gowtham304 / **python**

Editing assignment 1

Page History New Page Delete Page

assignment 1

Write Preview

h1 h2 h3 B *i*

Edit mode: Markdown

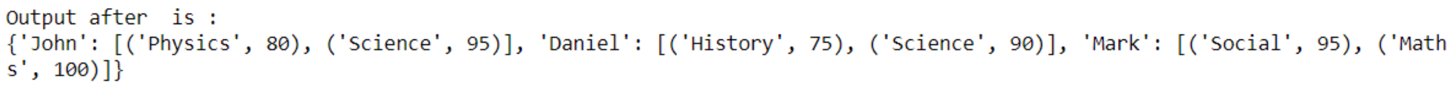
Assignment 1

TASK 1 -To create a dictionary with keys as names and values as list of (subjects, marks)in sorted order.

We have created two methods the first method creates a dictionary and the second method gives the desired output by sorting the values base on marks.

```
def tup_to_dict(tup, dict):
    for a, b in tup:
        dict.setdefault(a, []).append(b)
    return dict

def sort_dict(dict):
    for idx,list_of_tups in dict.items():
        dict[idx] = sorted(list_of_tups,key=lambda x: x[1]) # sorts on 1st value of list
    return dict

tup = [( 'John', ('Physics', 80)) , ('Daniel', ('Science', 90)), ('John', ('Science', 95)),
       ('Mark',('Maths', 100)), ('Daniel', ('History', 75)), ('Mark', ('Social', 95))]

dict = {}
dict = tup_to_dict(tup,dict)

print("Output after  is : ")
print(sort_dict(dict))
```

Output is sorted by the marks.



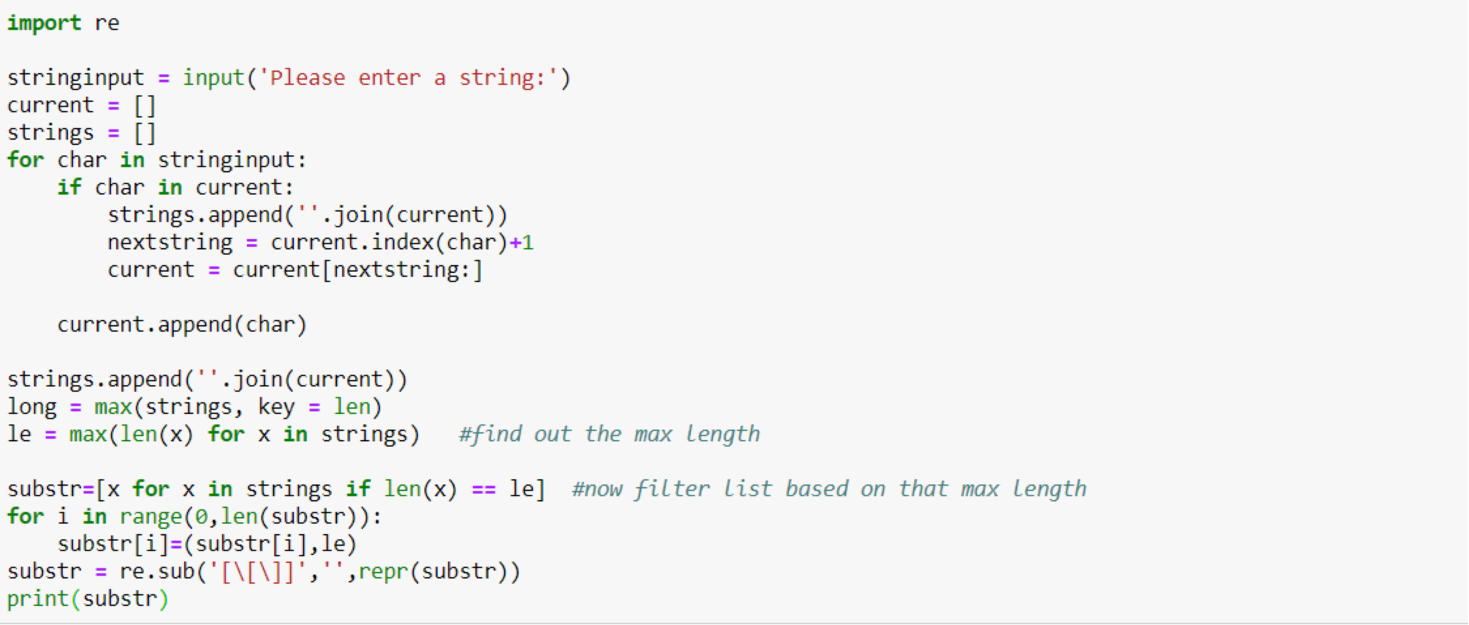
```
Output after  is :
{'John': [('Physics', 80), ('Science', 95)], 'Daniel': [('History', 75), ('Science', 90)], 'Mark': [('Social', 95), ('Math
s', 100)]}
```

This task gives insights into using dictionary,tuples and formatting them.

TASK 2- Given a string "pwwkew", find the longest sub strings without repeating characters along with the length as a tuple.

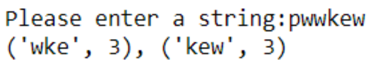
In this task we have created two different lists one for the sake of comparision.The second array is used to store all the non repeating substring.Then we have created an new array subset which just stores the values of largest length.

It is then formatted to give the output in desired way.



```
import re

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)
le = max(len(x) for x in strings)   #find out the max length

substr=[x for x in strings if len(x) == le]  #now filter list based on that max length
for i in range(0,len(substr)):
    substr[i]=(substr[i],le)
substr = re.sub('[\[\]]','',repr(substr))
print(substr)
```

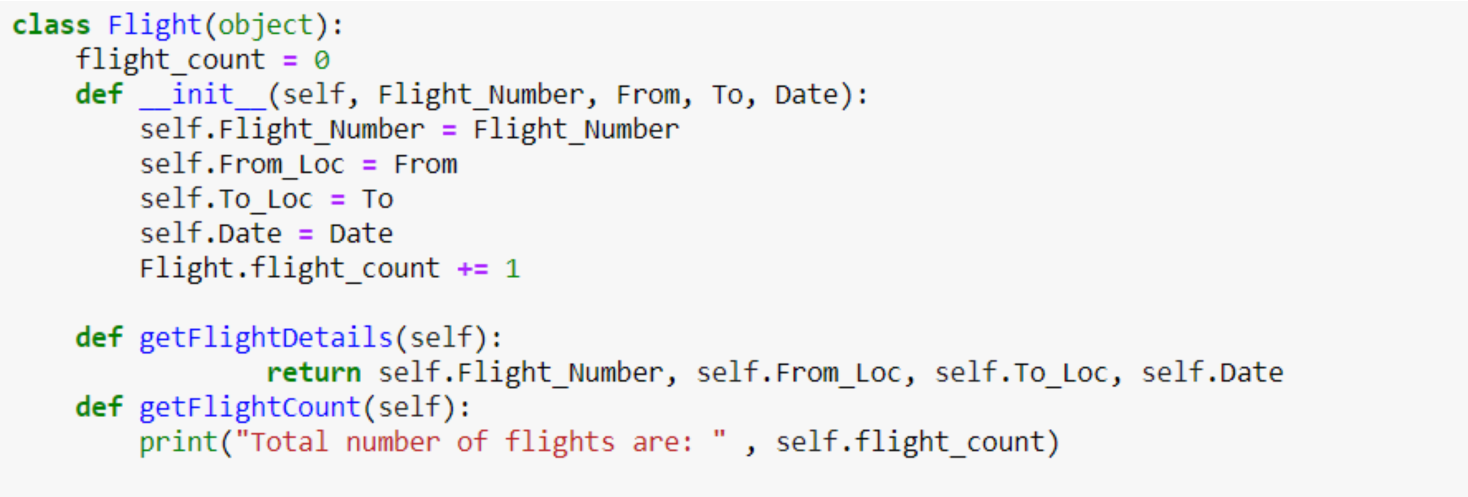
output:



```
Please enter a string:pwwkew
('wke', 3), ('kew', 3)
```

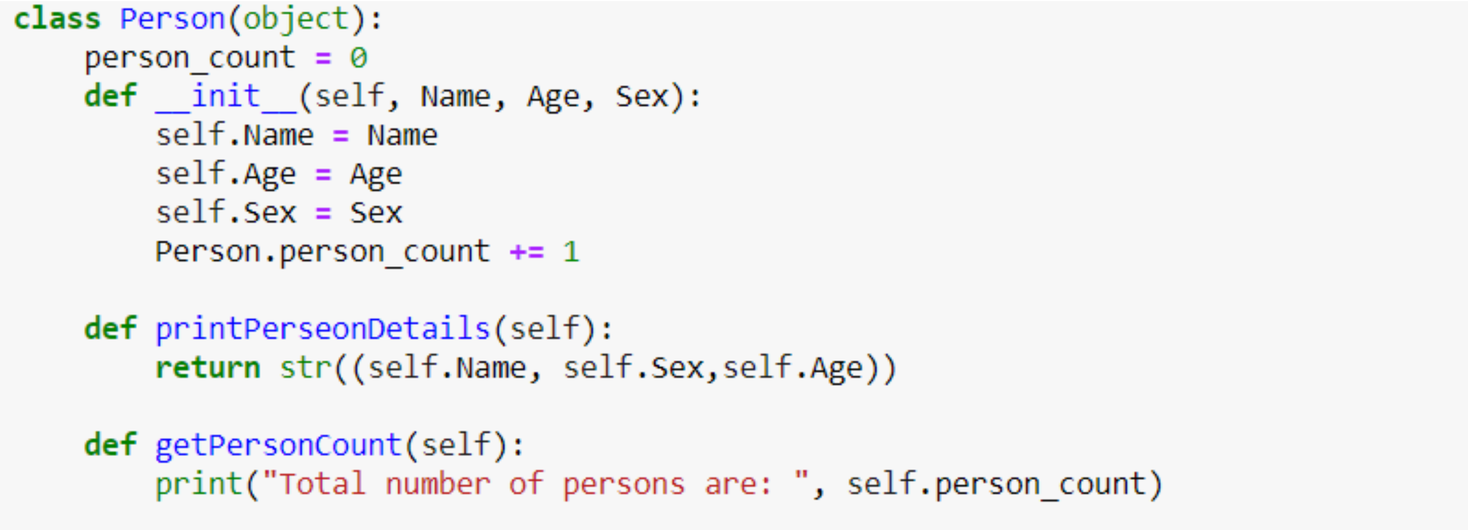
TASK 3- To create a Airline Booking Reservation System (e.g. classes Flight, Person, Employee, Passenger etc.)

Creating a class called flight.



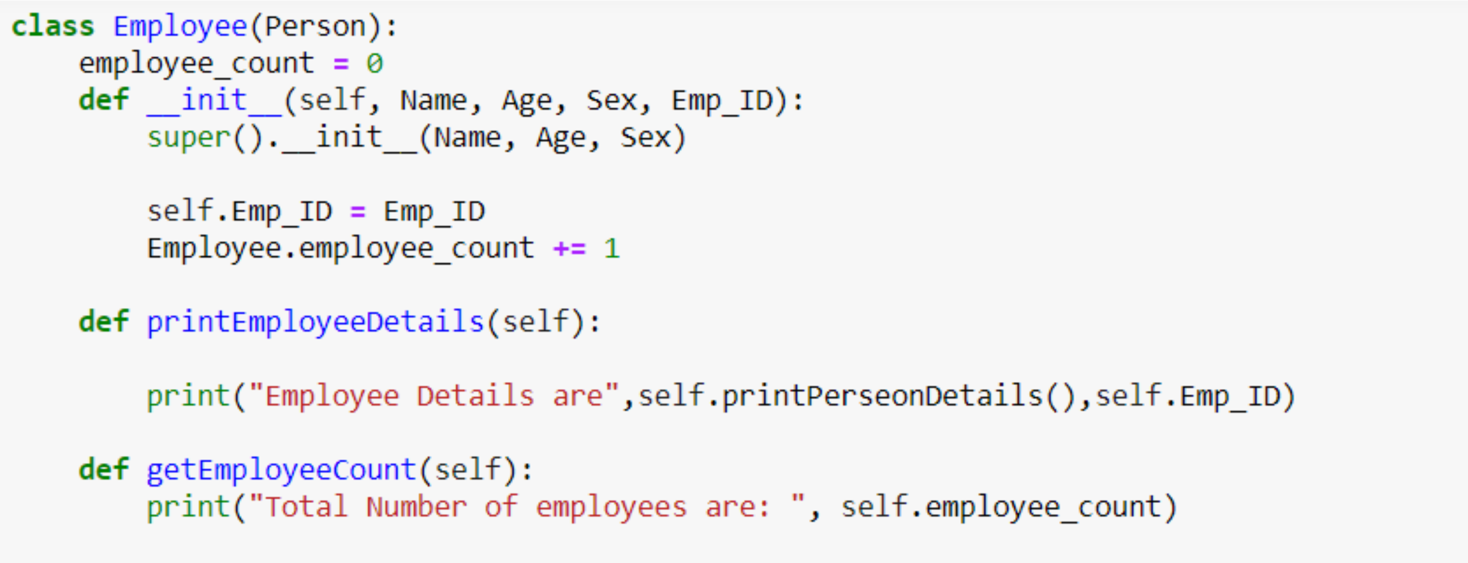
```
class Flight(object):
    flight_count = 0
    def __init__(self, Flight_Number, From, To, Date):
        self.Flight_Number = Flight_Number
        self.From_Loc = From
        self.To_Loc = To
        self.Date = Date
        Flight.flight_count += 1

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

Creating a class called person.



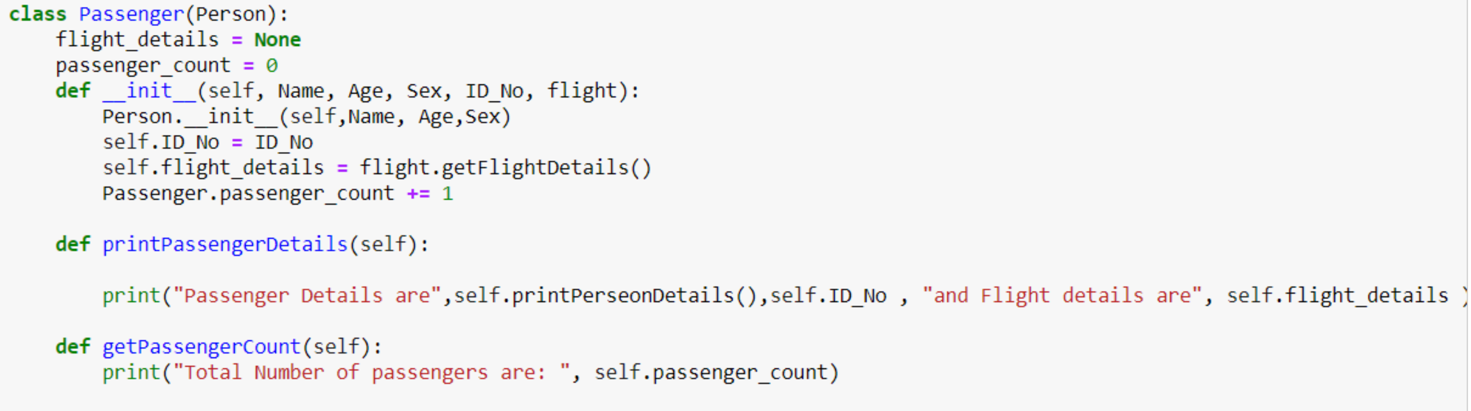
```
class Person(object):
    person_count = 0
    def __init__(self, Name, Age, Sex):
        self.Name = Name
        self.Age = Age
        self.Sex = Sex
        Person.person_count += 1

    def printPerseonDetails(self):
        return str((self.Name, self.Sex,self.Age))

    def getPersonCount(self):
        print("Total number of persons are: ", self.person_count)
```

Inheritance the properties of person in employee class.



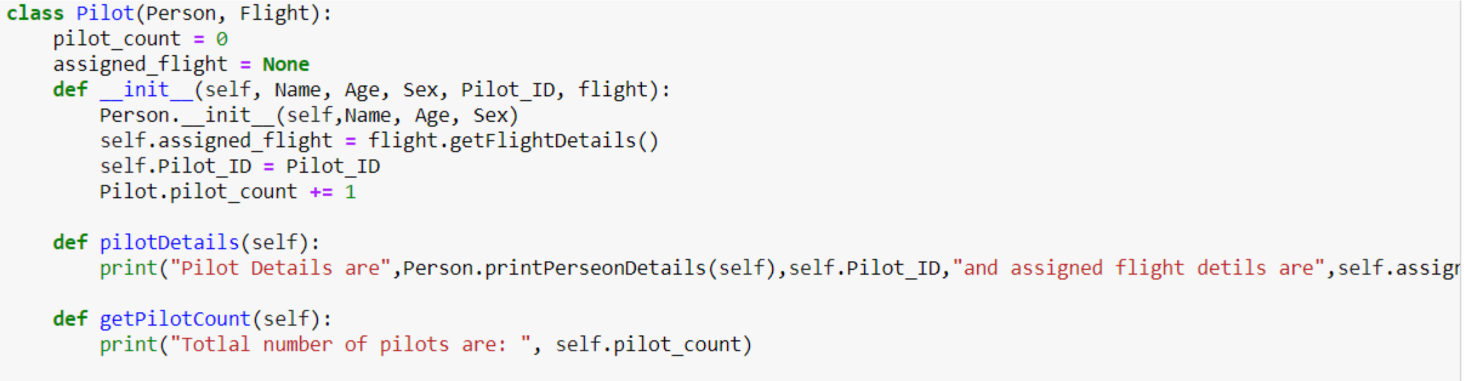
```
class Employee(Person):
    employee_count = 0
    def __init__(self, Name, Age, Sex, Emp_ID):
        super().__init__(Name, Age, Sex)

        self.Emp_ID = Emp_ID
        Employee.employee_count += 1

    def printEmployeeDetails(self):

        print("Employee Details are",self.printPerseonDetails(),self.Emp_ID)

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

Inheritance the properties of person in passenger class.



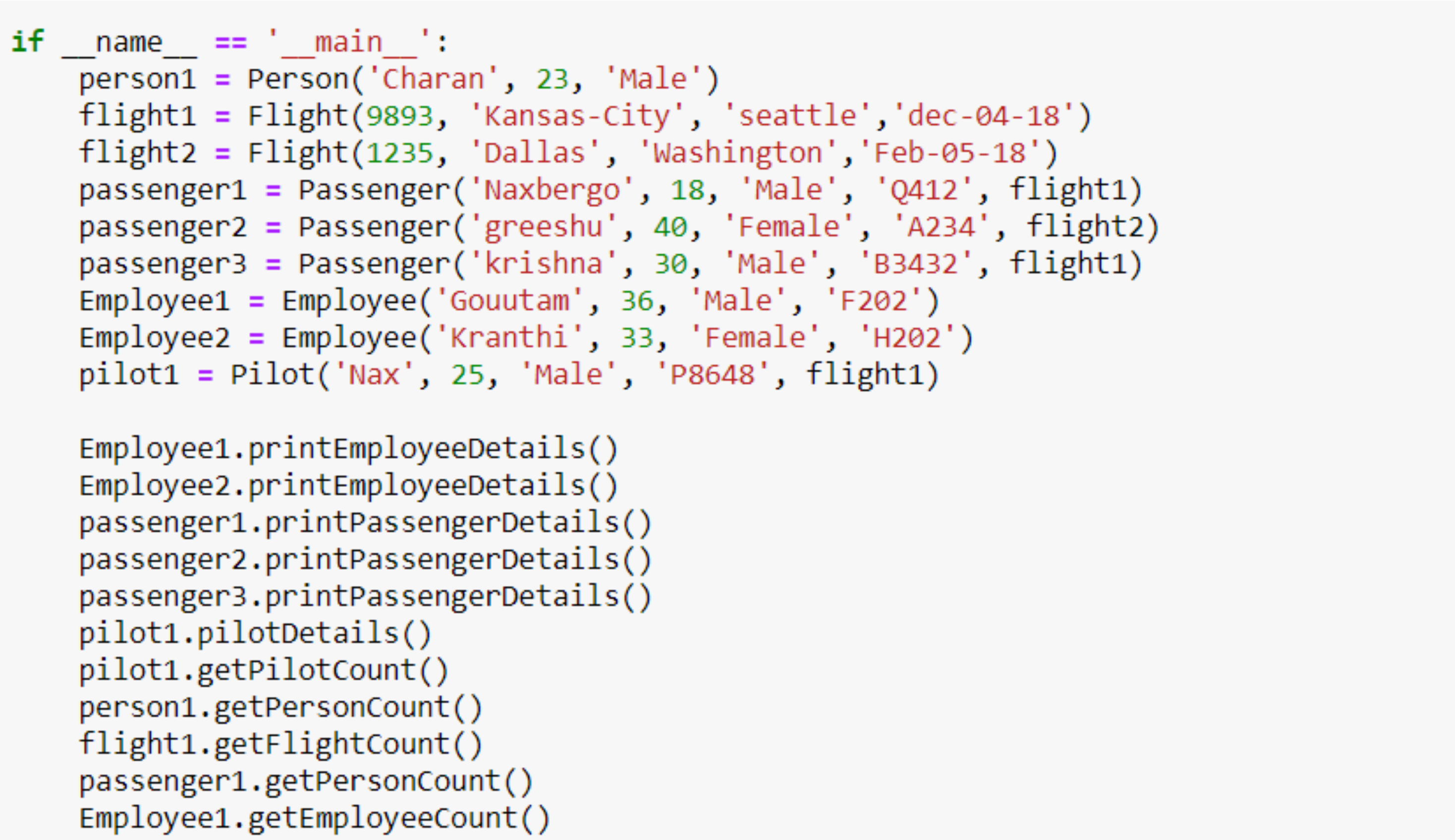
```
class Passenger(Person):
    flight_details = None
    passenger_count = 0
    def __init__(self, Name, Age, Sex, ID_No, flight):
        Person.__init__(self,Name, Age,Sex)
        self.ID_No = ID_No
        self.flight_details = flight.getFlightDetails()
        Passenger.passenger_count += 1

    def printPassengerDetails(self):

        print("Passenger Details are",self.printPerseonDetails(),self.ID_No , "and Flight details are", self.flight_details )

    def getPassengerCount(self):
        print("Total Number of passengers are: ", self.passenger_count)
```

Creating class called pilot and inheritance the properties of person and flight.



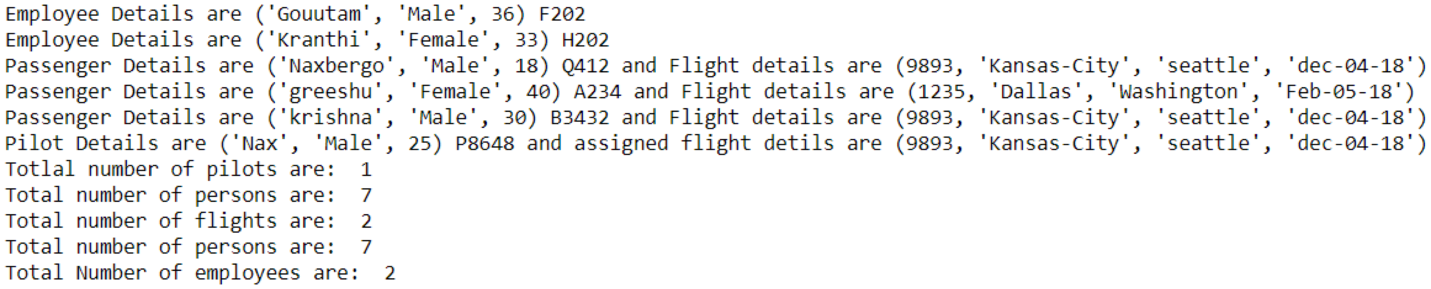
```
class Pilot(Person, Flight):
    pilot_count = 0
    assigned_flight = None
    def __init__(self, Name, Age, Sex, Pilot_ID, flight):
        Person.__init__(self,Name, Age, Sex)
        self.assigned_flight = flight.getFlightDetails()
        self.Pilot_ID = Pilot_ID
        Pilot.pilot_count += 1

    def pilotDetails(self):
        print("Pilot Details are",Person.printPerseonDetails(self),self.Pilot_ID,"and assigned flight detils are",self.assign

    def getPilotCount(self):
        print("Totlal number of pilots are: ", self.pilot_count)
```

Properties of class.



```
if __name__ == '__main__':
    person1 = Person('Charan', 23, 'Male')
    flight1 = Flight(9893, 'Kansas-City', 'seattle','dec-04-18')
    flight2 = Flight(1235, 'Dallas', 'Washington','Feb-05-18')
    passenger1 = Passenger('Naxbergo', 18, 'Male', 'Q412', flight1)
    passenger2 = Passenger('greeshu', 40, 'Female', 'A234', flight2)
    passenger3 = Passenger('krishna', 30, 'Male', 'B3432', flight1)
    Employee1 = Employee('Gouutam', 36, 'Male', 'F202')
    Employee2 = Employee('Kranthi', 33, 'Female', 'H202')
    pilot1 = Pilot('Nax', 25, 'Male', 'P8648', flight1)

    Employee1.printEmployeeDetails()
    Employee2.printEmployeeDetails()
    passenger1.printPassengerDetails()
    passenger2.printPassengerDetails()
    passenger3.printPassengerDetails()
    pilot1.pilotDetails()
    pilot1.getPilotCount()
    person1.getPersonCount()
    flight1.getFlightCount()
    passenger1.getPersonCount()
    Employee1.getEmployeeCount()
```

Output:



```
Employee Details are ('Gouutam', 'Male', 36) F202
Employee Details are ('Kranthi', 'Female', 33) H202
Passenger Details are ('Naxbergo', 'Male', 18) Q412 and Flight details are (9893, 'Kansas-City', 'seattle', 'dec-04-18')
Passenger Details are ('greeshu', 'Female', 40) A234 and Flight details are (1235, 'Dallas', 'Washington', 'Feb-05-18')
Passenger Details are ('krishna', 'Male', 30) B3432 and Flight details are (9893, 'Kansas-City', 'seattle', 'dec-04-18')
Pilot Details are ('Nax', 'Male', 25) P8648 and assigned flight detils are (9893, 'Kansas-City', 'seattle', 'dec-04-18')
Totlal 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
```

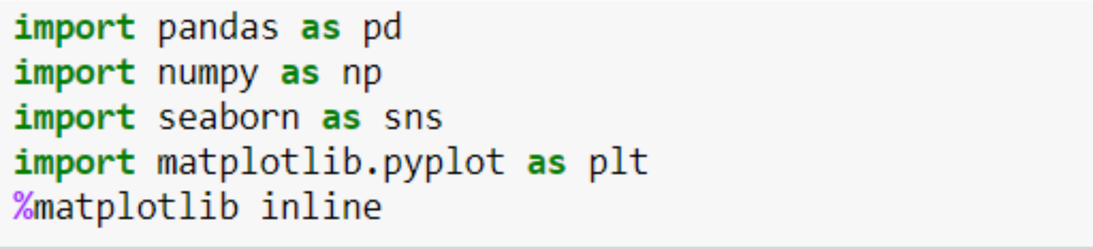
Summary:

- We have created different classes and some of the classes inherit the super class which may be multiple.
- Th constructur of the child class can always call parent class to get its attribute.Through the above Implementations we have used inheritence properties.

TASK 4- Create Multiple Regression and Evaluate the model

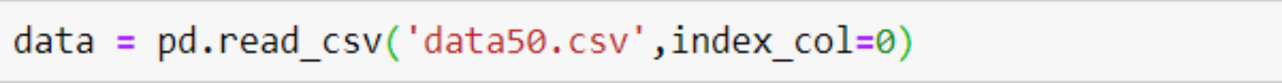
using RMSE and R2.

importing packages



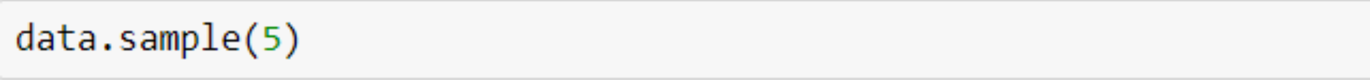
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Reading the csv file into data.

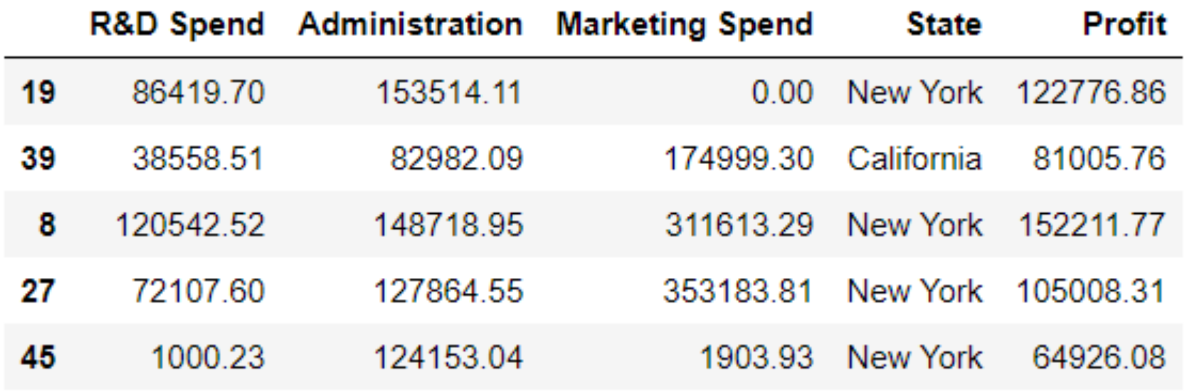


```
data = pd.read_csv('data50.csv',index_col=0)
```

print sample data

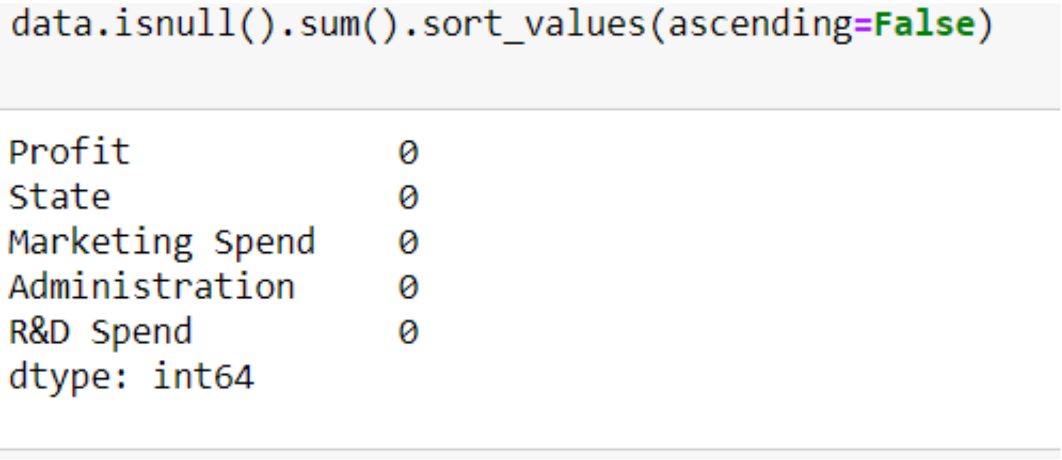


```
data.sample(5)
```



	R&D Spend	Administration	Marketing Spend	State	Profit
19	86419.70	153514.11	0.00	New York	122776.86
39	38558.51	82982.09	174999.30	California	81005.76
8	120542.52	148718.95	311613.29	New York	152211.77
27	72107.60	127864.55	353183.81	New York	105008.31
45	1000.23	124153.04	1903.93	New York	64926.08

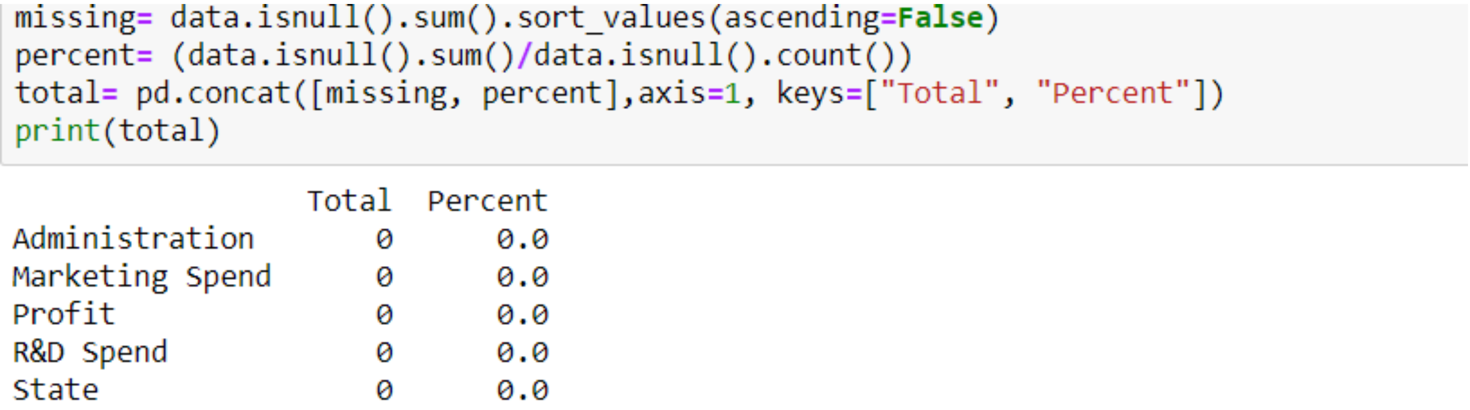
check for missing values



```
data.isnull().sum().sort_values(ascending=False)
```

```
Profit             0
State              0
Marketing Spend    0
Administration     0
R&D Spend          0
dtype: int64
```

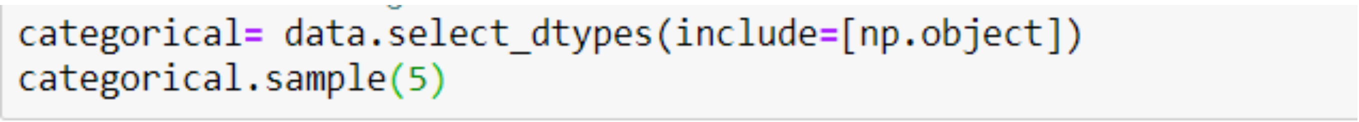
Table of Missing values



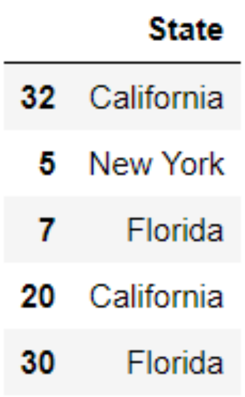
```
missing= data.isnull().sum().sort_values(ascending=False)
percent= (data.isnull().sum()/data.isnull().count())
total= pd.concat([missing, percent],axis=1, keys=["Total", "Percent"])
print(total)
```

```
                 Total  Percent
Administration       0      0.0
Marketing Spend      0      0.0
Profit               0      0.0
R&D Spend            0      0.0
State                0      0.0
```

Geting all the categorical variables

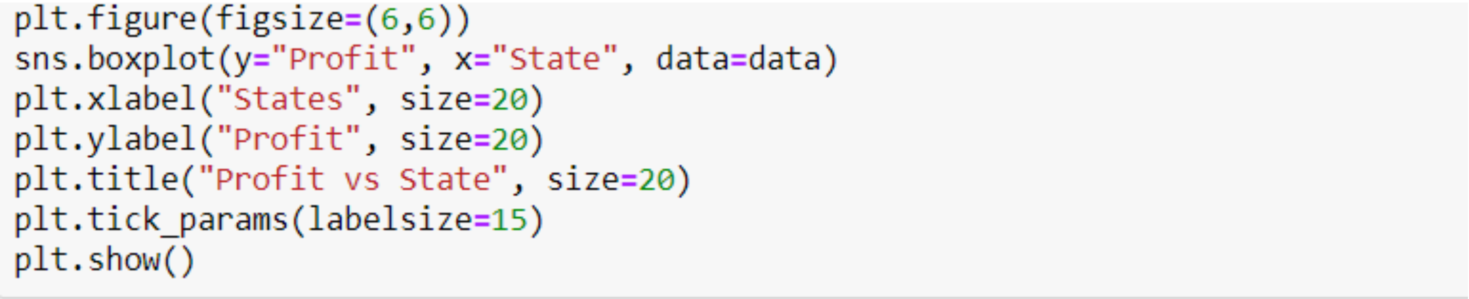


```
categorical= data.select_dtypes(include=[np.object])
categorical.sample(5)
```

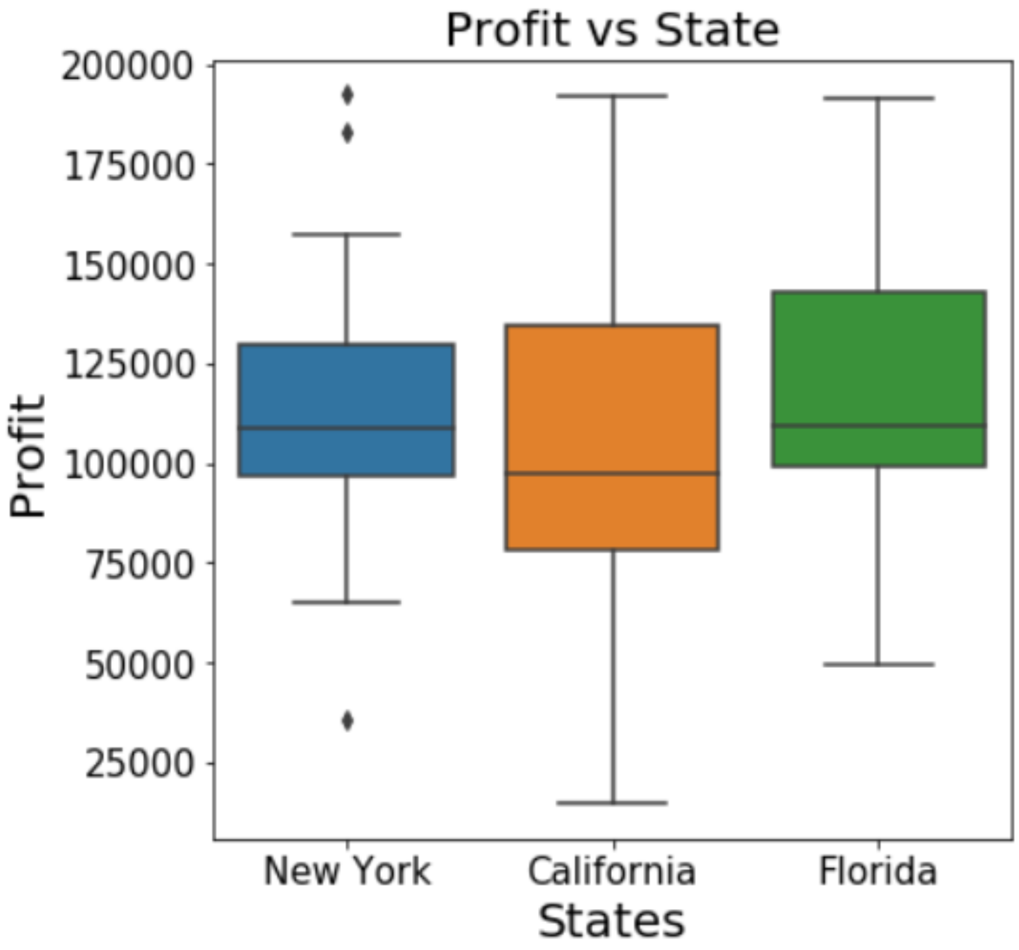


	State
32	California
5	New York
7	Florida
20	California
30	Florida

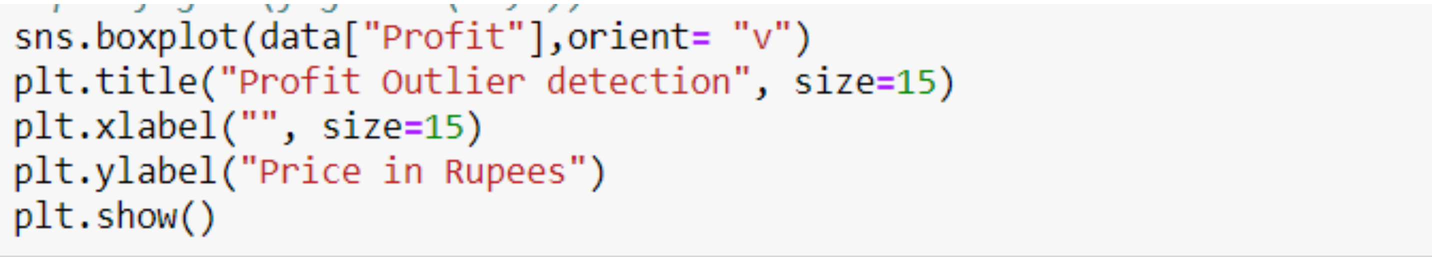
Boxplot of Profit and Marketing spend



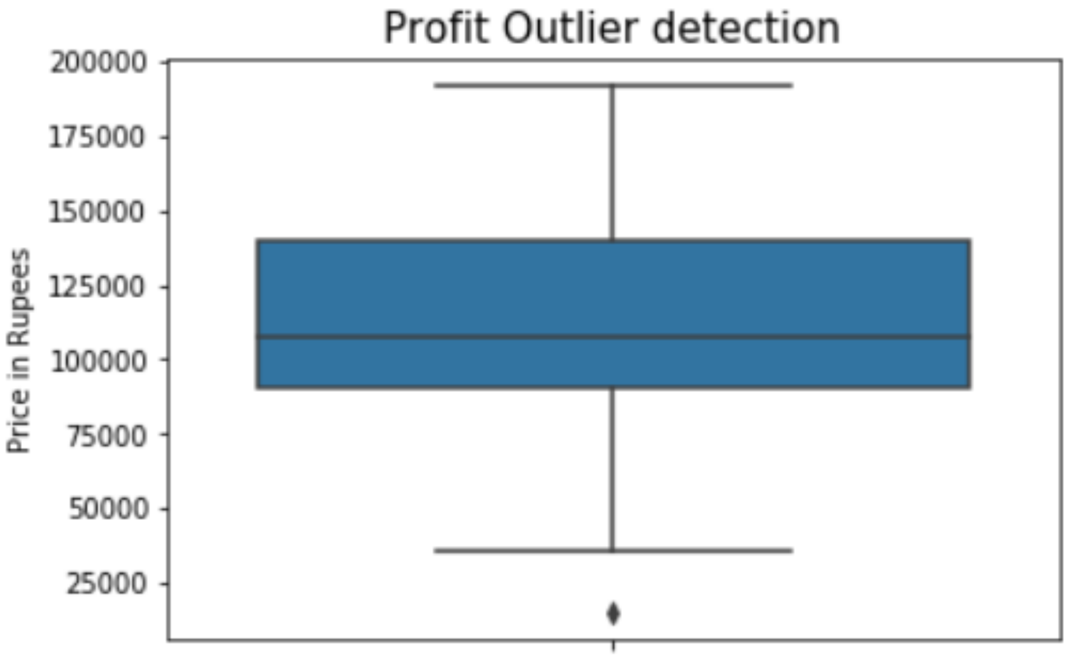
```
plt.figure(figsize=(6,6))
sns.boxplot(y="Profit", x="State", data=data)
plt.xlabel("States", size=20)
plt.ylabel("Profit", size=20)
plt.title("Profit vs State", size=20)
plt.tick_params(labelsize=15)
plt.show()
```



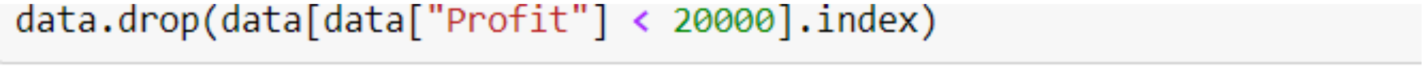
Outlier detection.



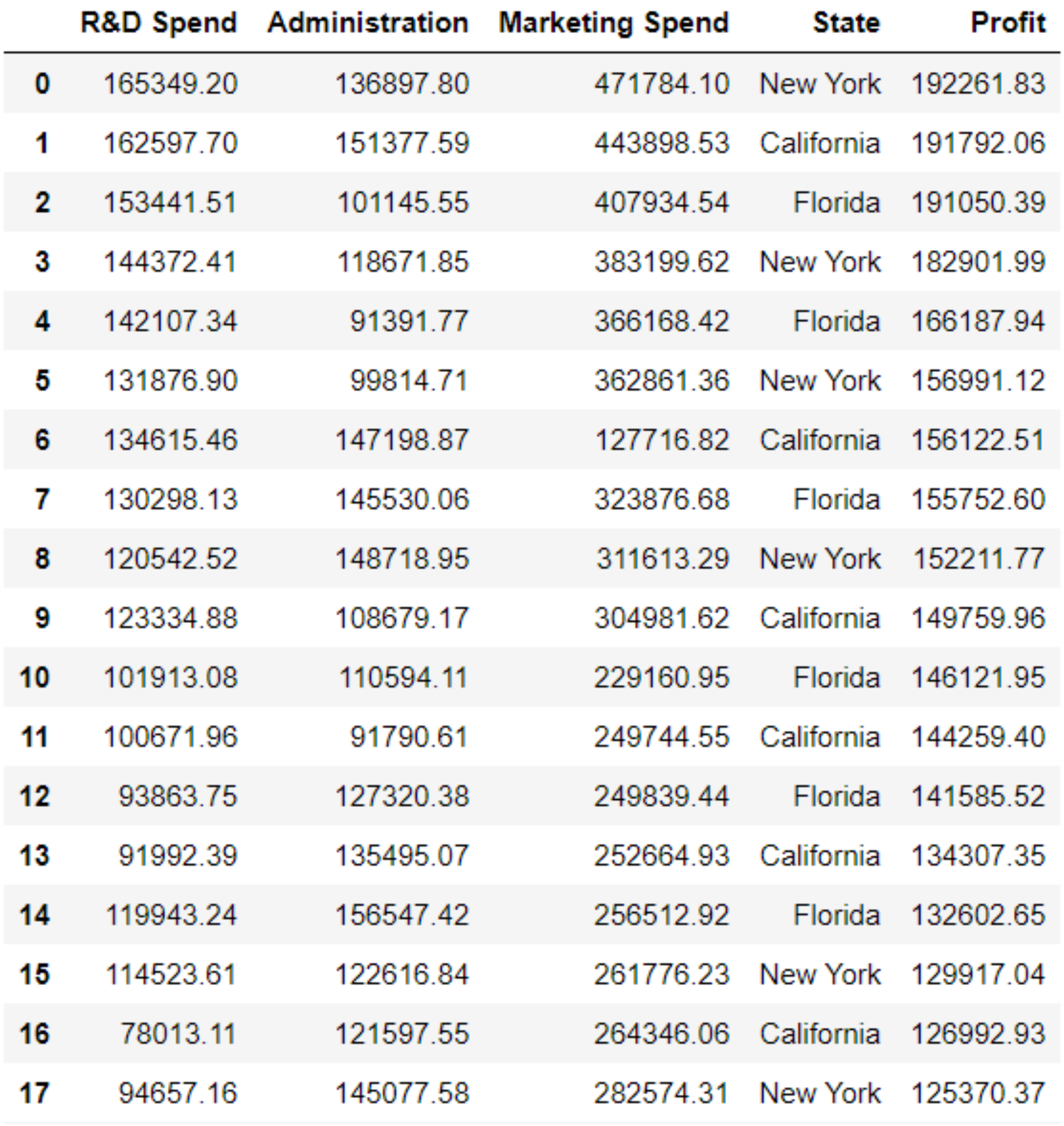
```
sns.boxplot(data["Profit"],orient= "v")
plt.title("Profit Outlier detection", size=15)
plt.xlabel("", size=15)
plt.ylabel("Price in Rupees")
plt.show()
```



Remove outlier.

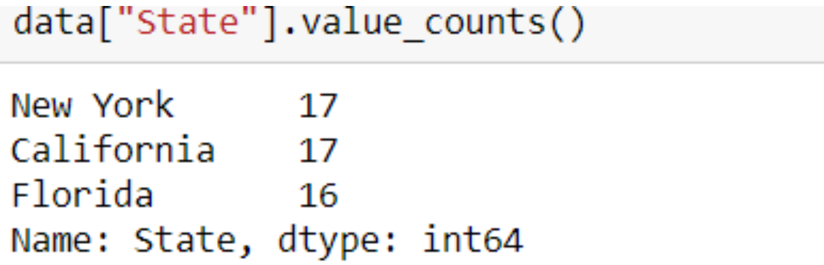


```
data.drop(data[data["Profit"] < 20000].index)
```



	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37

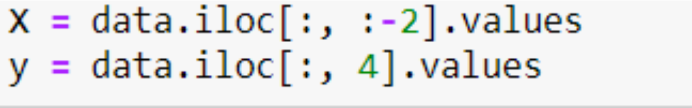
Here State is a categorical varible which we need to convert to numeric



```
data["State"].value_counts()
```

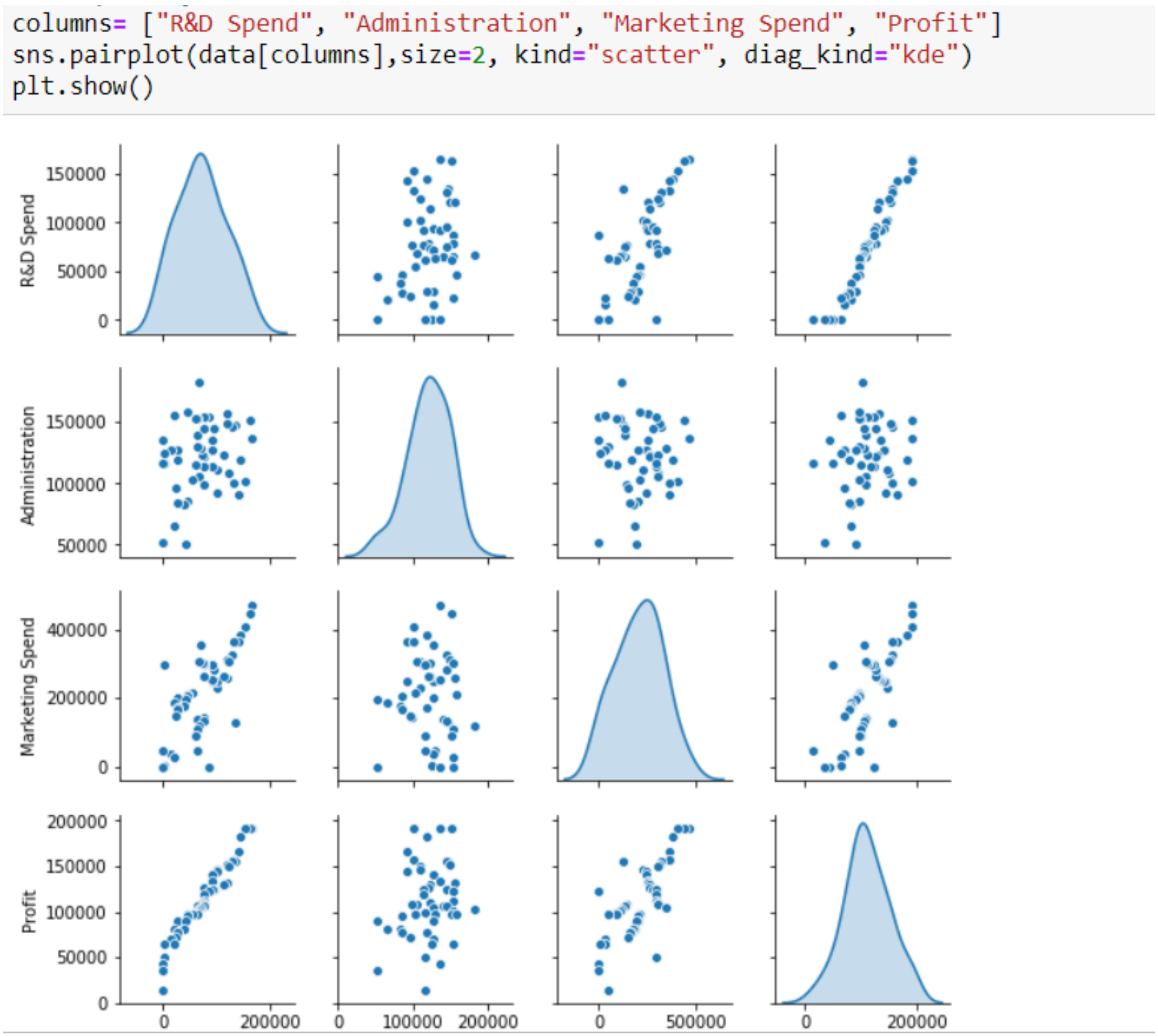
```
New York      17
California    17
Florida       16
Name: State, dtype: int64
```

Split the data into training and test set



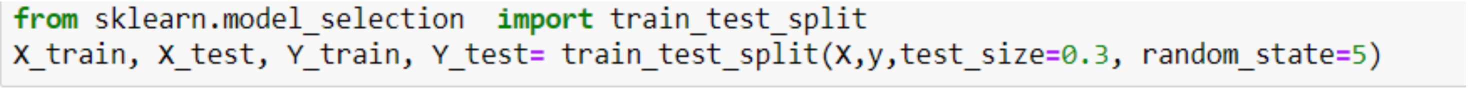
```
X = data.iloc[:, :-2].values
y = data.iloc[:, 4].values
```

Pairplot of numeric variables



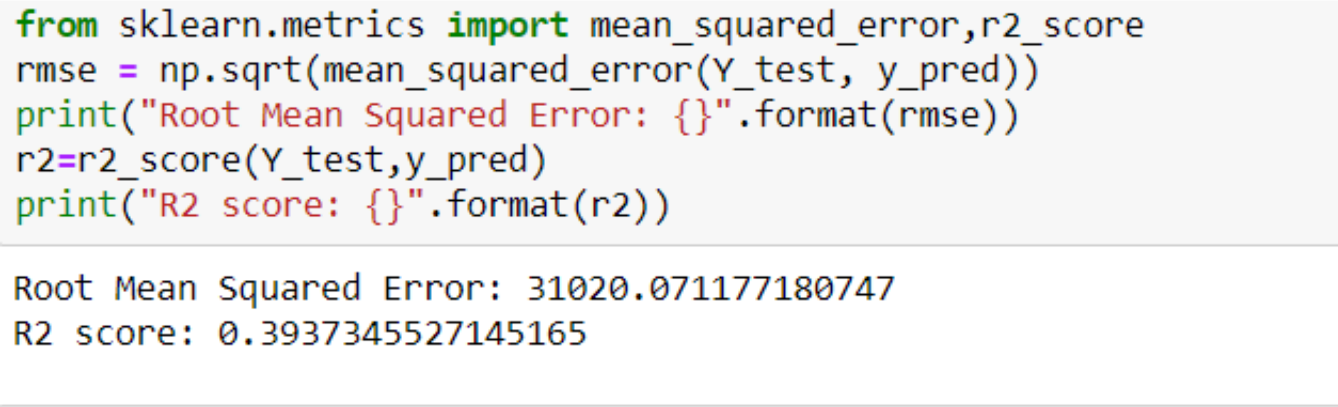
```
columns= ["R&D Spend", "Administration", "Marketing Spend", "Profit"]
sns.pairplot(data[columns],size=2, kind="scatter", diag_kind="kde")
plt.show()
```

Split data into training and testing set



```
from sklearn.model_selection  import train_test_split
X_train, X_test, Y_train, Y_test= train_test_split(X,y,test_size=0.3, random_state=5)
```

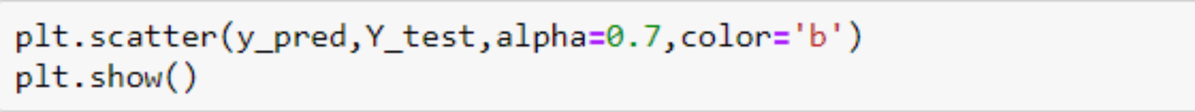
RMSE on train set.



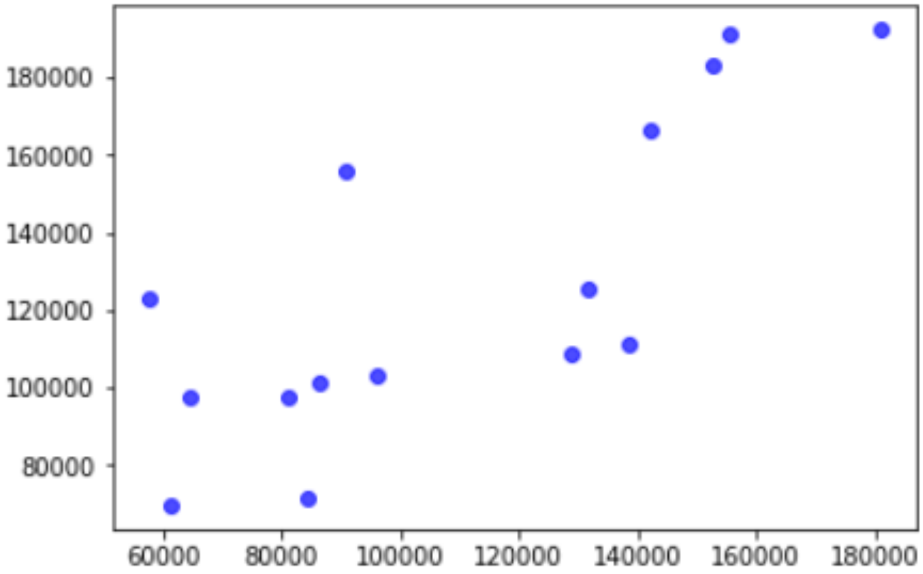
```
from sklearn.metrics import mean_squared_error,r2_score
rmse = np.sqrt(mean_squared_error(Y_test, y_pred))
print("Root Mean Squared Error: {}".format(rmse))
r2=r2_score(Y_test,y_pred)
print("R2 score: {}".format(r2))
```

```
Root Mean Squared Error: 31020.071177180747
R2 score: 0.3937345527145165
```

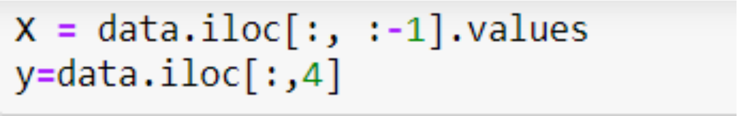
Scatter plot



```
plt.scatter(y_pred,Y_test,alpha=0.7,color='b')
plt.show()
```

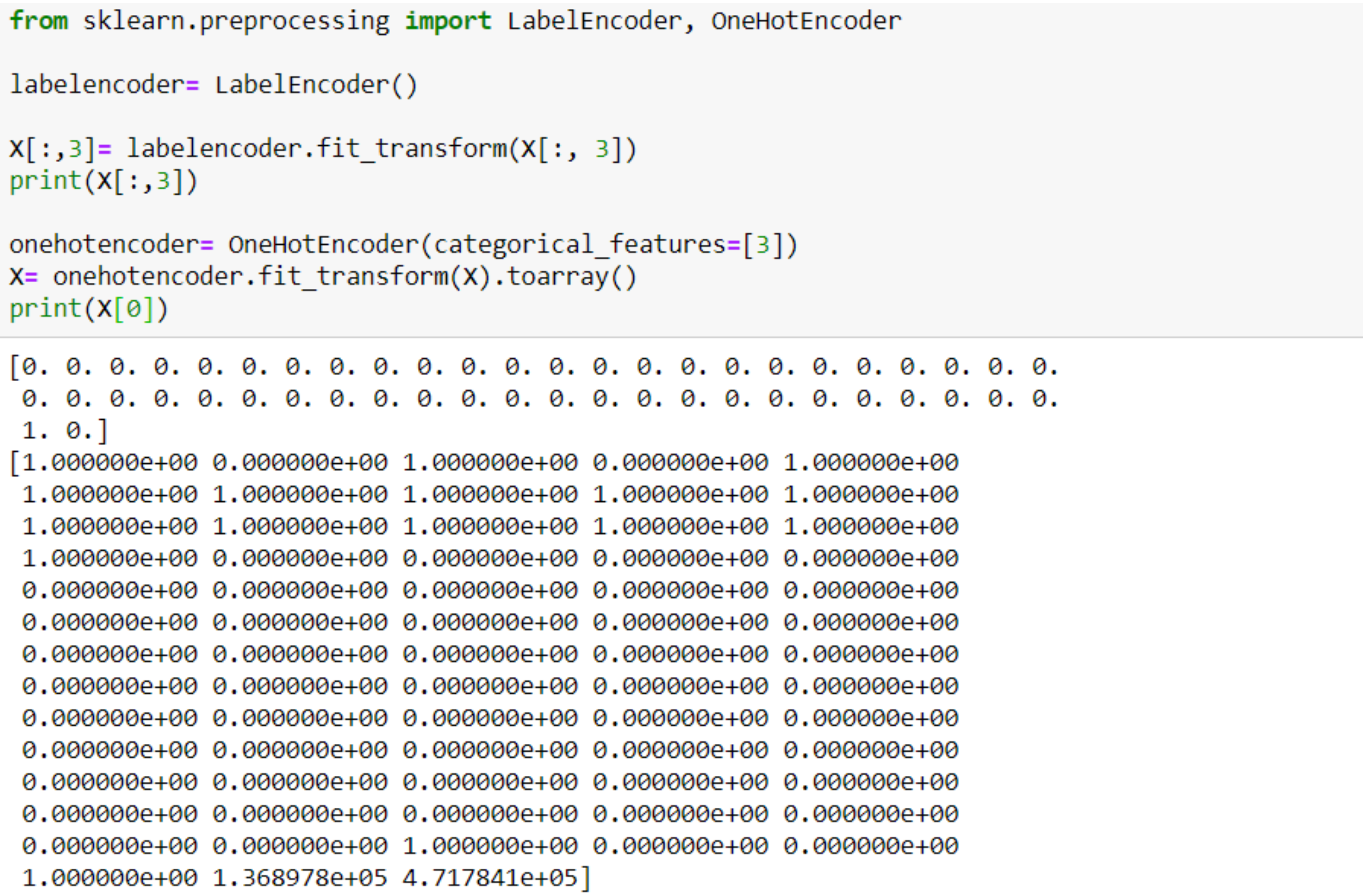


After Applying OneHotEncoding there is change in the scores



```
X = data.iloc[:, :-1].values
y=data.iloc[:,4]
```

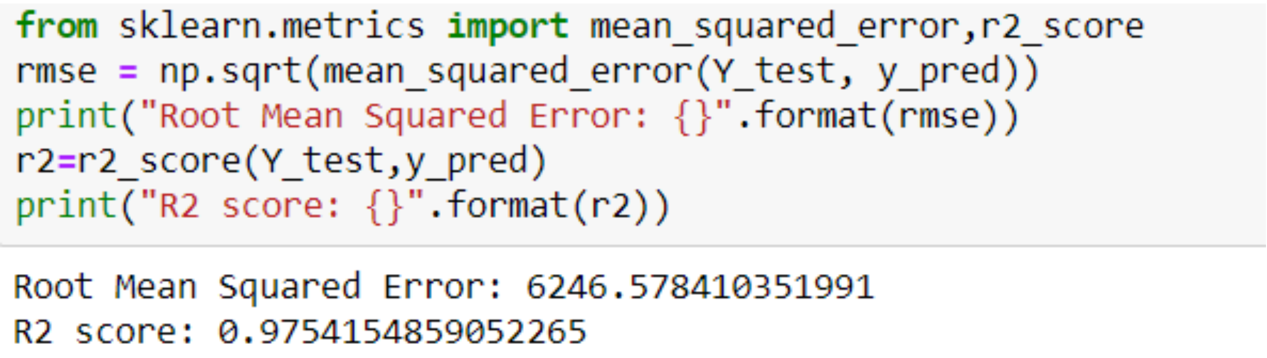
Encoding categorical data



```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder= LabelEncoder()

X[:,3]= labelencoder.fit_transform(X[:, 3])
print(X[:,3])

onehotencoder= OneHotEncoder(categorical_features=[3])
X= onehotencoder.fit_transform(X).toarray()
print(X[0])
```

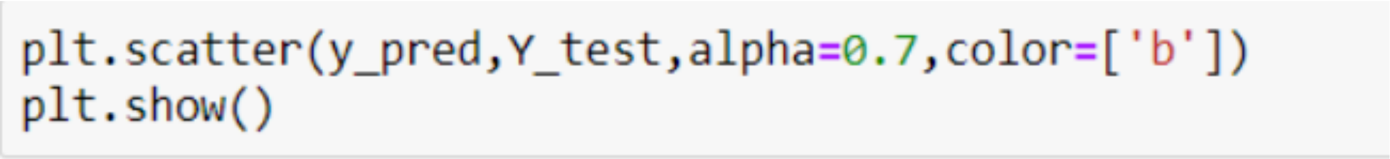
```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 1. 0.]
[1.000000e+00 0.000000e+00 1.000000e+00 0.000000e+00 1.000000e+00
 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00
 1.000000e+00 1.368978e+05 4.717841e+05]
```

RMSE on train set

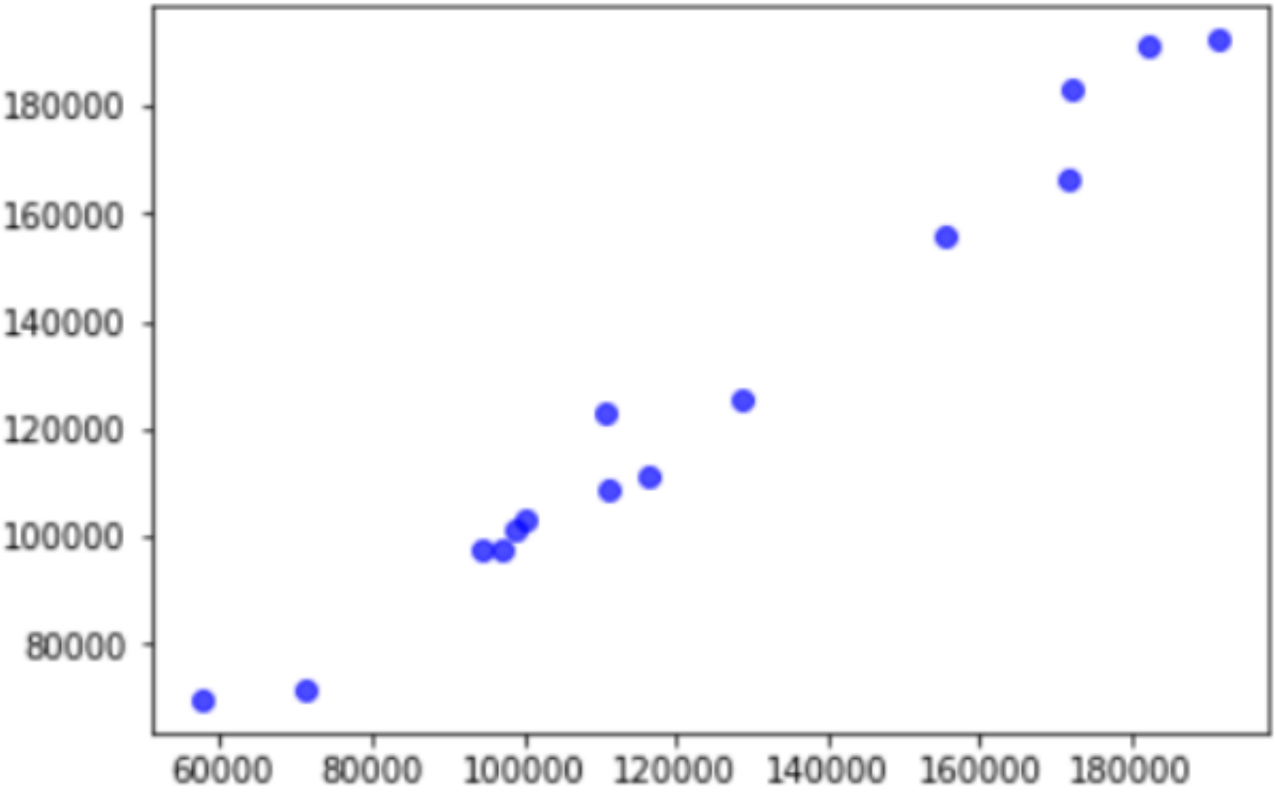


```
from sklearn.metrics import mean_squared_error,r2_score
rmse = np.sqrt(mean_squared_error(Y_test, y_pred))
print("Root Mean Squared Error: {}".format(rmse))
r2=r2_score(Y_test,y_pred)
print("R2 score: {}".format(r2))
```

```
Root Mean Squared Error: 6246.578410351991
R2 score: 0.9754154859052265
```



```
plt.scatter(y_pred,Y_test,alpha=0.7,color=['b'])
plt.show()
```

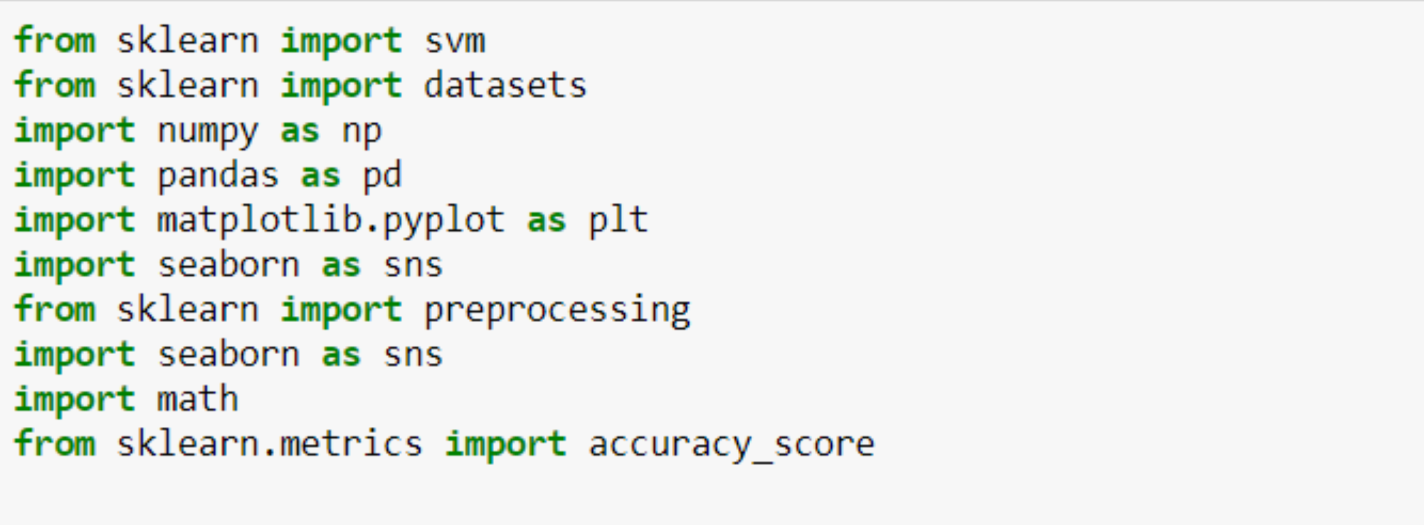


- ## Summary:

- Through various examples we have shown various was of EDA data.
- By using label encoder and one hot encoding(in a lament way it has one hot bit and remaining as normal bits for a given feature or class based on context)

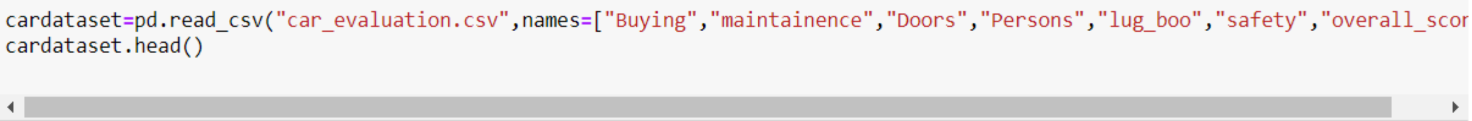
TASK 5- Perform exploratory data analysis on the data set and apply the three classification algorithms Naïve Baye's, SVM and KNN .

importing packages

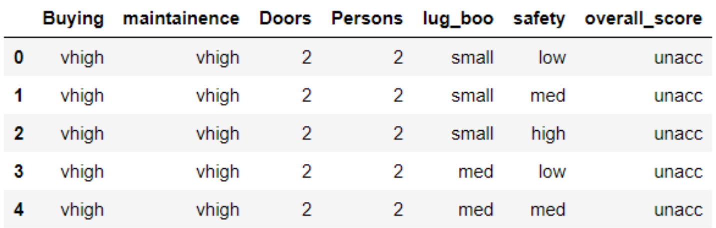


```
from sklearn import svm
from sklearn import datasets
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import seaborn as sns
import math
from sklearn.metrics import accuracy_score
```

Loading csv file data to cardataset.

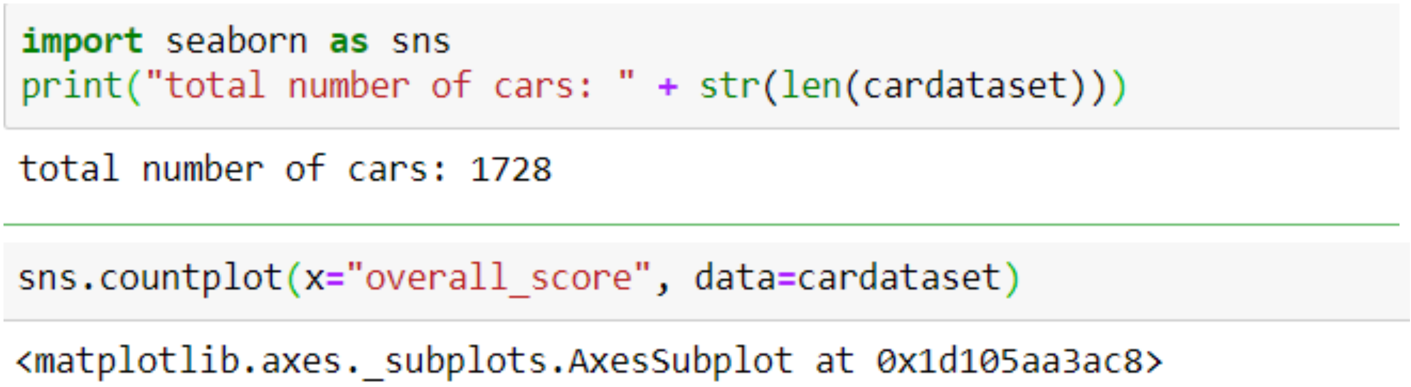


```
cardataset=pd.read_csv("car_evaluation.csv",names=["Buying","maintainence","Doors","Persons","lug_boo","safety","overall_scor
cardataset.head()
```



	Buying	maintainence	Doors	Persons	lug_boo	safety	overall_score
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

Analyzing the data

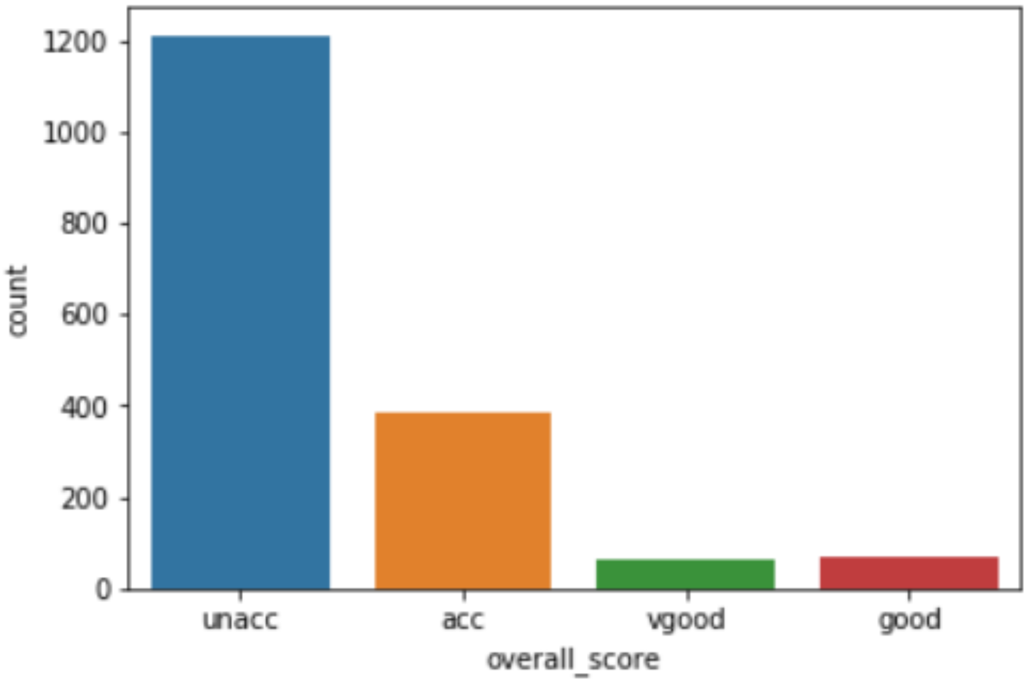


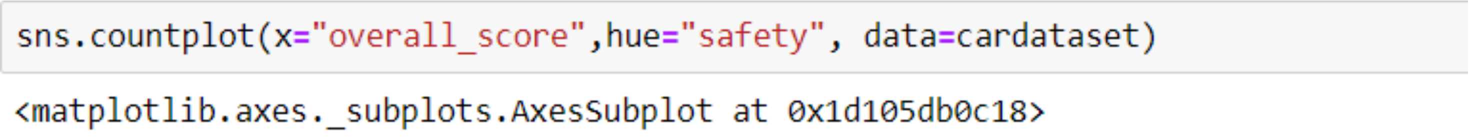
```
import seaborn as sns
print("total number of cars: " + str(len(cardataset)))
```

```
total number of cars: 1728
```

```
sns.countplot(x="overall_score", data=cardataset)
```

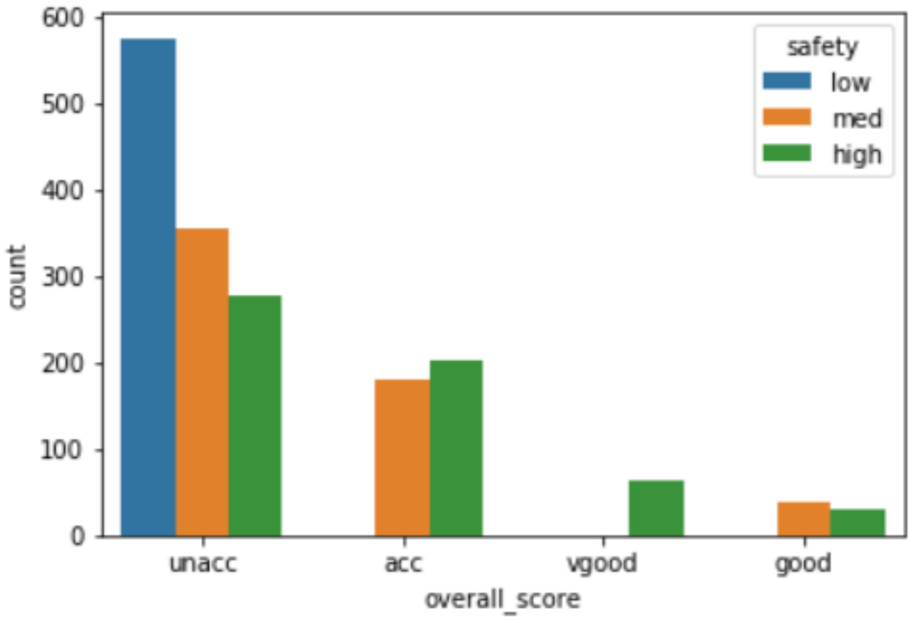
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d105aa3ac8>
```

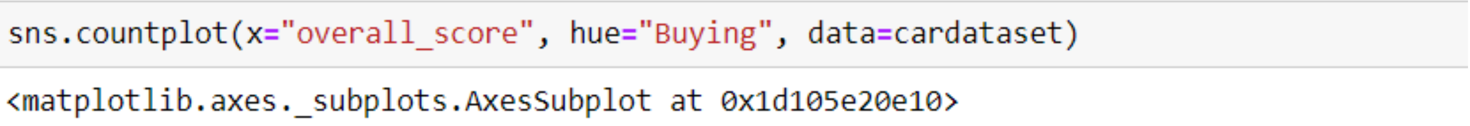




```
sns.countplot(x="overall_score",hue="safety", data=cardataset)
```

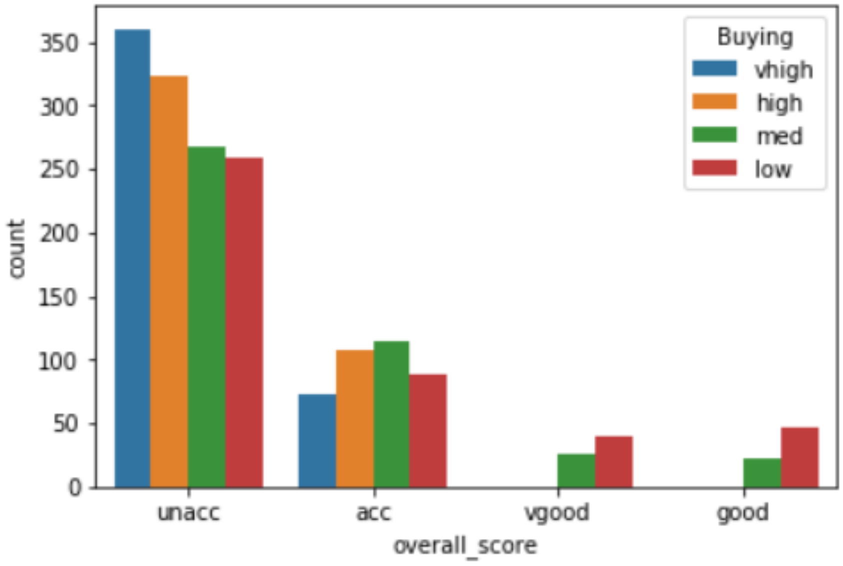
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d105db0c18>
```



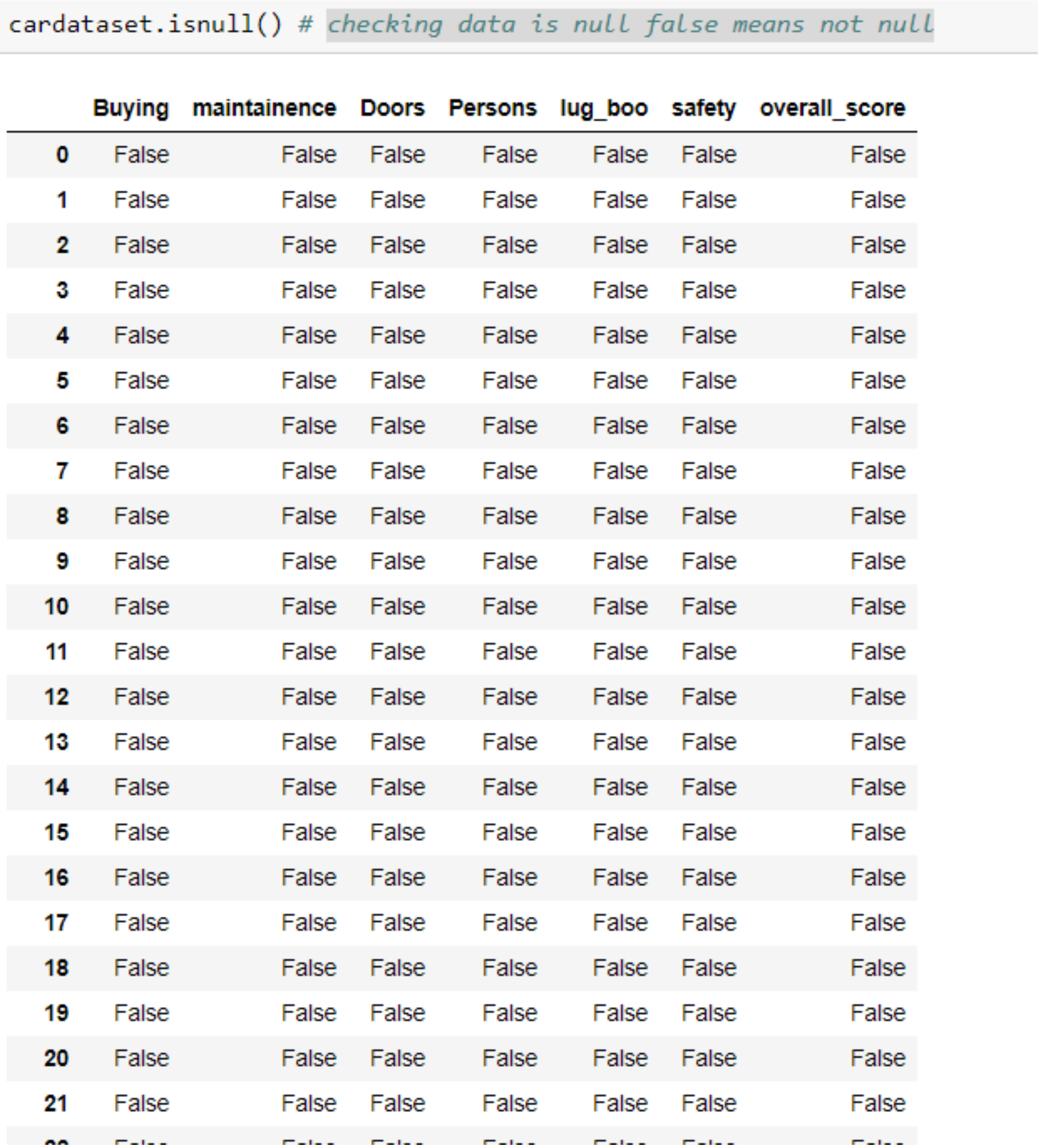


```
sns.countplot(x="overall_score", hue="Buying", data=cardataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d105e20e10>
```

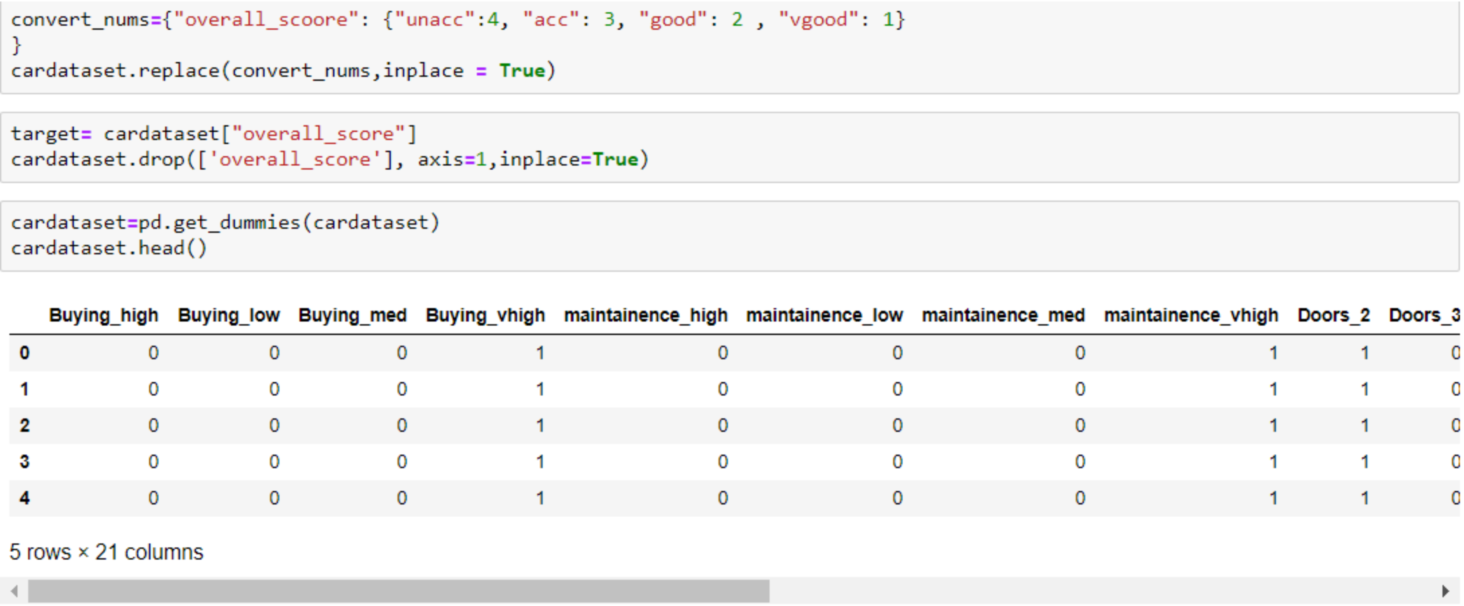


checking data is null false means not null



```
cardataset.isnull() # checking data is null false means not null
```

	Buying	maintainence	Doors	Persons	lug_boo	safety	overall_score
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False
12	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False
15	False	False	False	False	False	False	False
16	False	False	False	False	False	False	False
17	False	False	False	False	False	False	False
18	False	False	False	False	False	False	False
19	False	False	False	False	False	False	False
20	False	False	False	False	False	False	False
21	False	False	False	False	False	False	False
22	False	False	False	False	False	False	False



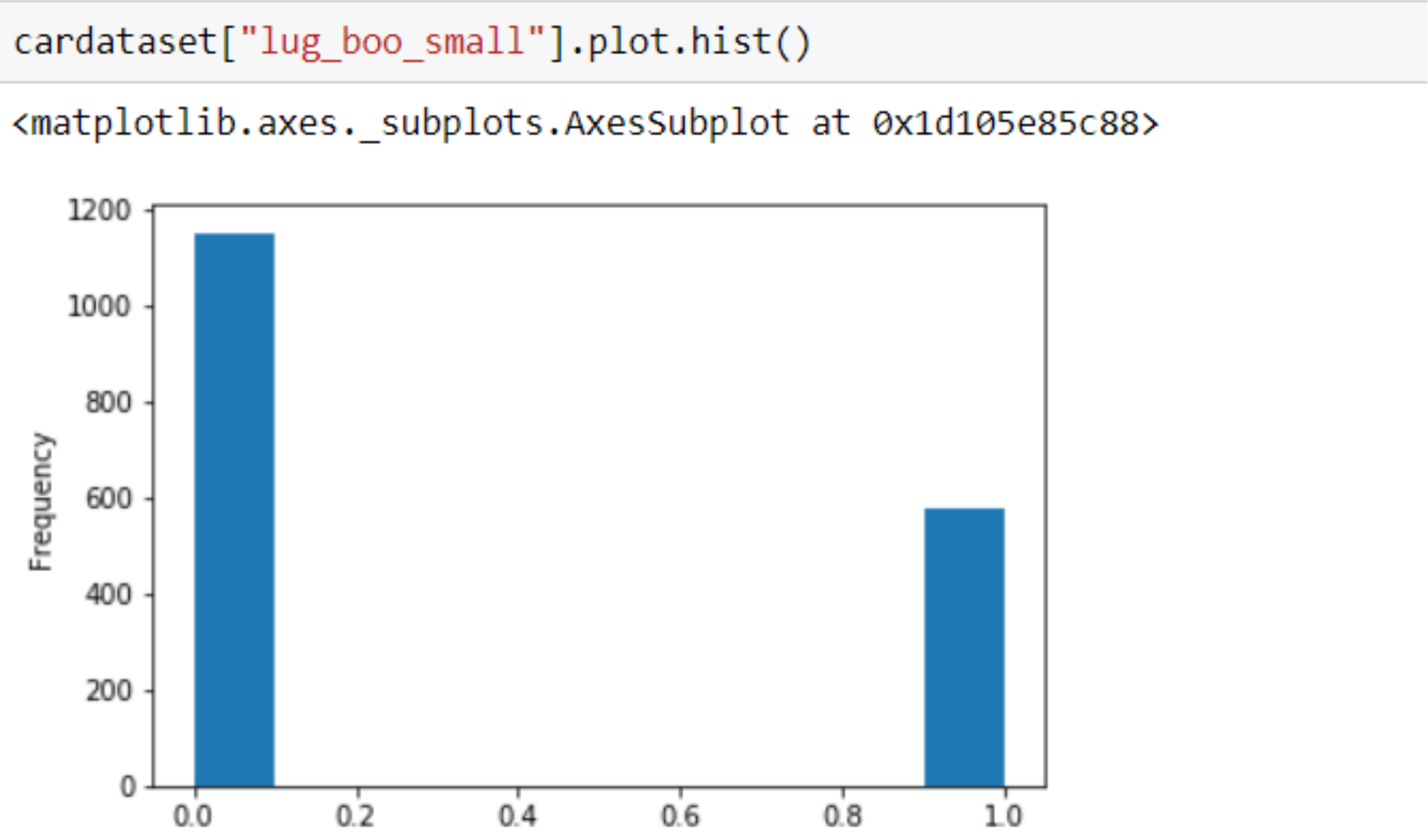
```
convert_nums={"overall_scoore": {"unacc":4, "acc": 3, "good": 2 , "vgood": 1}
}
cardataset.replace(convert_nums,inplace = True)
```

```
target= cardataset["overall_score"]
cardataset.drop(['overall_score'], axis=1,inplace=True)
```

```
cardataset=pd.get_dummies(cardataset)
cardataset.head()
```

	Buying_high	Buying_low	Buying_med	Buying_vhigh	maintainence_high	maintainence_low	maintainence_med	maintainence_vhigh	Doors_2	Doors_3
0	0	0	0	1	0	0	0	1	1	0
1	0	0	0	1	0	0	0	1	1	0
2	0	0	0	1	0	0	0	1	1	0
3	0	0	0	1	0	0	0	1	1	0
4	0	0	0	1	0	0	0	1	1	0

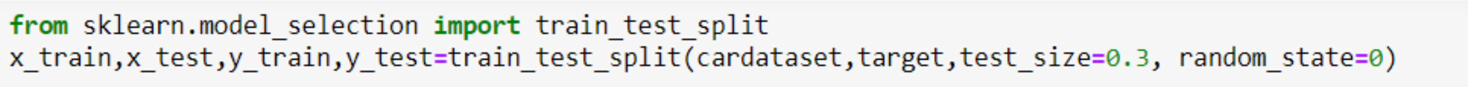
5 rows × 21 columns



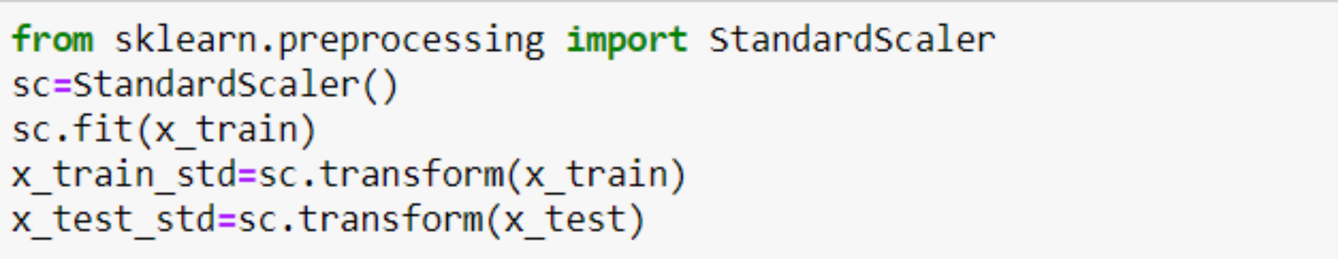
```
cardataset["lug_boo_small"].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d105e85c88>
```

Spliting the data into Testing and Training.

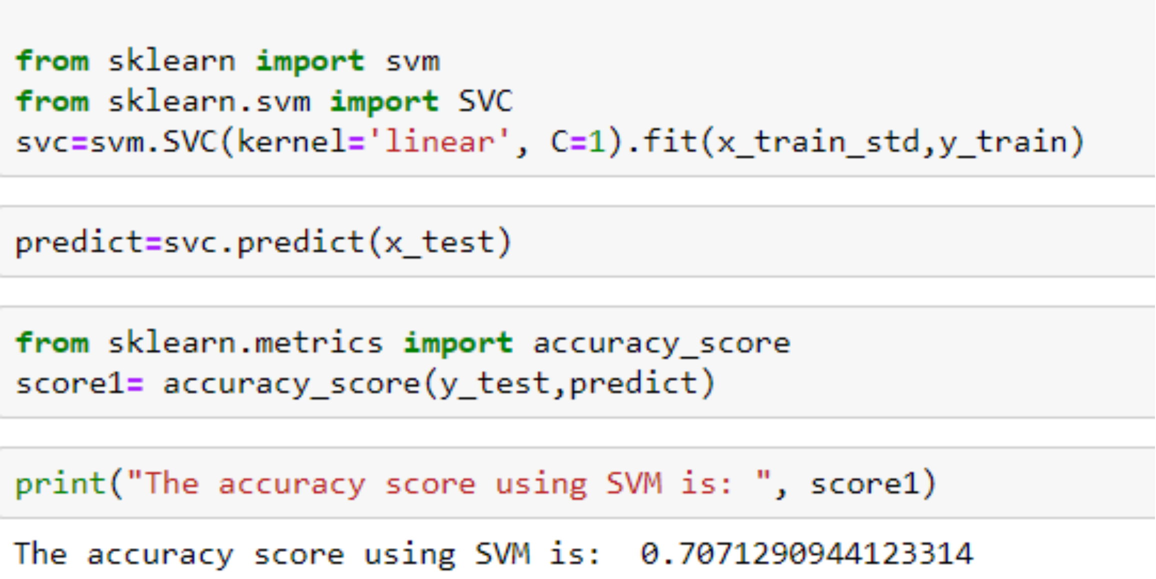


```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(cardataset,target,test_size=0.3, random_state=0)
```



```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_train)
x_train_std=sc.transform(x_train)
x_test_std=sc.transform(x_test)
```

SVM ALGORITHM



```
from sklearn import svm
from sklearn.svm import SVC
svc=svm.SVC(kernel='linear', C=1).fit(x_train_std,y_train)
```

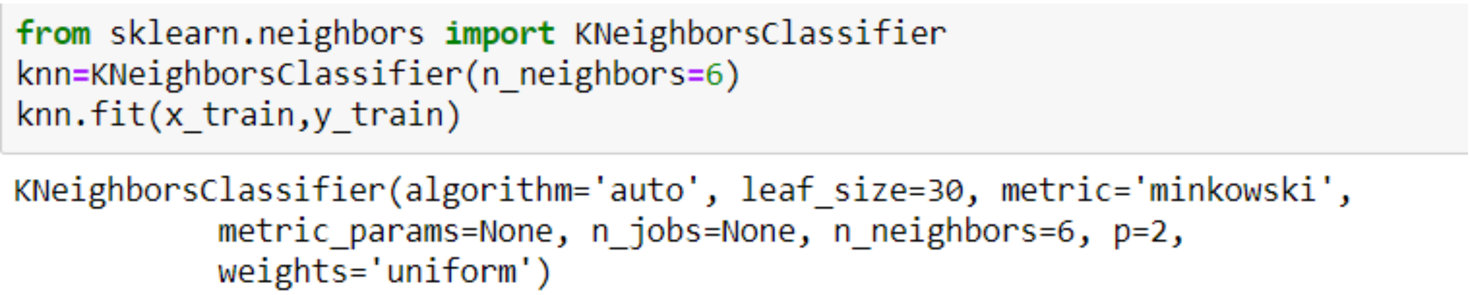
```
predict=svc.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
score1= accuracy_score(y_test,predict)
```

```
print("The accuracy score using SVM is: ", score1)
```

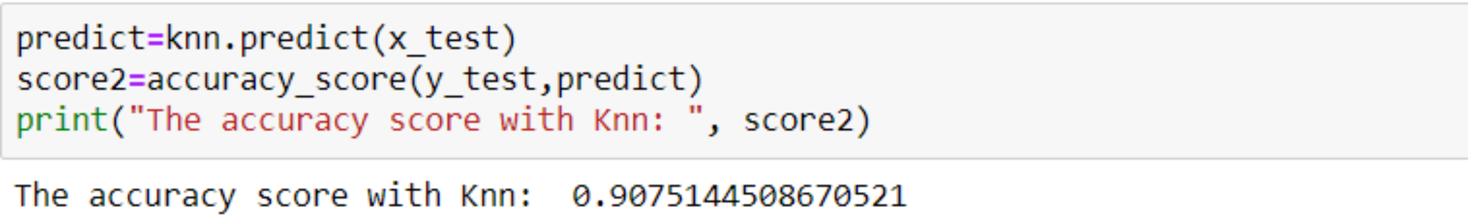
```
The accuracy score using SVM is:  0.7071290944123314
```

KNN ALGORITHM



```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=6)
knn.fit(x_train,y_train)
```

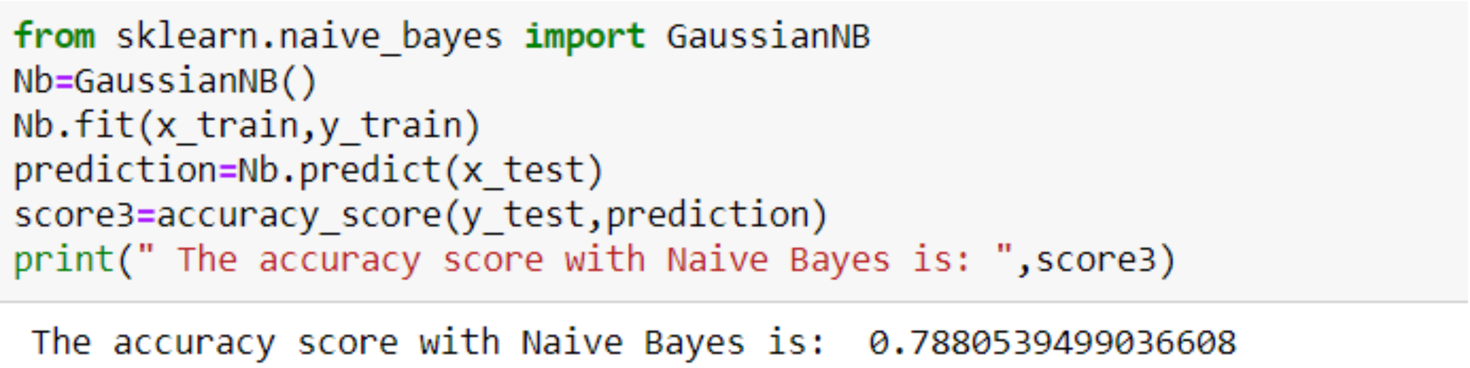
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=6, p=2,
           weights='uniform')
```



```
predict=knn.predict(x_test)
score2=accuracy_score(y_test,predict)
print("The accuracy score with Knn: ", score2)
```

```
The accuracy score with Knn:  0.9075144508670521
```

Naive Bayes



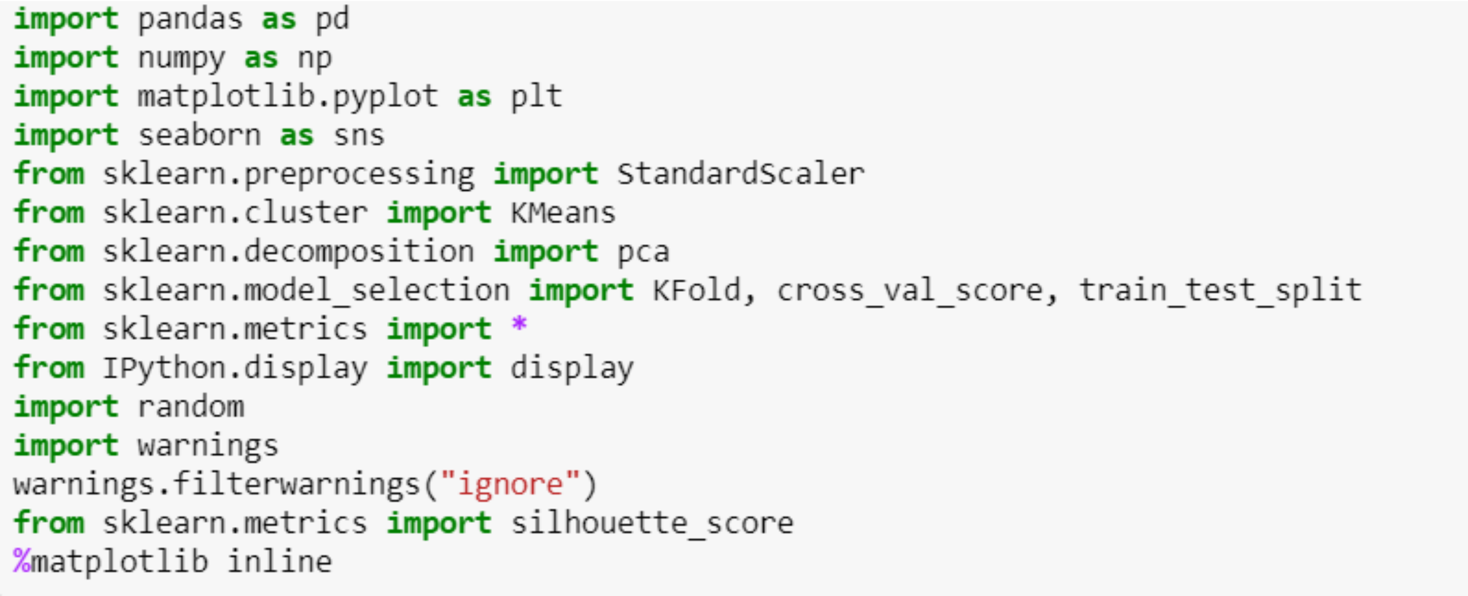
```
from sklearn.naive_bayes import GaussianNB
Nb=GaussianNB()
Nb.fit(x_train,y_train)
prediction=Nb.predict(x_test)
score3=accuracy_score(y_test,prediction)
print(" The accuracy score with Naive Bayes is: ",score3)
```

```
 The accuracy score with Naive Bayes is:  0.7880539499036608
```

Analyzing the score The accuracy of KNN algorithm is the best.

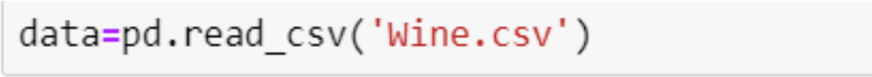
TASK 6- Apply K-means on the dataset and visualize the clusters using matplotlib or seaborn and Report which K is the best using the elbow method and Evaluate with silhouette score .

Import packages required for clustering.



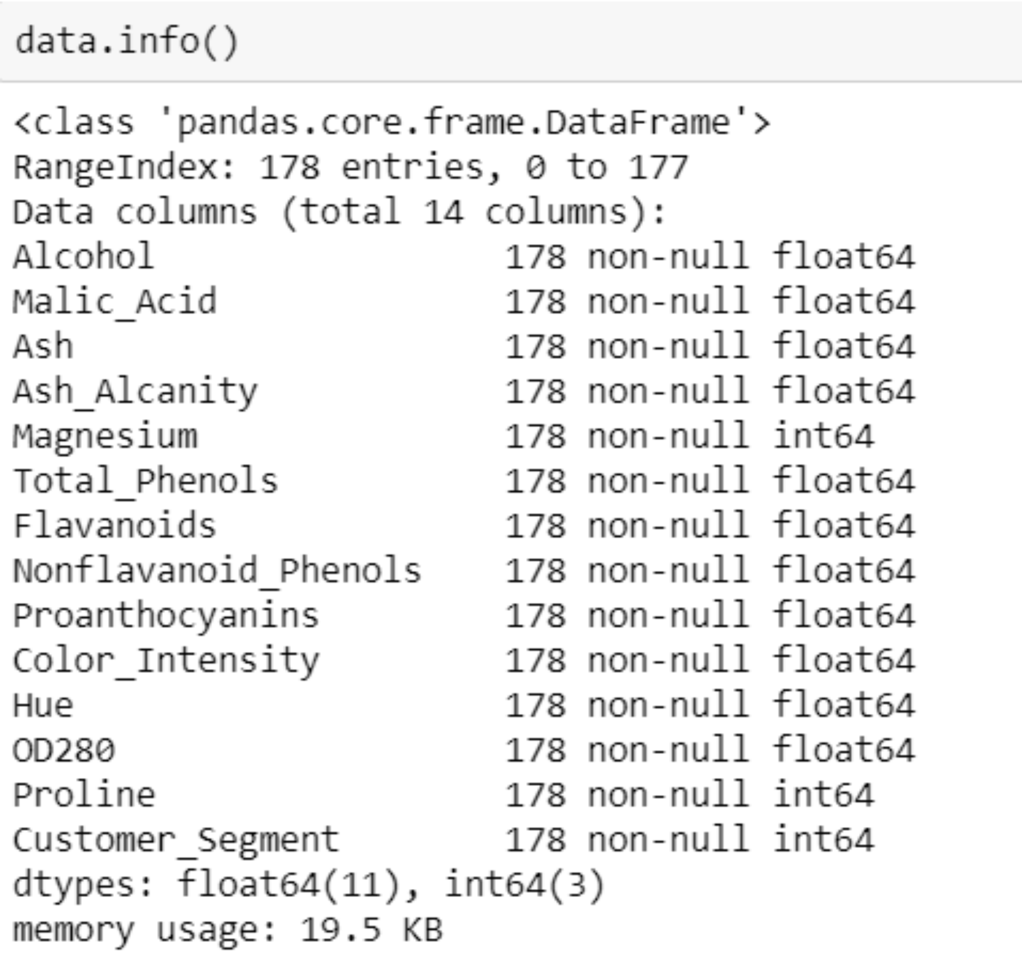
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import pca
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from sklearn.metrics import *
from IPython.display import display
import random
import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import silhouette_score
%matplotlib inline
```

Read the data



```
data=pd.read_csv('Wine.csv')
```

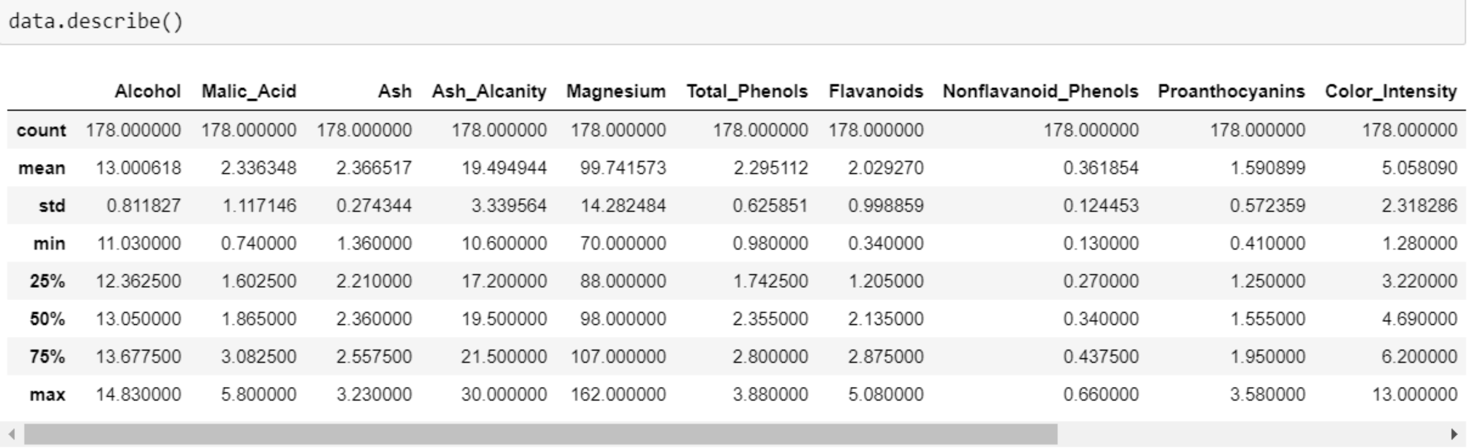
Data information



```
data.info()
```

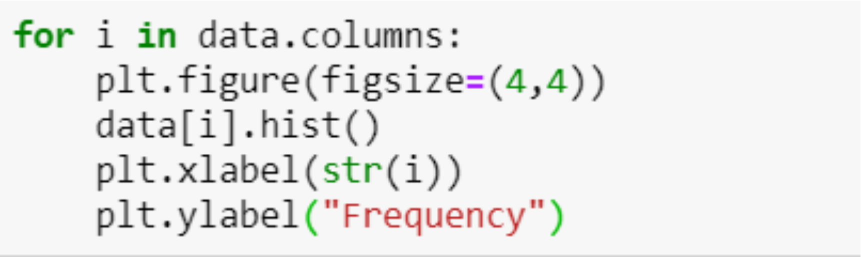
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
Alcohol                 178 non-null float64
Malic_Acid              178 non-null float64
Ash                     178 non-null float64
Ash_Alcanity            178 non-null float64
Magnesium               178 non-null int64
Total_Phenols           178 non-null float64
Flavanoids              178 non-null float64
Nonflavanoid_Phenols    178 non-null float64
Proanthocyanins         178 non-null float64
Color_Intensity         178 non-null float64
Hue                     178 non-null float64
OD280                   178 non-null float64
Proline                 178 non-null int64
Customer_Segment        178 non-null int64
dtypes: float64(11), int64(3)
memory usage: 19.5 KB
```

Data description

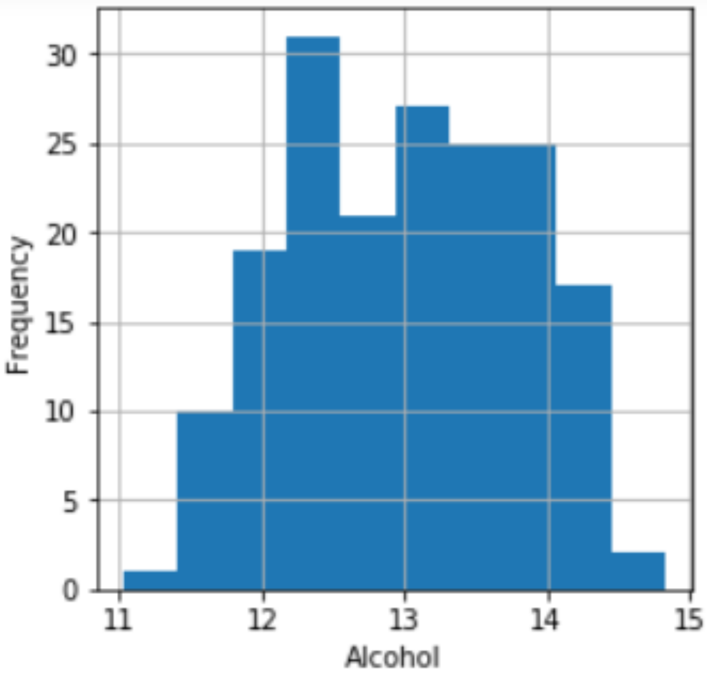


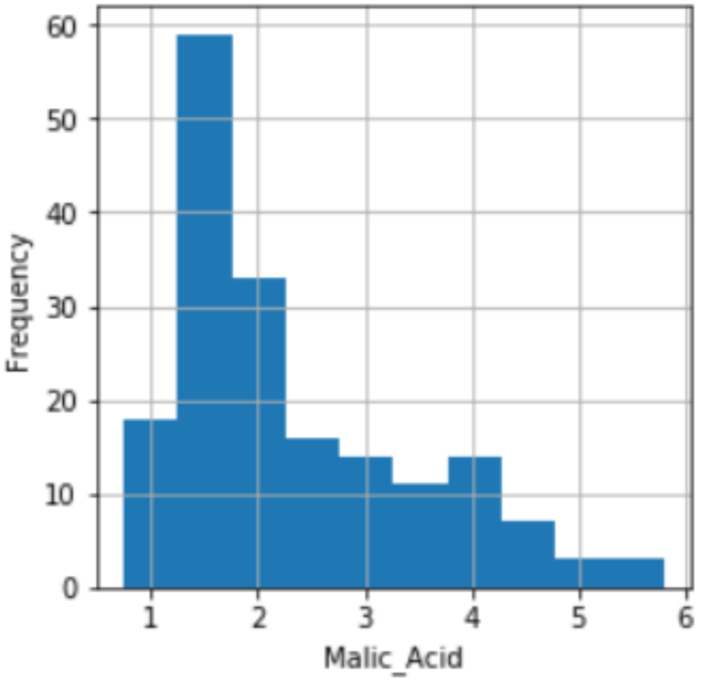
```
data.describe()
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000

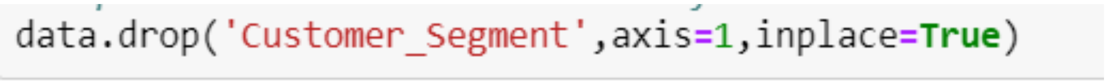


```
for i in data.columns:
    plt.figure(figsize=(4,4))
    data[i].hist()
    plt.xlabel(str(i))
    plt.ylabel("Frequency")
```



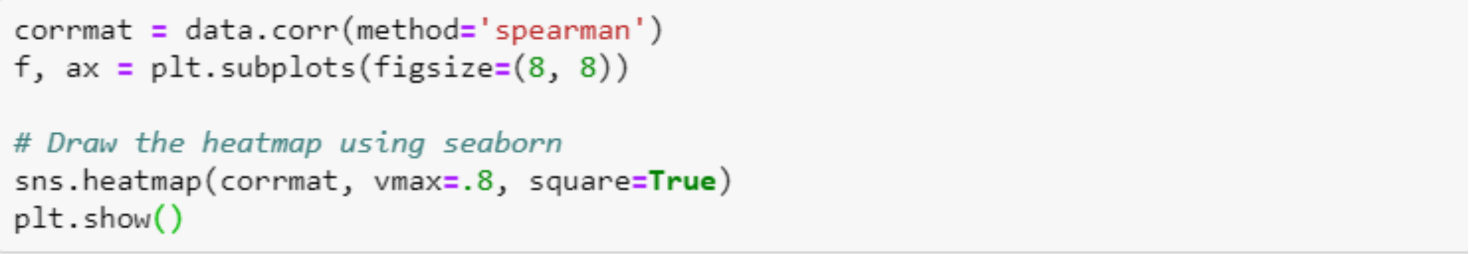


Drop the column which is not useful

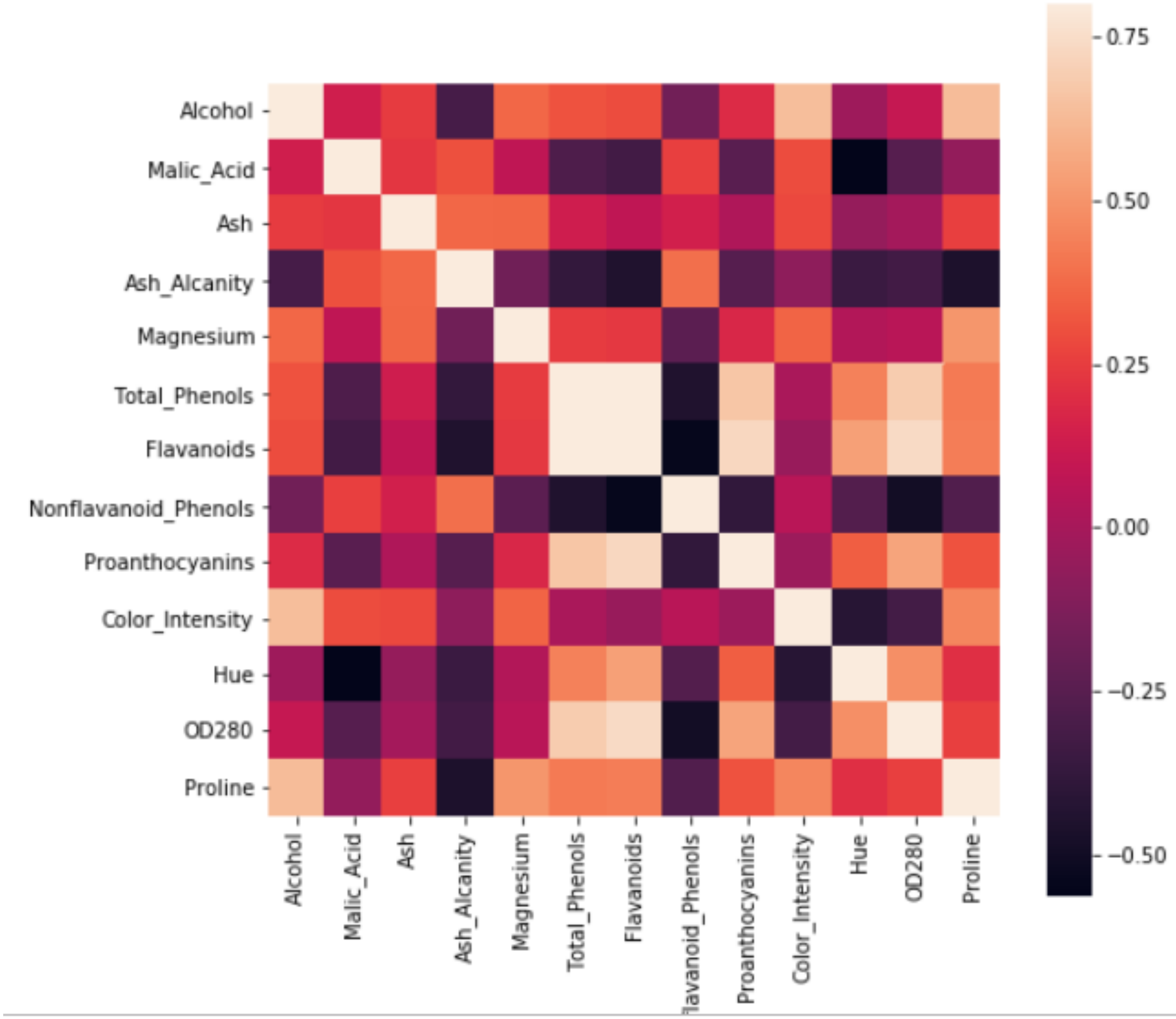


```
data.drop('Customer_Segment',axis=1,inplace=True)
```

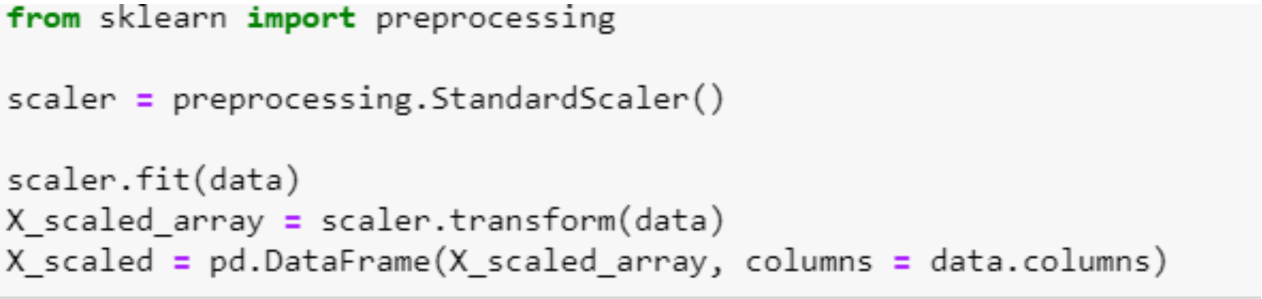
Drawing the heatmap using seaborn.



```
corrmat = data.corr(method='spearman')
f, ax = plt.subplots(figsize=(8, 8))

# Draw the heatmap using seaborn
sns.heatmap(corrmat, vmax=.8, square=True)
plt.show()
```

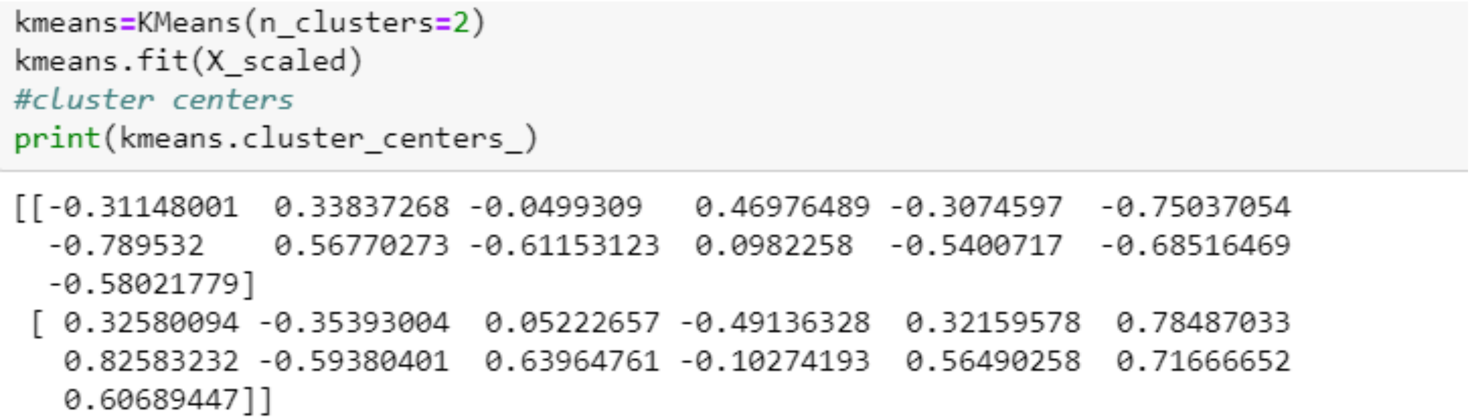


Preprocess the data and apply mean normalizaion for every column



```
from sklearn import preprocessing

scaler = preprocessing.StandardScaler()

scaler.fit(data)
X_scaled_array = scaler.transform(data)
X_scaled = pd.DataFrame(X_scaled_array, columns = data.columns)
```

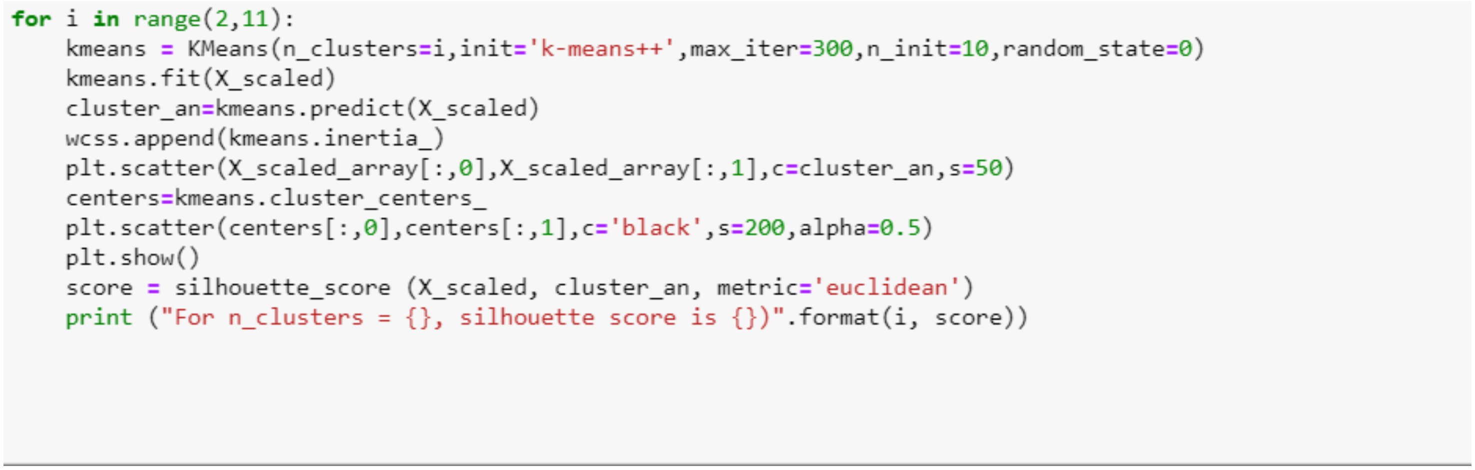
Apply kmeans clustering algorithm.



```
kmeans=KMeans(n_clusters=2)
kmeans.fit(X_scaled)
#cluster centers
print(kmeans.cluster_centers_)
```

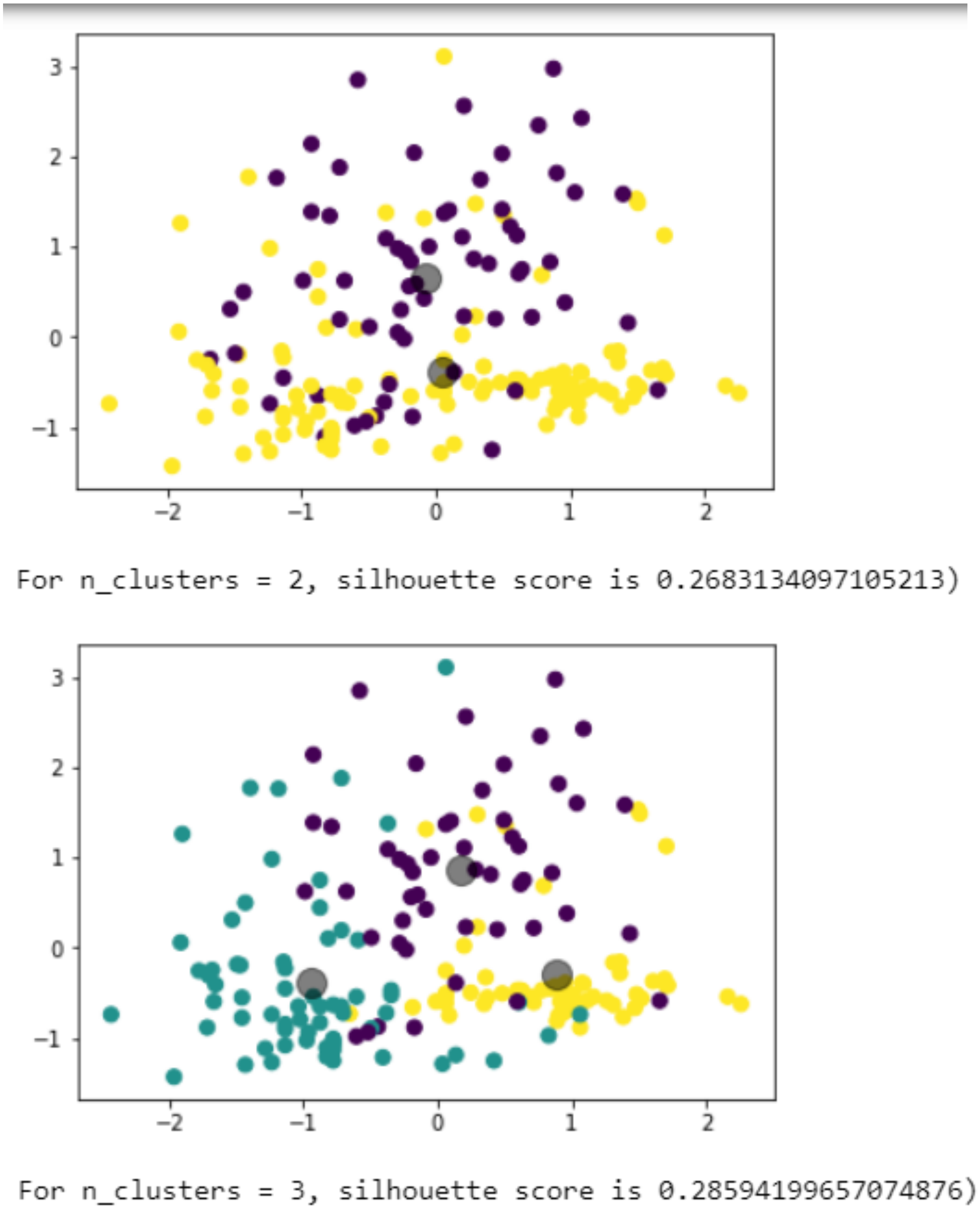
```
[[-0.31148001  0.33837268 -0.0499309   0.46976489 -0.3074597  -0.75037054
  -0.789532    0.56770273 -0.61153123  0.0982258  -0.5400717  -0.68516469
  -0.58021779]
 [ 0.32580094 -0.35393004  0.05222657 -0.49136328  0.32159578  0.78487033
   0.82583232 -0.59380401  0.63964761 -0.10274193  0.56490258  0.71666652
   0.60689447]]
```

Finding the no of clusters and using elbow method to know the number of clusters



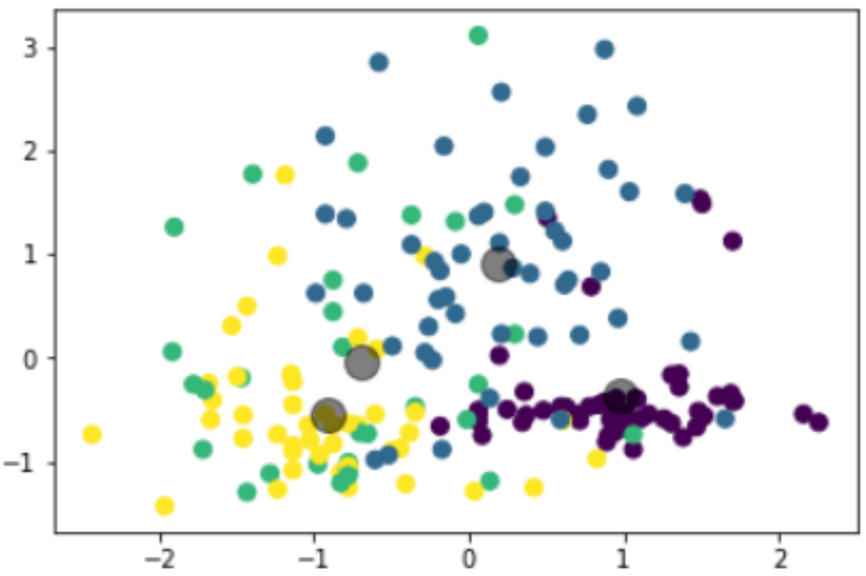
```
for i in range(2,11):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(X_scaled)
    cluster_an=kmeans.predict(X_scaled)
    wcss.append(kmeans.inertia_)
    plt.scatter(X_scaled_array[:,0],X_scaled_array[:,1],c=cluster_an,s=50)
    centers=kmeans.cluster_centers_
    plt.scatter(centers[:,0],centers[:,1],c='black',s=200,alpha=0.5)
    plt.show()
    score = silhouette_score (X_scaled, cluster_an, metric='euclidean')
    print ("For n_clusters = {}, silhouette score is {})".format(i, score))
```

OUTPUTS:



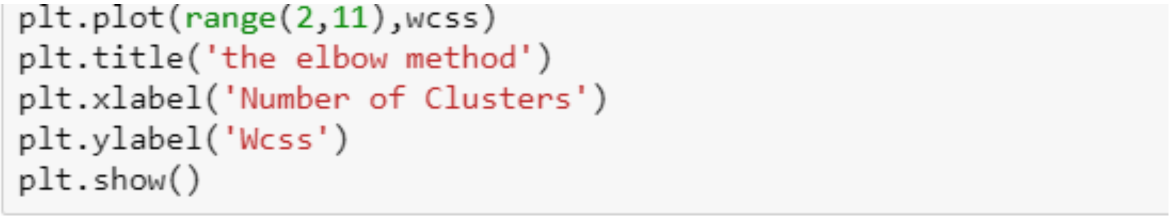
```
For n_clusters = 2, silhouette score is 0.2683134097105213)
```

```
For n_clusters = 3, silhouette score is 0.28594199657074876)
```

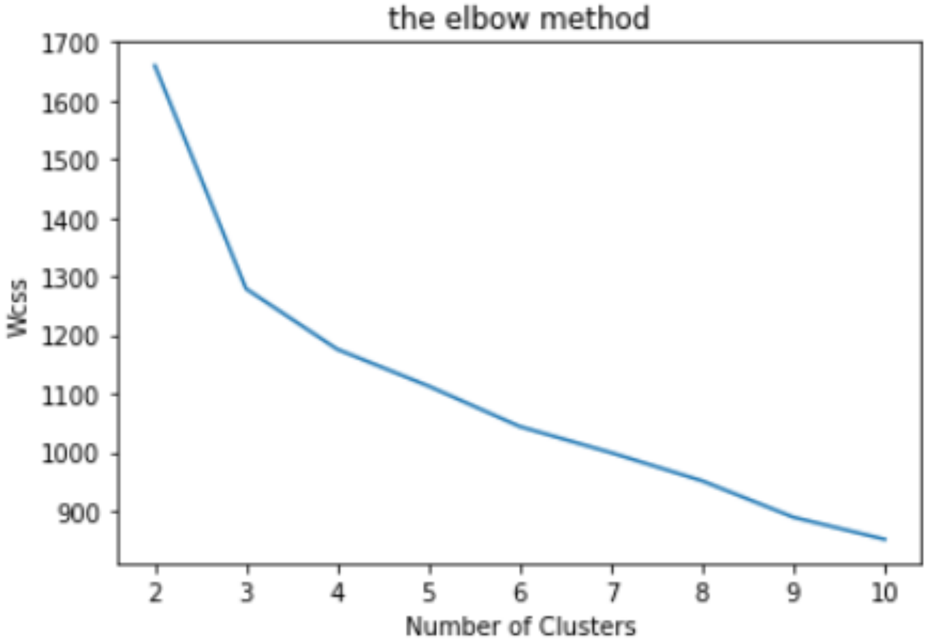


```
For n_clusters = 4, silhouette score is 0.25173343011696475)
```

Ploting the data for elbow method.



```
plt.plot(range(2,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```



Conclusion:

- We have performed various task assigned.
- Good learning experience while working on the assignment.
- We have added few more depth from resources available and was a nice learning curve.
- Used both supervised and un-supervised learning techniques across various tasks.

Edit message

Write a small message here explaining this change. (Optional)

Save Page