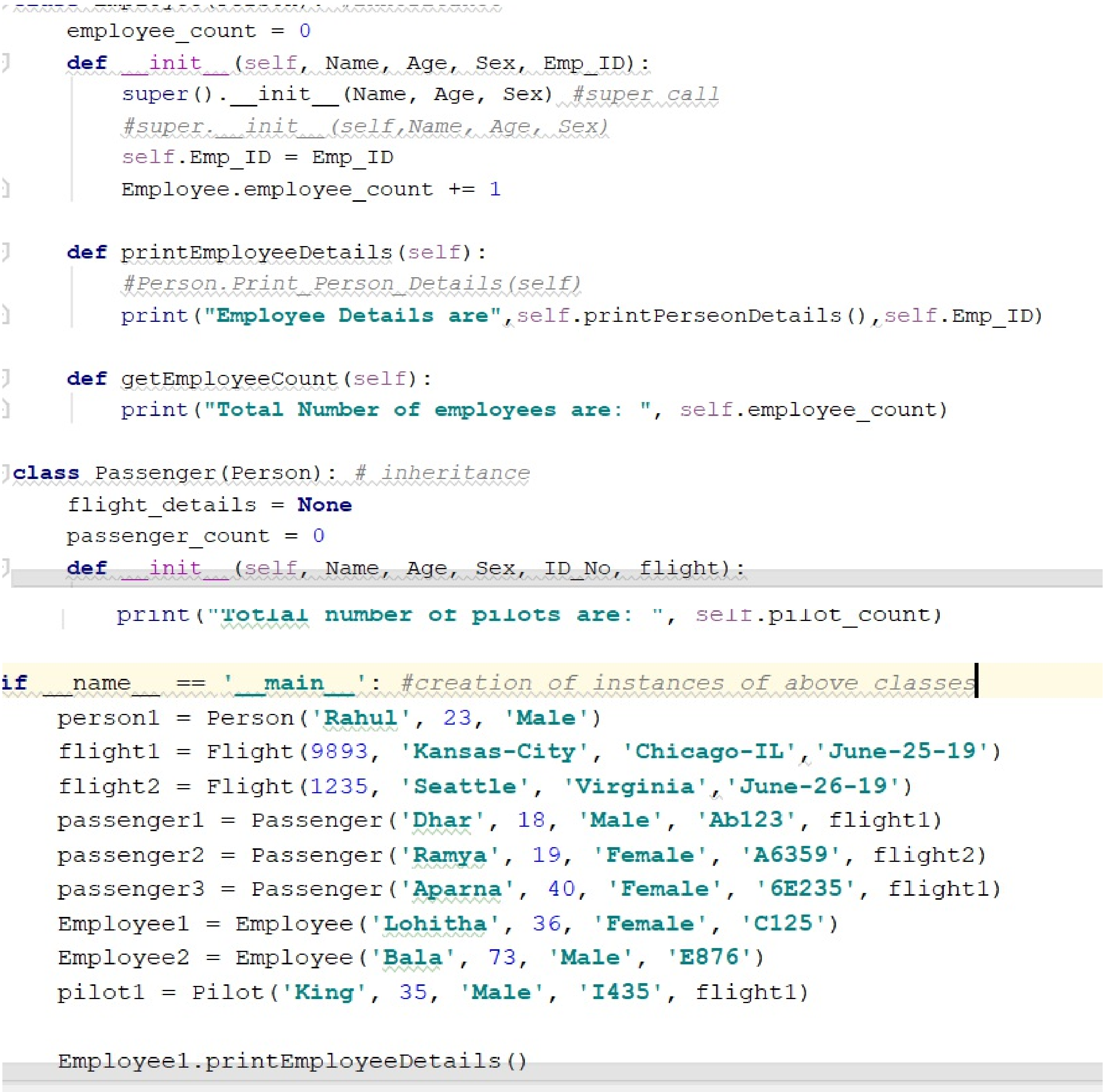
c. Use at least one private data member in your code.

d. Create instances of all classes and show the relationship between them

Code:

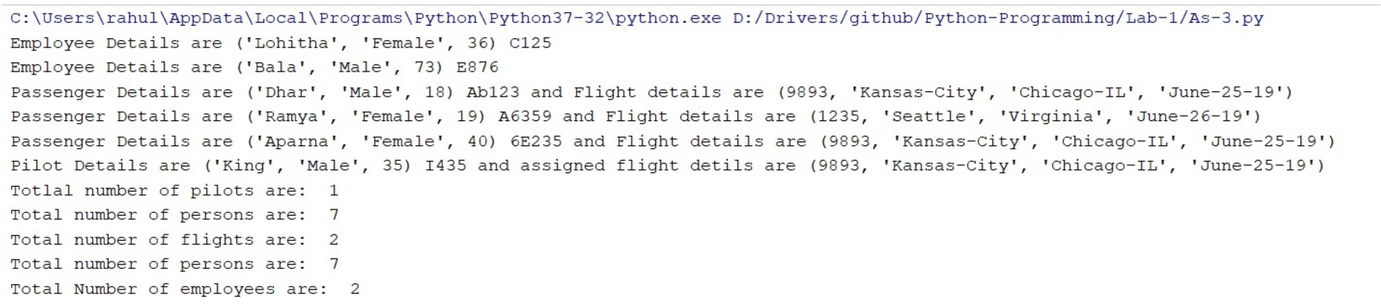


```
class Flight(object):
    flight_count = 0
    def __init__(self, Flight_Number, From, To, Date):
        self.Flight_Number = Flight_Number
        self.From_Loc = From
        self.To_Loc = To
        self.Date = Date
        Flight.flight_count += 1

    def getFlightDetails(self):
        #print("Details of flight are: ", self.Flight_Number, self.From_Loc, self.To_Loc, self.Date)
        return self.Flight_Number, self.From_Loc, self.To_Loc, self.Date
    def getFlightCount(self):
        print("Total number of flights are: ", self.flight_count)


class Person(object):
    person_count = 0
```



```
    employee_count = 0
    def __init__(self, Name, Age, Sex, Emp_ID):
        super().__init__(Name, Age, Sex) #super call
        #super.__init__(self,Name, Age, Sex)
        self.Emp_ID = Emp_ID
        Employee.employee_count += 1

    def printEmployeeDetails(self):
        #Person.Print_Person_Details(self)
        print("Employee Details are",self.printPerseonDetails(),self.Emp_ID)

    def getEmployeeCount(self):
        print("Total Number of employees are: ", self.employee_count)

class Passenger(Person): # inheritance
    flight_details = None
    passenger_count = 0
    def __init__(self, Name, Age, Sex, ID_No, flight):

        print("Totial number of pilots are: ", self.pilot_count)

if __name__ == '__main__': #creation of instances of above classes
    person1 = Person('Rahul', 23, 'Male')
    flight1 = Flight(9893, 'Kansas-City', 'Chicago-IL','June-25-19')
    flight2 = Flight(1235, 'Seattle', 'Virginia','June-26-19')
    passenger1 = Passenger('Dhar', 18, 'Male', 'Ab123', flight1)
    passenger2 = Passenger('Ramya', 19, 'Female', 'A6359', flight2)
    passenger3 = Passenger('Aparna', 40, 'Female', '6E235', flight1)
    Employee1 = Employee('Lohitha', 36, 'Female', 'C125')
    Employee2 = Employee('Bala', 73, 'Male', 'E876')
    pilot1 = Pilot('King', 35, 'Male', 'I435', flight1)

    Employee1.printEmployeeDetails()
```

Output:

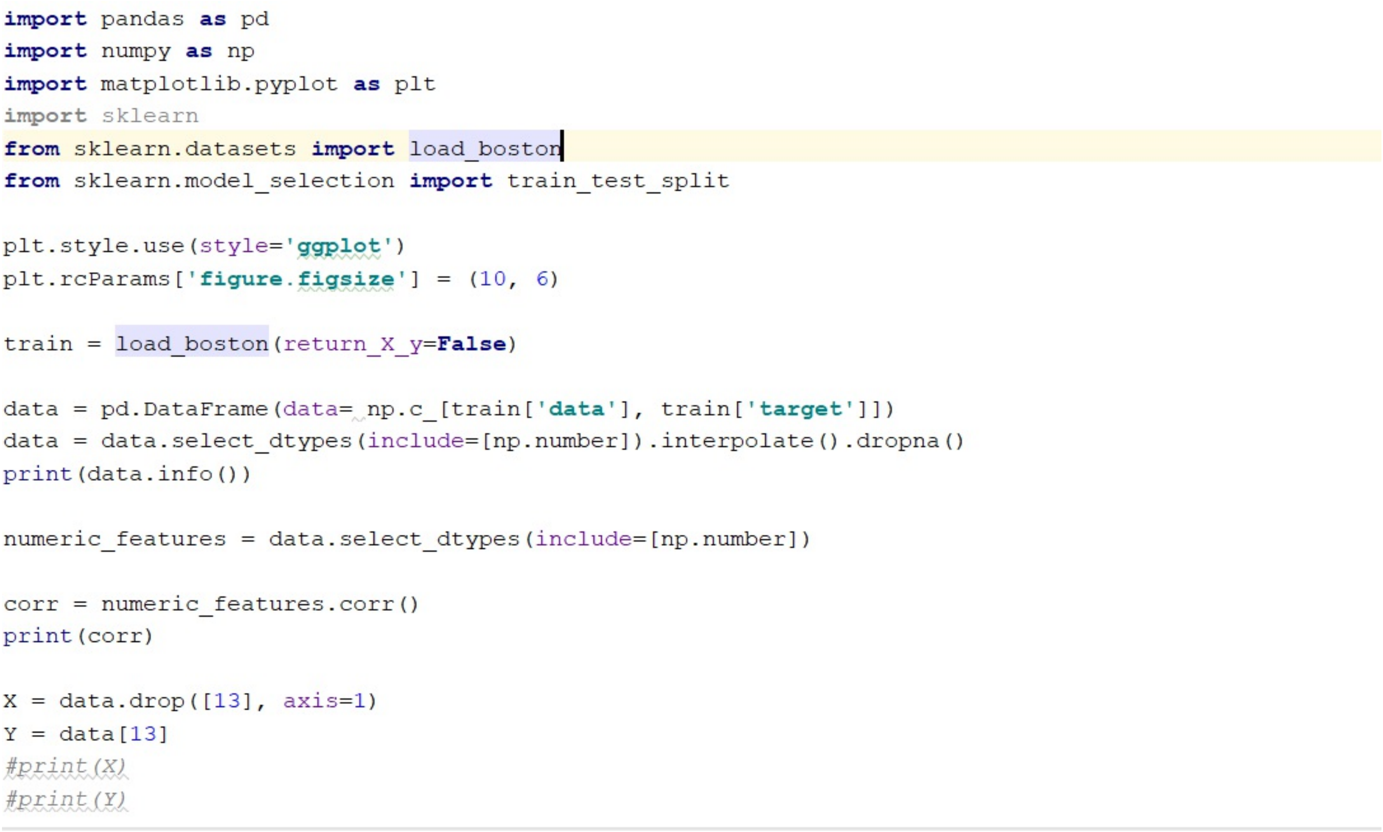


```
C:\Users\rahul\AppData\Local\Programs\Python\Python37-32\python.exe D:/Drivers/github/Python-Programming/Lab-1/As-3.py
Employee Details are ('Lohitha', 'Female', 36) C125
Employee Details are ('Bala', 'Male', 73) E876
Passenger Details are ('Dhar', 'Male', 18) Ab123 and Flight details are (9893, 'Kansas-City', 'Chicago-IL', 'June-25-19')
Passenger Details are ('Ramya', 'Female', 19) A6359 and Flight details are (1235, 'Seattle', 'Virginia', 'June-26-19')
Passenger Details are ('Aparna', 'Female', 40) 6E235 and Flight details are (9893, 'Kansas-City', 'Chicago-IL', 'June-25-19')
Pilot Details are ('King', 'Male', 35) I435 and assigned flight detils are (9893, 'Kansas-City', 'Chicago-IL', 'June-25-19')
Totial number of pilots are:  1
Total number of persons are:  7
Total number of flights are:  2
Total number of persons are:  7
Total Number of employees are:  2
```

Task-4: Create Multiple Regression by choosing a dataset of your choice (again beforeevaluating, clean the data set with the EDA learned in the class). Evaluate the model using RMSE and R2 and also report if you saw any improvement before and after the EDA.

Observation: After applying EDA on the correlated values, we observed that there is slight increase in the R-squared value and the resulted in the decrease of Root Mean Square Error value.

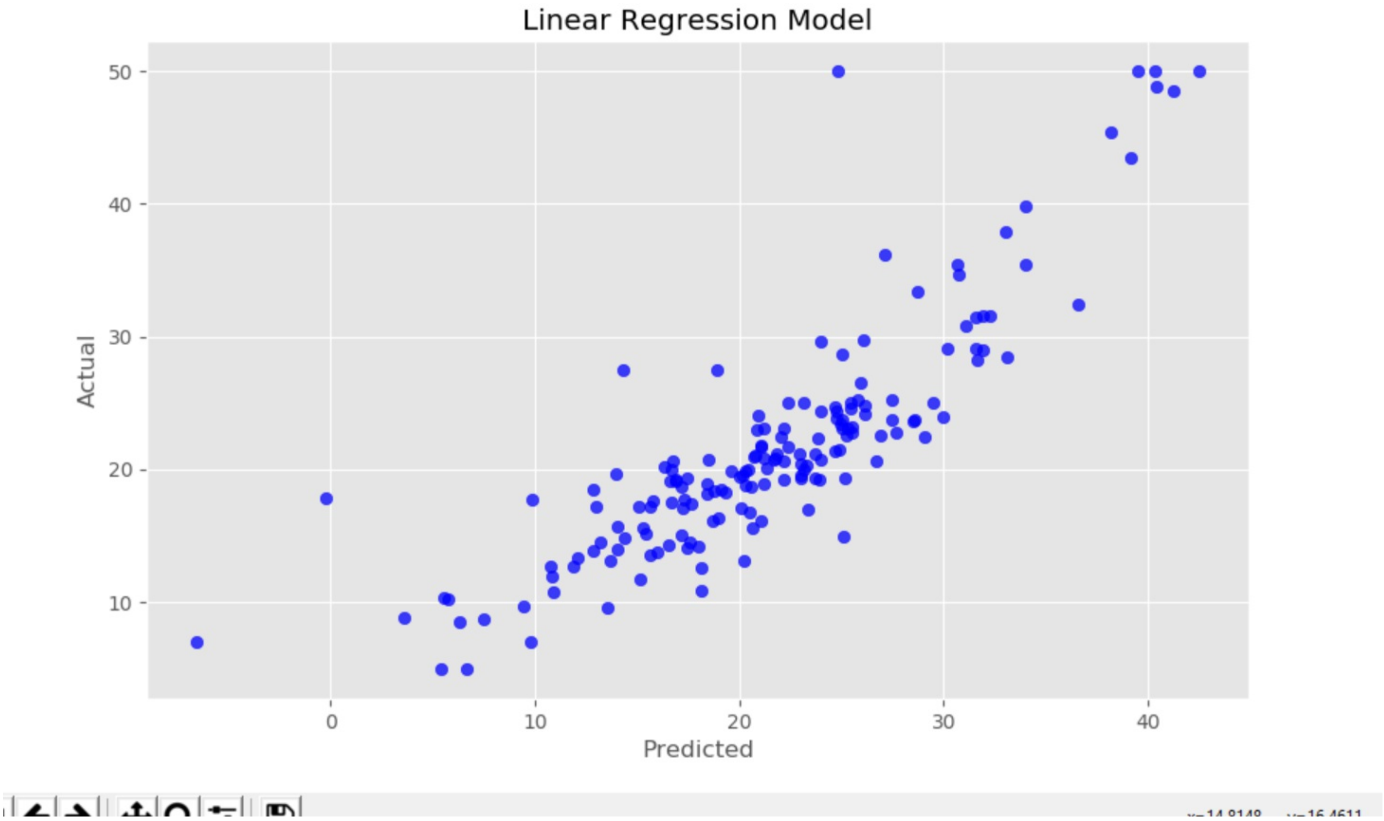
Code:



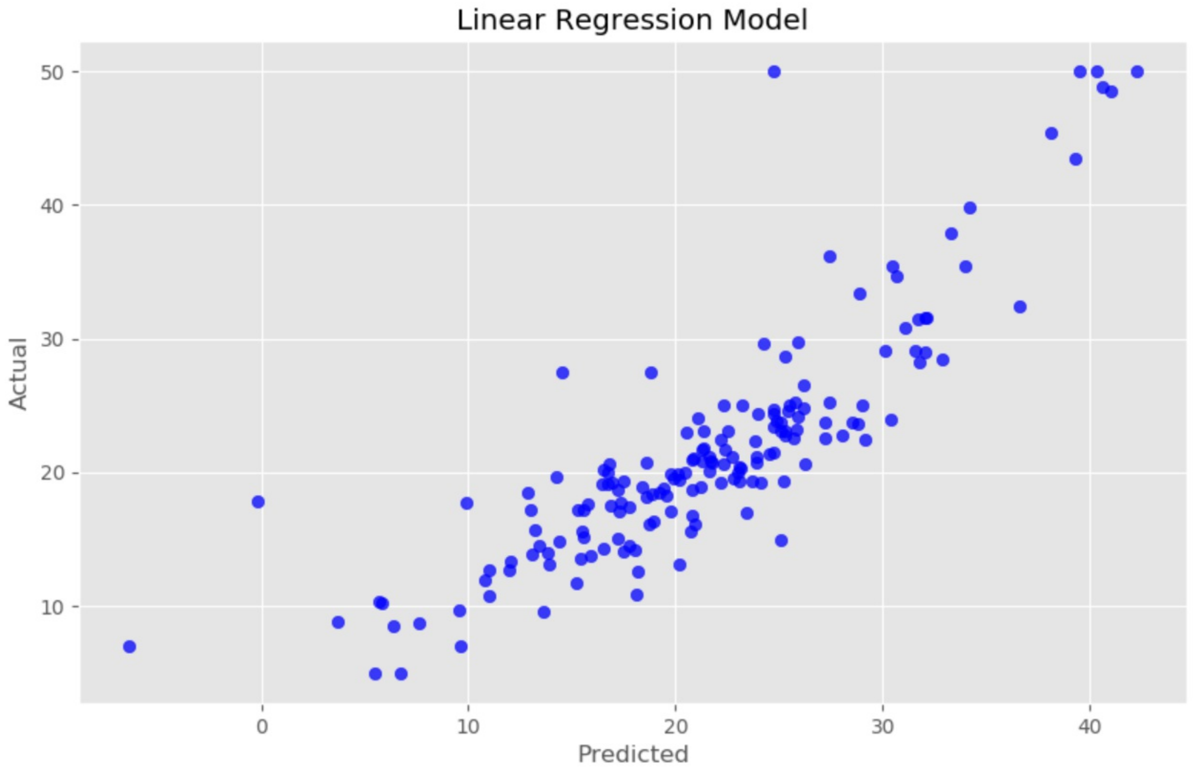
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split

plt.style.use(style='ggplot')
plt.rcParams['figure.figsize'] = (10, 6)

train = load_boston(return_X_y=False)

data = pd.DataFrame(data= np.c_[train['data'], train['target']])
data = data.select_dtypes(include=[np.number]).interpolate().dropna()
print(data.info())

numeric_features = data.select_dtypes(include=[np.number])

corr = numeric_features.corr()
print(corr)

X = data.drop([13], axis=1)
Y = data[13]
#print(X)
#print(Y)
```

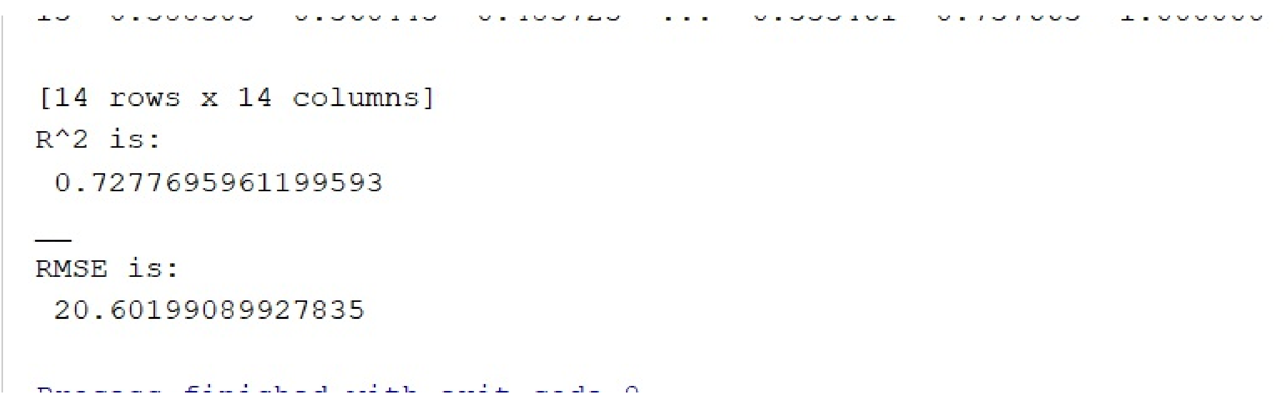
```
X_train, X_test,y_train, y_test = train_test_split(
                                    X, Y, random_state=42, test_size=.33)

from sklearn import linear_model
lr = linear_model.LinearRegression()
model = lr.fit(X_train, y_train)
##Evaluate the performance and visualize results
print ("R^2 is: \n", model.score(X_test, y_test))
print("__")
predictions = model.predict(X_test)
from sklearn.metrics import mean_squared_error
print ('RMSE is: \n', mean_squared_error(y_test, predictions))

##visualize

actual_values = y_test
plt.scatter(predictions, actual_values, alpha=.75,
            color='b') #alpha helps to show overlapping data
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Linear Regression Model')
plt.show()
```

Output:



```
None
          0         1         2   ...        11        12        13
0   1.000000 -0.200469  0.406583  ... -0.385064  0.455621 -0.388305
1  -0.200469  1.000000 -0.533828  ...  0.175520 -0.412995  0.360445
2   0.406583 -0.533828  1.000000  ... -0.356977  0.603800 -0.483725
3  -0.055892 -0.042697  0.062938  ...  0.048788 -0.053929  0.175260
4   0.420972 -0.516604  0.763651  ... -0.380051  0.590879 -0.427321
5  -0.219247  0.311991 -0.391676  ...  0.128069 -0.613808  0.695360
6   0.352734 -0.569537  0.644779  ... -0.273534  0.602339 -0.376955
7  -0.379670  0.664408 -0.708027  ...  0.291512 -0.496996  0.249929
8   0.625505 -0.311948  0.595129  ... -0.444413  0.488676 -0.381626
9   0.582764 -0.314563  0.720760  ... -0.441808  0.543993 -0.468536
10  0.289946 -0.391679  0.383248  ... -0.177383  0.374044 -0.507787
11 -0.385064  0.175520 -0.356977  ...  1.000000 -0.366087  0.333461
12  0.455621 -0.412995  0.603800  ... -0.366087  1.000000 -0.737663
13 -0.388305  0.360445 -0.483725  ...  0.333461 -0.737663  1.000000

[14 rows x 14 columns]
R^2 is:
 0.7261570836552479
__
RMSE is:
 20.724023437339735
```





```
[14 rows x 14 columns]
R^2 is:
 0.7277695961199593
__
RMSE is:
 20.60199089927835

Process finished with exit code 0
```

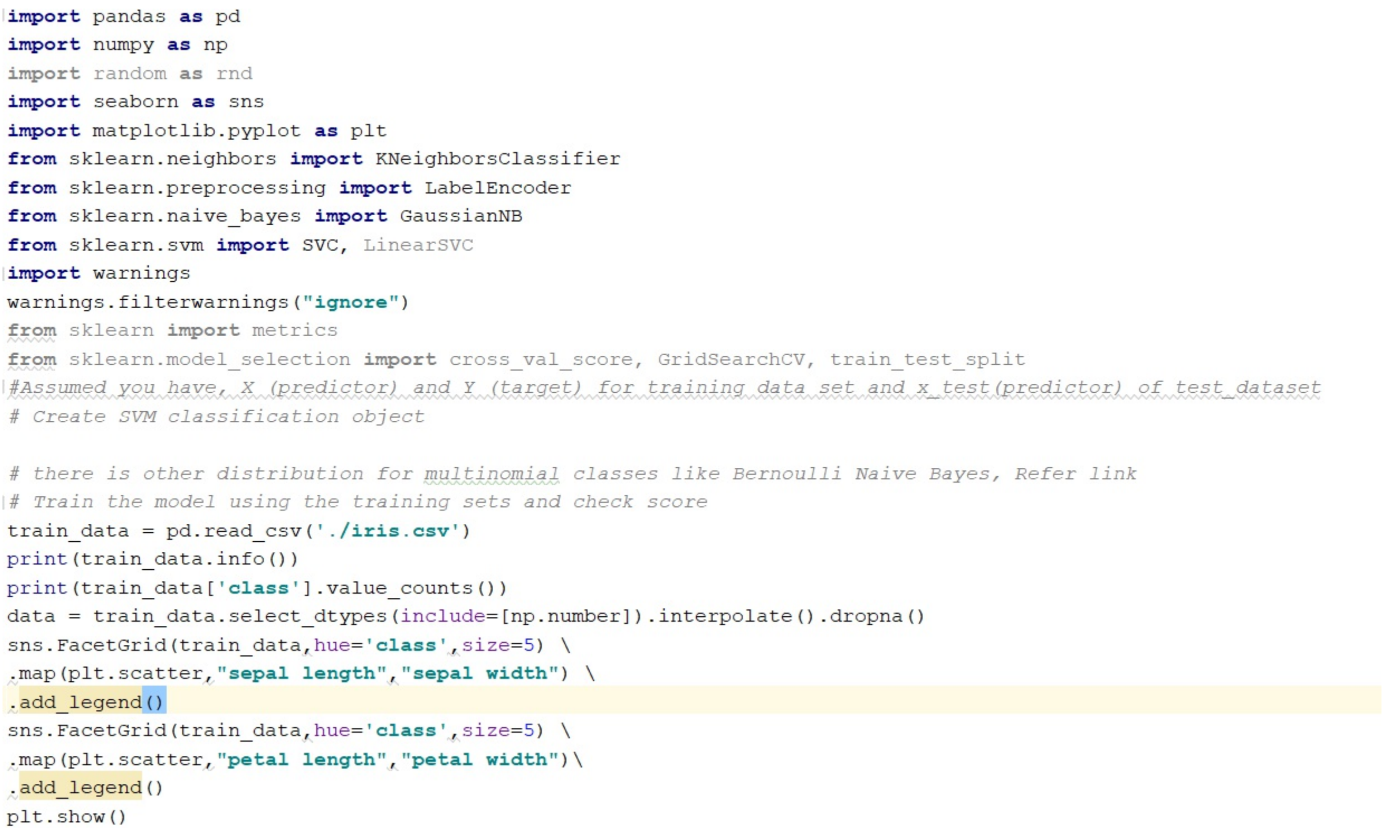
Task-5: Pick any dataset from the dataset sheet in the class sheet or online which includes both numeric and non-numeric features.

a. Perform exploratory data analysis on the data set (like Handling null values, removing the features not correlated to the target class, encoding the categorical features, …)

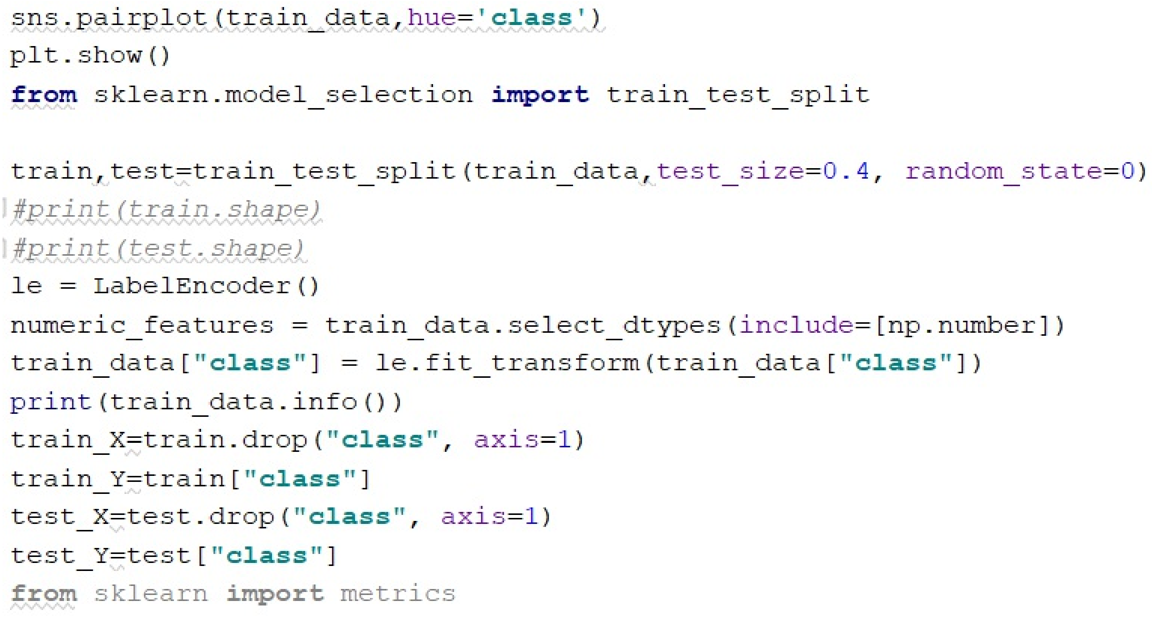
b. Apply the three classification algorithms Naïve Baye’s, SVM and KNN on the chosen data set and report which classifier gives better result.

Observation: For the choosen dataset, SVM classifier gave a better accuracy rate.

Code:



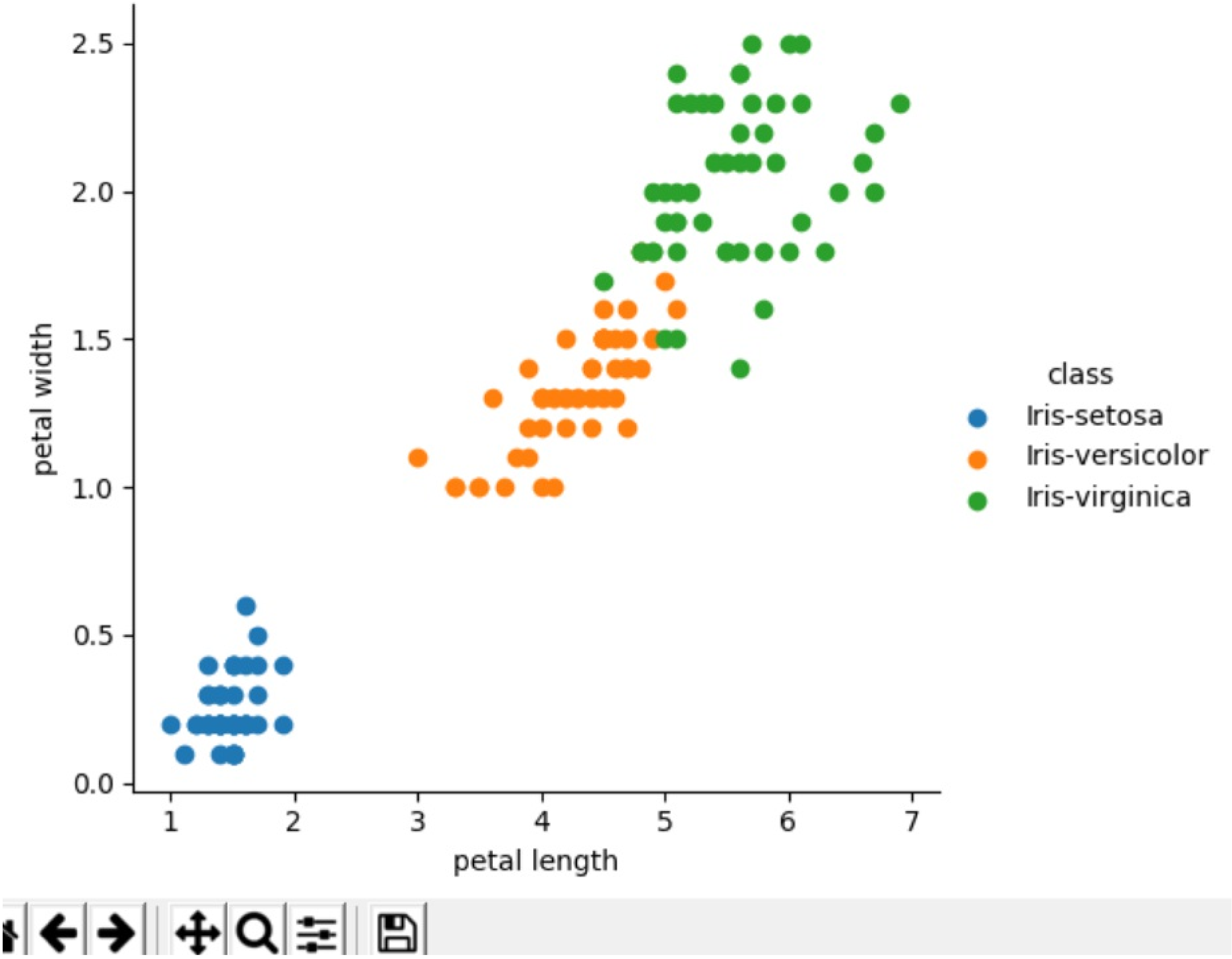
```
import pandas as pd
import numpy as np
import random as rnd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC, LinearSVC
import warnings
warnings.filterwarnings("ignore")
from sklearn import metrics
from sklearn.model_selection import cross_val_score, GridSearchCV, train_test_split
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create SVM classification object

# there is other distribution for multinomial classes like Bernoulli Naive Bayes, Refer link
# Train the model using the training sets and check score
train_data = pd.read_csv('./iris.csv')
print(train_data.info())
print(train_data['class'].value_counts())
data = train_data.select_dtypes(include=[np.number]).interpolate().dropna()
sns.FacetGrid(train_data,hue='class',size=5) \
.map(plt.scatter,"sepal length","sepal width") \
.add_legend()
sns.FacetGrid(train_data,hue='class',size=5) \
.map(plt.scatter,"petal length","petal width")\
.add_legend()
plt.show()
```



```
sns.pairplot(train_data,hue='class')
plt.show()
from sklearn.model_selection import train_test_split

train,test=train_test_split(train_data,test_size=0.4, random_state=0)
#print(train.shape)
#print(test.shape)
le = LabelEncoder()
numeric_features = train_data.select_dtypes(include=[np.number])
train_data["class"] = le.fit_transform(train_data["class"])
print(train_data.info())
train_X=train.drop("class", axis=1)
train_Y=train["class"]
test_X=test.drop("class", axis=1)
test_Y=test["class"]
from sklearn import metrics
```

```
#naive bayes model
model = GaussianNB()
model.fit(train_X, train_Y)
print(" naive Accuracy is : ", model.score(test_X,test_Y))

#knn
model = KNeighborsClassifier(n_neighbors=3)
model.fit(train_X, train_Y)
pred_Y = model.predict(test_X)
print(" knn Accuracy is : ",model.score(test_X,test_Y))


#svm
model = SVC()
model.fit(train_X, train_Y)
print(" svm Accuracy is : ",model.score(test_X,test_Y))
```

Output:


2.5
2.0
1.5
1.0
0.5
0.0
petal width
1
2
3
4
5
6
7
petal length
class
Iris-setosa
Iris-versicolor
Iris-virginica

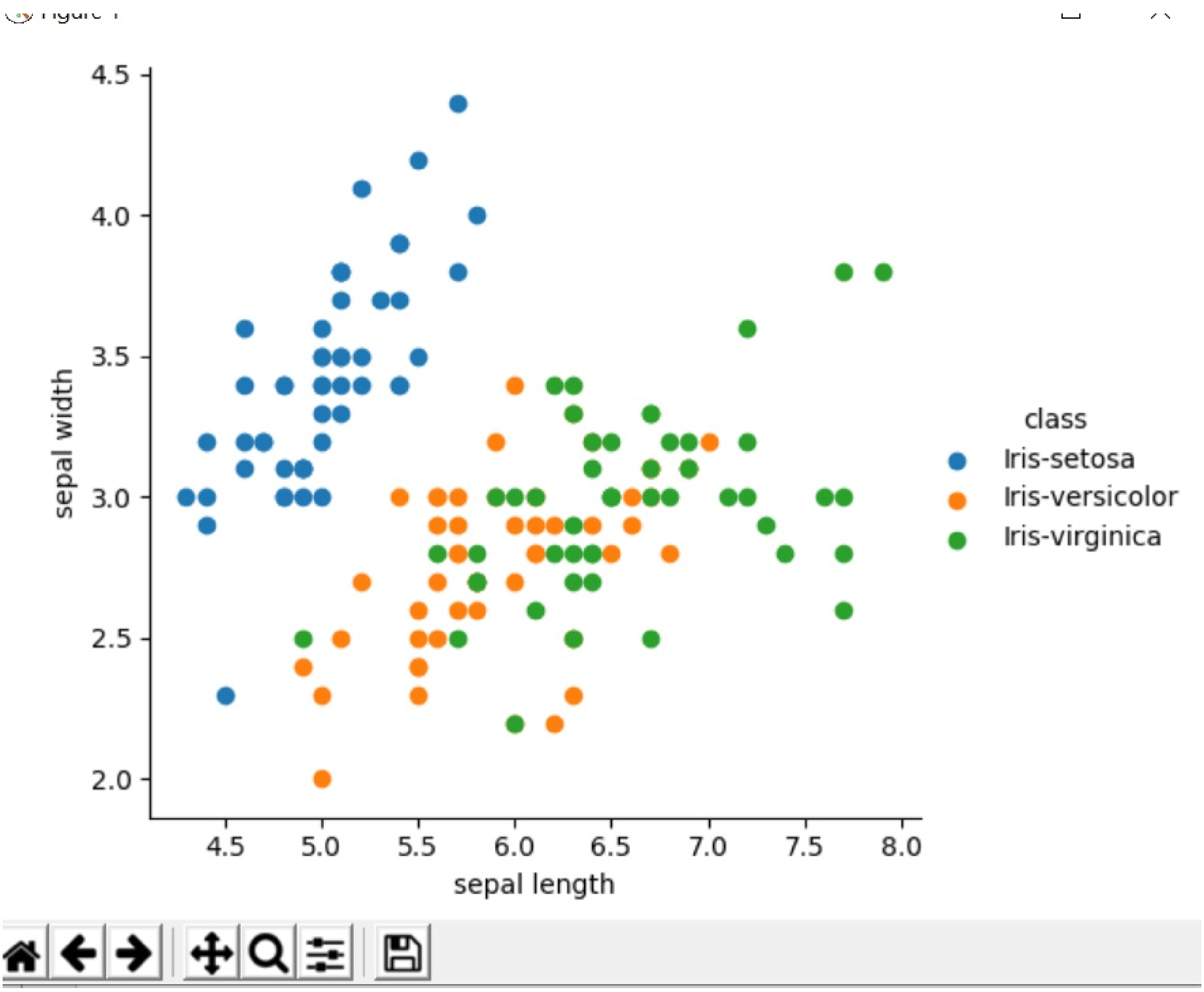


Figure 1
4.5
4.0
3.5
3.0
2.5
2.0
sepal width
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal length
class
Iris-setosa
Iris-versicolor
Iris-virginica

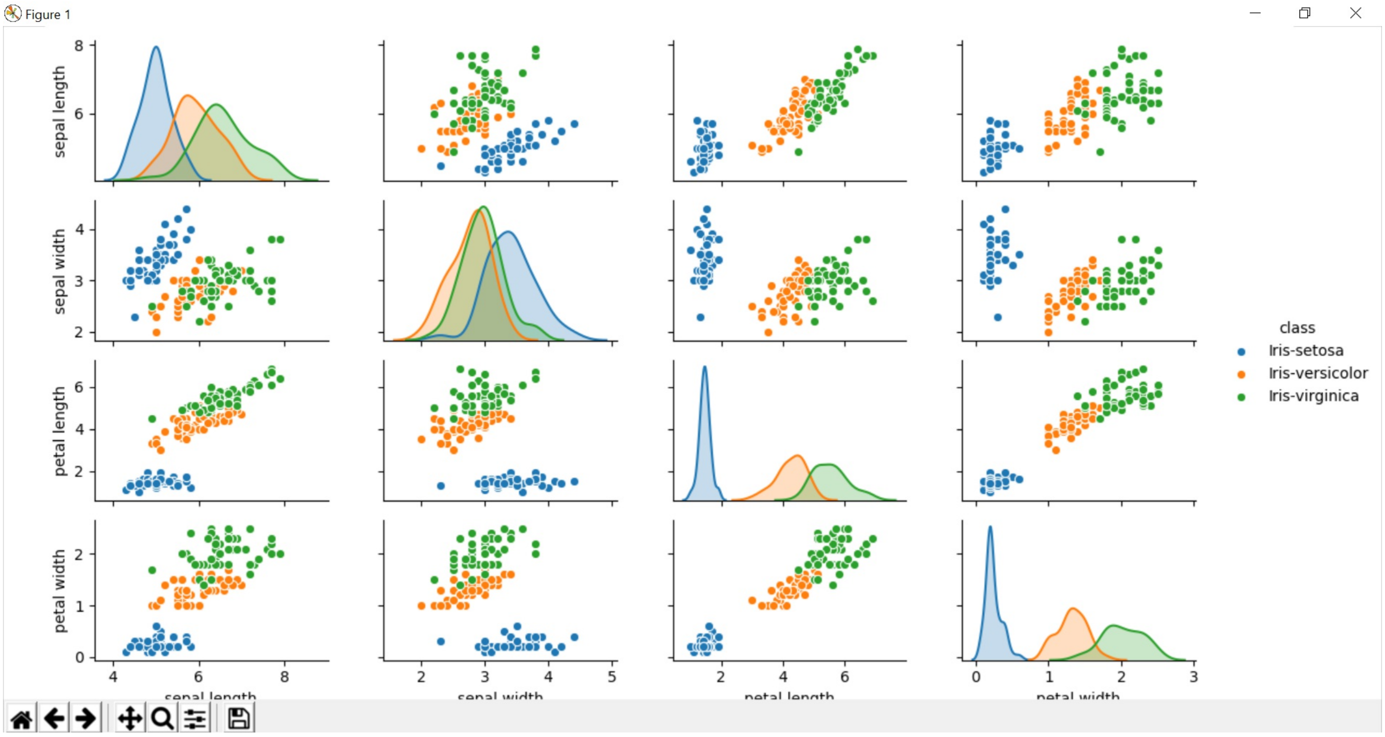
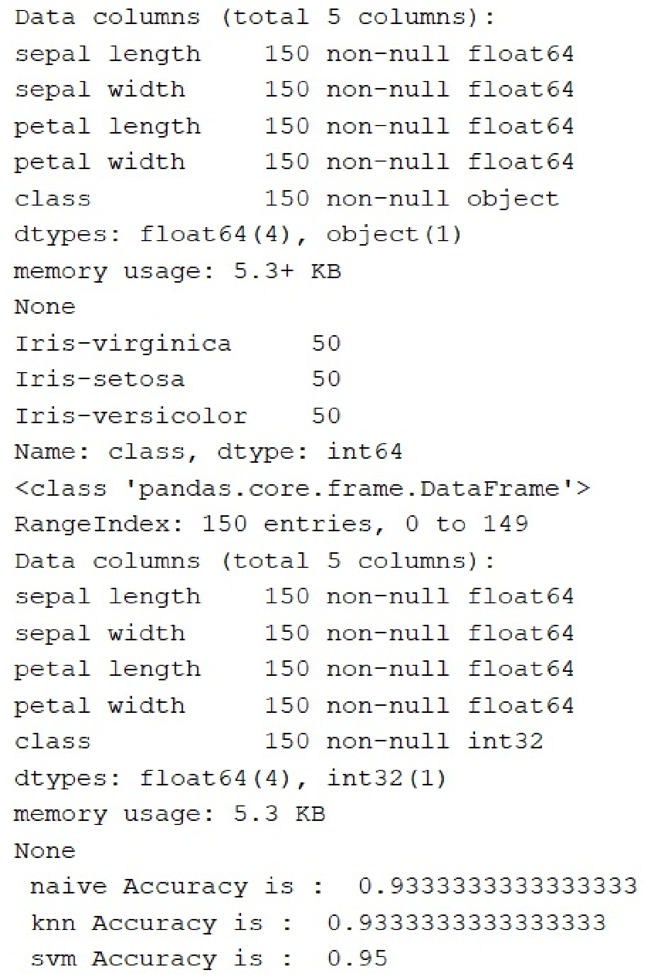


Figure 1
sepal length
sepal width
petal length
petal width
8
6
4
3
2
6
4
2
2
1
0
4
6
8
2
3
4
5
2
4
6
0
1
2
3
sepal length
sepal width
petal length
petal width
class
Iris-setosa
Iris-versicolor
Iris-virginica



```
Data columns (total 5 columns):
sepal length    150 non-null float64
sepal width     150 non-null float64
petal length    150 non-null float64
petal width     150 non-null float64
class           150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.3+ KB
None
Iris-virginica     50
Iris-setosa        50
Iris-versicolor    50
Name: class, dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal length    150 non-null float64
sepal width     150 non-null float64
petal length    150 non-null float64
petal width     150 non-null float64
class           150 non-null int32
dtypes: float64(4), int32(1)
memory usage: 5.3 KB
None
 naive Accuracy is :  0.9333333333333333
 knn Accuracy is :  0.9333333333333333
 svm Accuracy is :  0.95
```

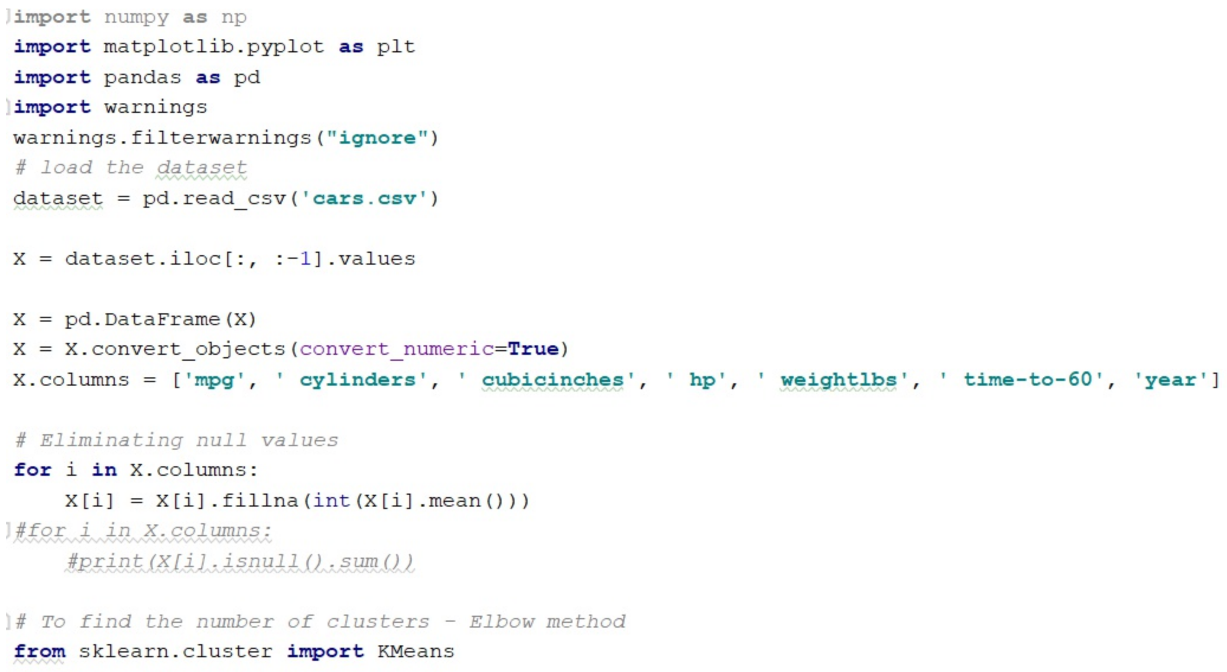
Task-6: Choose any dataset of your choice. Apply K-means on the dataset and visualize the clusters using matplotlib or seaborn.

a. Report which K is the best using the elbow method.

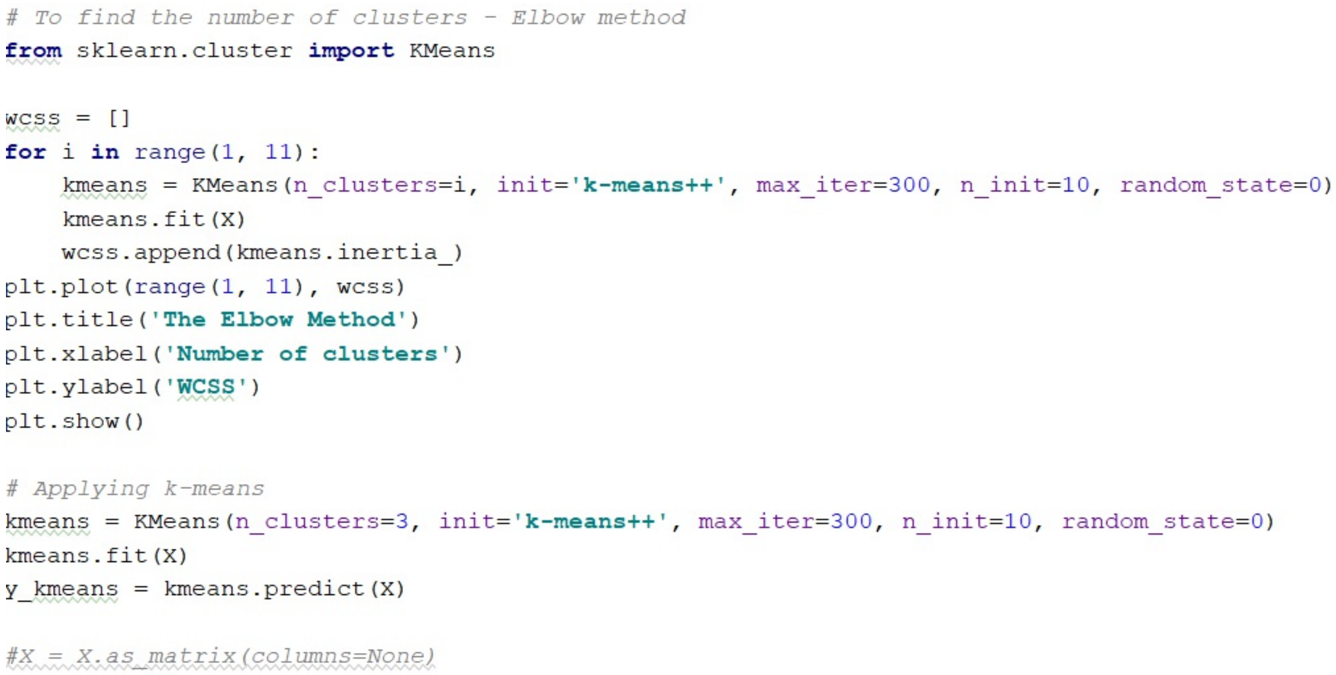
b. Evaluate with silhouette score or other scores relevant for unsupervised approaches (before applying clustering clean the data set with the EDA learned in the class)

Observation: After applying the elbow method , we observed the number of clusters as 3 and applied K-means clustering accordingly and then plotted it using the “mpg” and “year” columns.

Code:

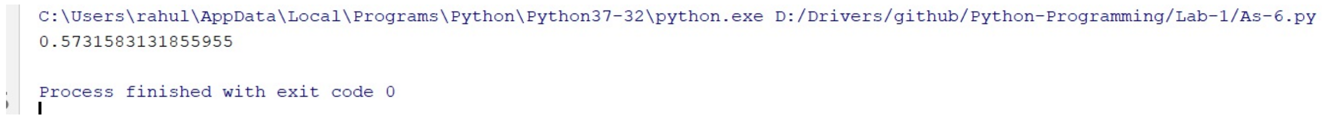


```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
# load the dataset
dataset = pd.read_csv('cars.csv')

X = dataset.iloc[:, :-1].values

X = pd.DataFrame(X)
X = X.convert_objects(convert_numeric=True)
X.columns = ['mpg', ' cylinders', ' cubicinches', ' hp', ' weightlbs', ' time-to-60', 'year']

# Eliminating null values
for i in X.columns:
    X[i] = X[i].fillna(int(X[i].mean()))
#for i in X.columns:
    #print(X[i].isnull().sum())

# To find the number of clusters - Elbow method
from sklearn.cluster import KMeans
```

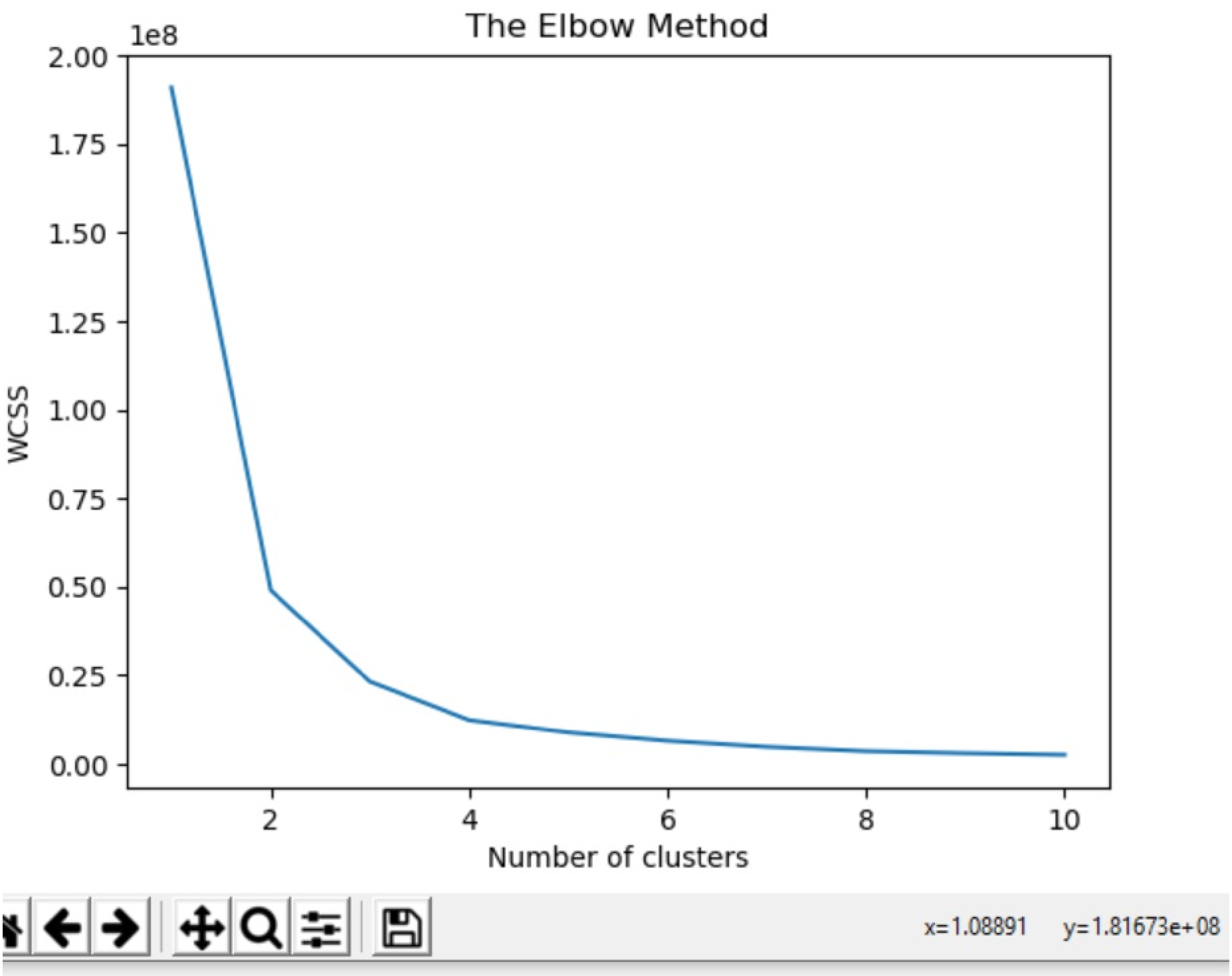


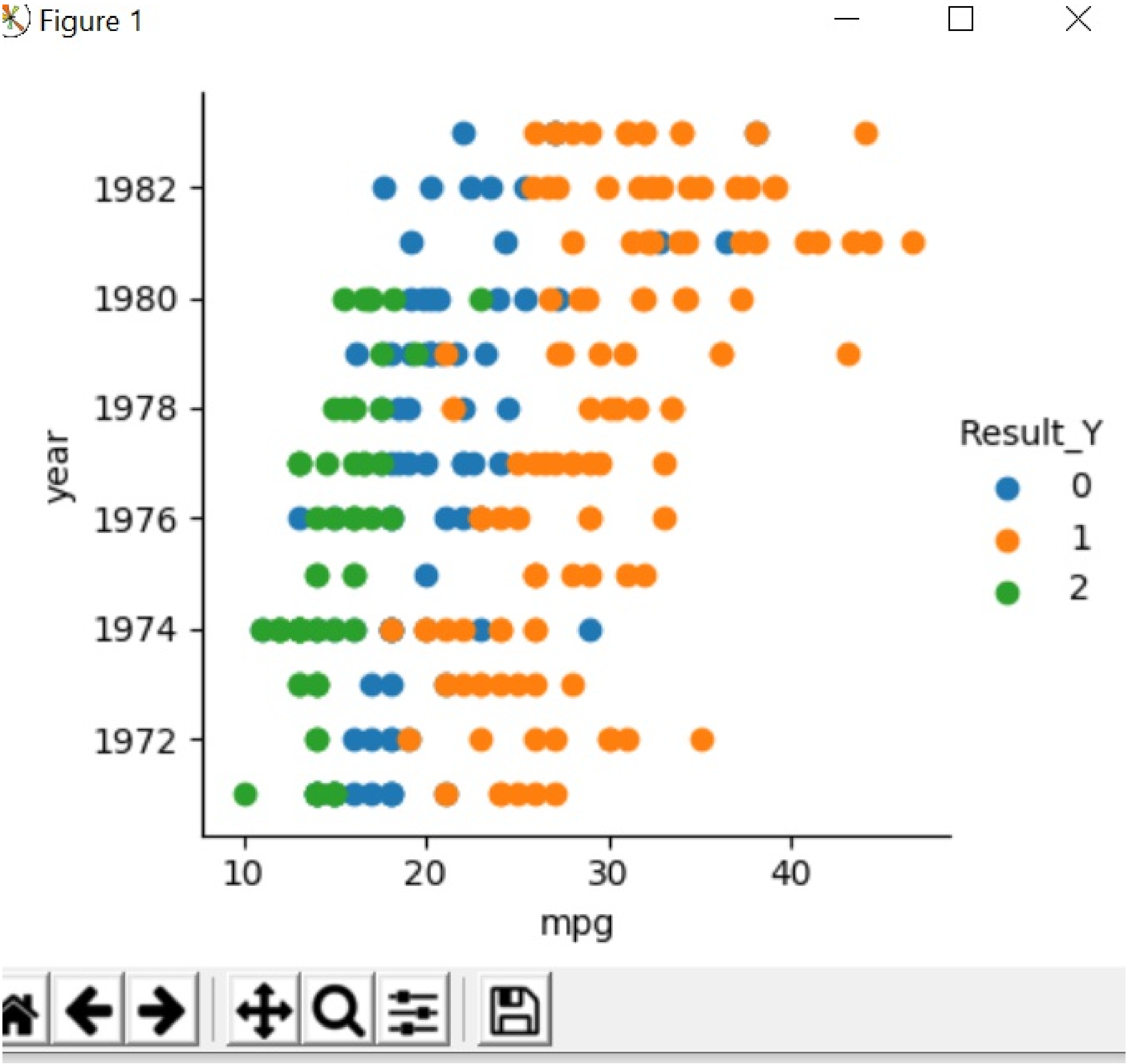
```
# To find the number of clusters - Elbow method
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Applying k-means
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

#X = X.as_matrix(columns=None)
```

Output:



```
C:\Users\rahul\AppData\Local\Programs\Python\Python37-32\python.exe D:/Drivers/github/Python-Programming/Lab-1/As-6.py
0.5731583131855955

Process finished with exit code 0
```


The Elbow Method
1e8
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
WCSS
2
4
6
8
10
Number of clusters
x=1.08891
y=1.81673e+08



Contributions:

- Each team member contributed equally in the documentation and helped in search of datasets.

Rahul Dhamerla,8:

- Implemented logic for task 1 and task 3.

- Implemented K-means Clustering(task-6)

Ramya Boyapati,6:

- Implemented logic for task-2.

- Created Multiple Regression model(task-4) and implemented the trained classifier models like Naive baye’s, SVM and KNN.

Conclusion:

We have understood and implemented the above mentioned concepts and created multiple regression and evaluated R2 and RMSE scores and classified trained models like Naive Baye’s,SVM and KNN and applied kmeans clustering and plotted them.