*Lab Assignment-1*

*Team Members:*

Aparna Manda: 23

Lohitha Yenugu: 43

*Youtube Link:* Video Link

*Introduction:* This lab assignment focuses on Python programming, implementing object-oriented concepts, and building machine learning models using Scikit-Learn library. In this assignment, we analyzed various datasets using Classification, Regression and Clustering algorithms.

*Objective:* To implement below tasks:

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

*Requirements:*

1. PyCharm IDE

2. Python 3.7
3. Anaconda Interpreter

*Approaches/Methods:*

1. Input is taken dynamically from the user.

2. Numpy and pandas packages are imported for data cleaning.

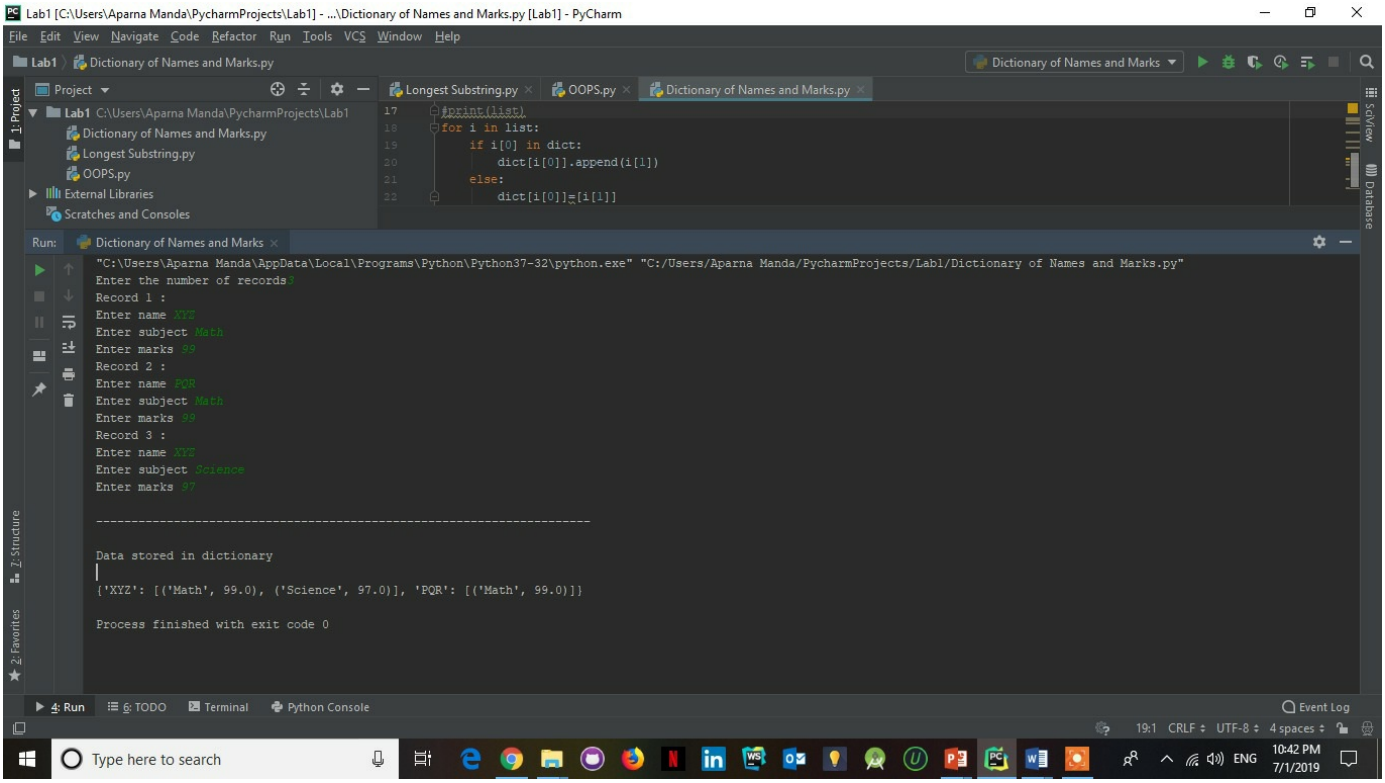
3.Performed EDA for all the machine learning tasks before building the model.

4. Performed RMSE and R2 calculation for evaluating classification algorithms, Silhouette scores for evaluating clustering algorithms.

*WorkFlow:*

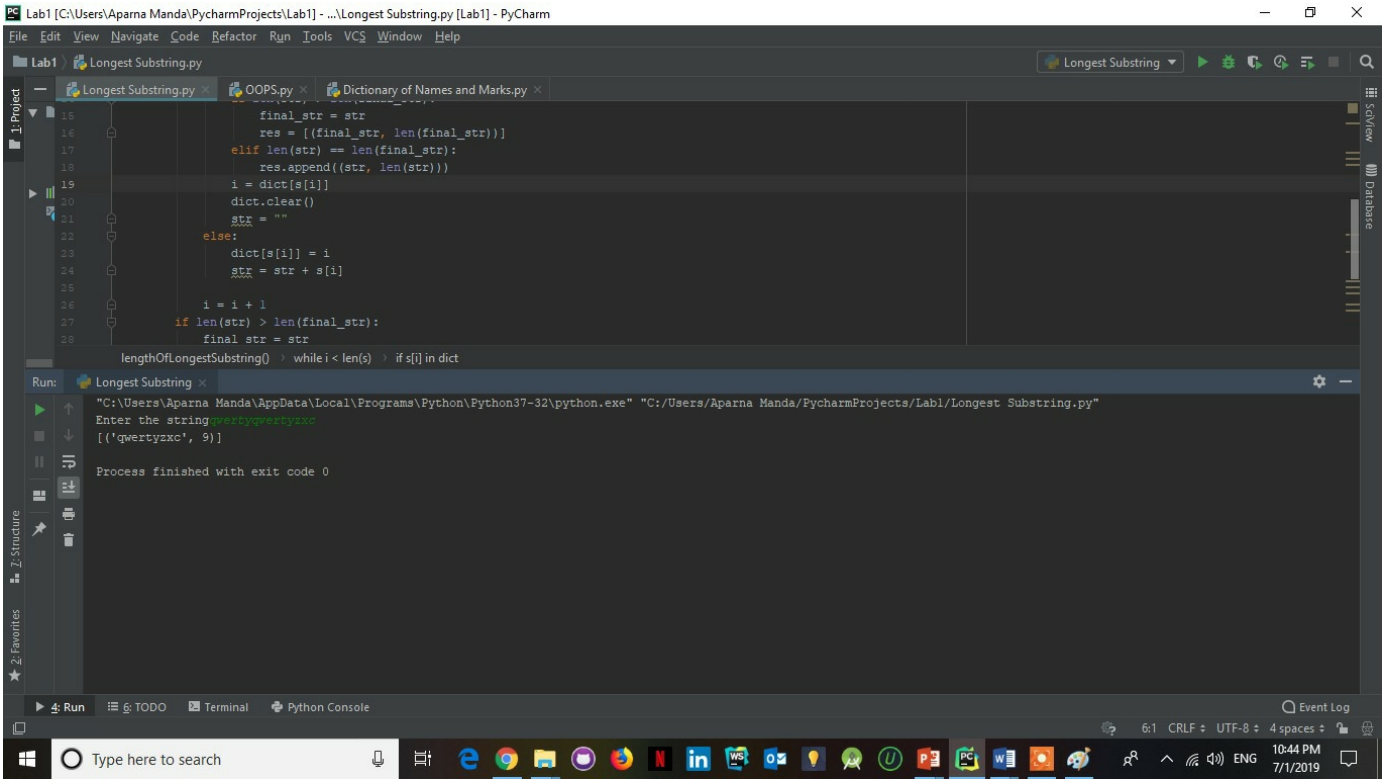
*Task-1:* Create a dictionary with keys as names and values as a list of (subjects, marks)in sorted order with the given data.

*Output:*




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

*Output:*



*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:*

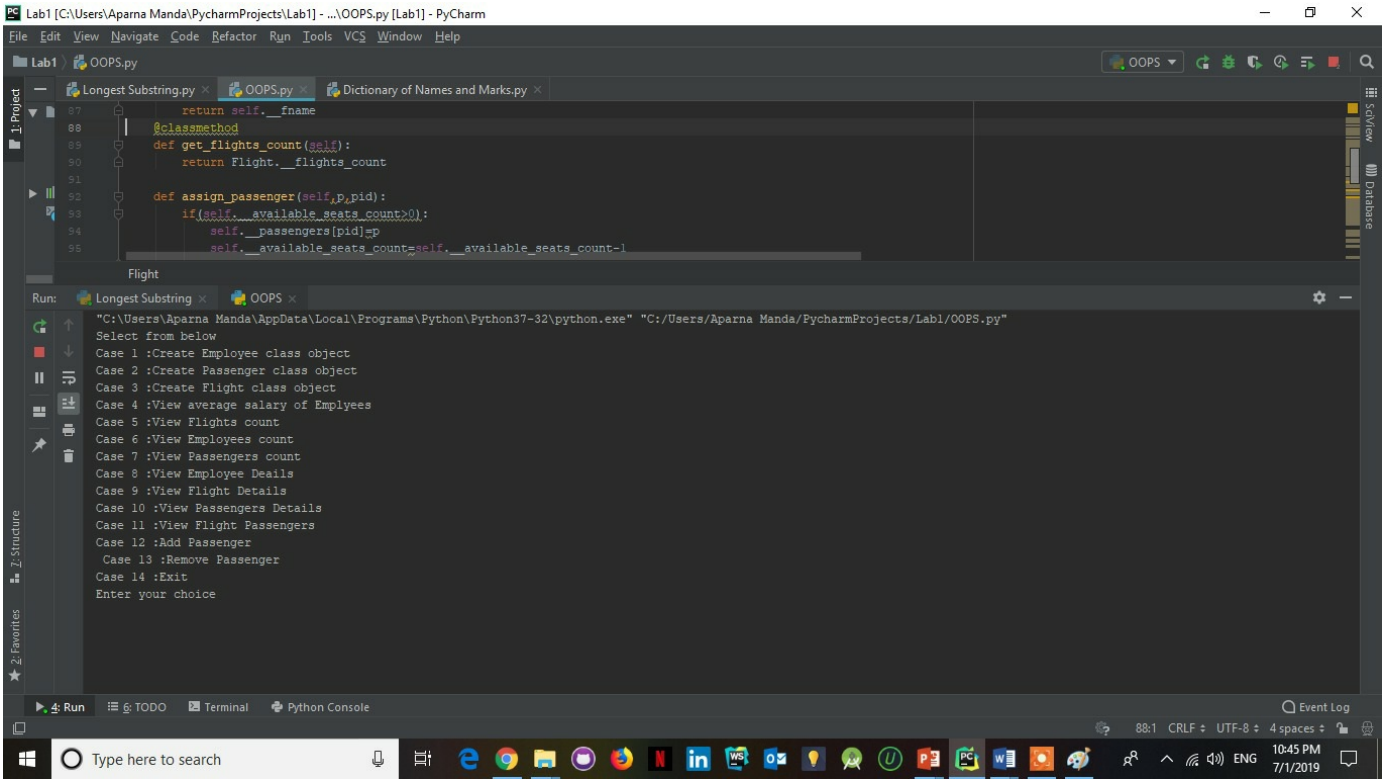
a. Your code should show inheritance at least once.

b. Your code should have one super call

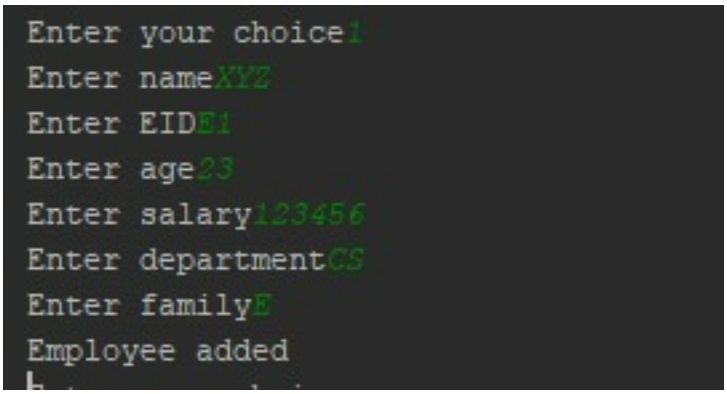
c. Use at least one private data member in your code.

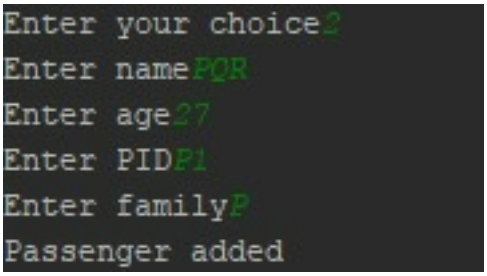
d. Create instances of all classes and show the relationship between them

*Code:*


Lab1 [C:\Users\Aparna Manda\PycharmProjects\Lab1] - ...\OOPS.py [Lab1] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Longest Substring.py
OOPS.py
Dictionary of Names and Marks.py
return self.__fname
@classmethod
def get_flights_count(self):
return Flight.__flights_count
def assign_passenger(self,p,pid):
if(self.__available_seats_count>0):
self.__passengers[pid]=p
self.__available_seats_count=self.__available_seats_count-1
Flight
Run:
Longest Substring
OOPS
"C:\Users\Aparna Manda\AppData\Local\Programs\Python\Python37-32\python.exe" "C:/Users/Aparna Manda/PycharmProjects/Lab1/OOPS.py"
Select from below
Case 1 :Create Employee class object
Case 2 :Create Passenger class object
Case 3 :Create Flight class object
Case 4 :View average salary of Emplyees
Case 5 :View Flights count
Case 6 :View Employees count
Case 7 :View Passengers count
Case 8 :View Employee Deails
Case 9 :View Flight Details
Case 10 :View Passengers Details
Case 11 :View Flight Passengers
Case 12 :Add Passenger
Case 13 :Remove Passenger
Case 14 :Exit
Enter your choice
Run
6: TODO
Terminal
Python Console
Event Log
Type here to search
ENG
10:45 PM
7/1/2019

*Output:*


Enter your choice1
Enter nameXYZ
Enter EIDE1
Enter age23
Enter salary123456
Enter departmentCS
Enter familyE
Employee added


Enter your choice2
Enter namePQR
Enter age27
Enter PIDP1
Enter familyP
Passenger added

```
Enter your choice3
Enter nameFXYZ
Enter fidF1
Enter seats count25
Flight added
```

```
Enter your choice4
Employee Average Salary: 123456.0
```

```
Enter your choice5
Flights Count: 1
Enter your choice6
Employee Count: 1
Enter your choice7
Passenger Count: 1
```

```
Enter your choice8
E1
Select EmployeeE1
Name : XYZ
Salary :123456.0
 Department : CS
 Family : E
 Age : 23
 EID : E1
```

```
Enter your choice9
F1
Select FlightF1
Name : FXYZ
 Seats Count : 25
 Available Seats Count : 25
 FID : F1
```

```
Enter your choice10
P1
Select PassengerP1
Name : PQR
 Age : 27
 Family : P
 PID : P1
```

```
Enter your choice12
F1
Select FlightF1
Selected Flight : FXYZ
P1
Select PassengerP1
Selected passenger : PQR
Passenger added
```

```
Enter your choice11
F1
Select FlightF1
P1
```

```
Enter your choice13
F1
Select FlightF1
Selected Flight : FXYZ
P1
Select PassengerP1
Passenger removed
```

*Task-4:* Create Multiple Regression by choosing a dataset of your choice (again before evaluating, 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.

**Observations:** After applying EDA(removing the null values and

dropping non-correlated features), we saw that R2 value is increased slightly and RMSE valued is decreased.

*Code:*

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

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

train = load_diabetes(return_X_y=False)

data = pd.DataFrame(data= np.c_[train['data'], train['target']],columns= train['feature_names'] + ['target'])

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

corr = numeric_features.corr()
print(corr)
data = data.drop(['s3'], axis=1)
X = data.drop(['target'], axis=1)
Y = data['target']
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))
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')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Linear Regression Model')
plt.show()
```

*Output:*

```
              age       sex       bmi  ...        s5        s6    target
age      1.000000  0.173737  0.185085  ...  0.270777  0.301731  0.187889
sex      0.173737  1.000000  0.088161  ...  0.149918  0.208133  0.043062
bmi      0.185085  0.088161  1.000000  ...  0.446159  0.388680  0.586450
bp       0.335427  0.241013  0.395415  ...  0.393478  0.390429  0.441484
s1       0.260061  0.035277  0.249777  ...  0.515501  0.325717  0.212022
s2       0.219243  0.142637  0.261170  ...  0.318353  0.290600  0.174054
s3      -0.075181 -0.379090 -0.366811  ... -0.398577 -0.273697 -0.394789
s4       0.203841  0.332115  0.413807  ...  0.617857  0.417212  0.430453
s5       0.270777  0.149918  0.446159  ...  1.000000  0.464670  0.565883
s6       0.301731  0.208133  0.388680  ...  0.464670  1.000000  0.382483
target   0.187889  0.043062  0.586450  ...  0.565883  0.382483  1.000000

[11 rows x 11 columns]
R^2 is:
 0.5102003015058936
RMSE is:
 2818.9245631226445
```

*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 a better result.

**Observation:** For the iris dataset, SVM classifier gave a better result.

*Code:*

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")
model = GaussianNB()
X_train = pd.read_csv('./iris.csv')
le = LabelEncoder()
X_train["class"] = le.fit_transform(X_train["class"])
X_train = X_train.select_dtypes(include=[np.number]).interpolate().dropna()
numeric_features = X_train.select_dtypes(include=[np.number])

corr = numeric_features.corr()
print(corr)
X_train = X_train.drop(['sepal width'], axis=1)
train_df, test_df = train_test_split(X_train, test_size=0.4, random_state=0)
X_train = train_df.drop("class", axis=1)
Y_train = train_df["class"]
X_test = test_df.drop("class", axis=1)
Y_test = test_df["class"]

##Naive Bayes
model.fit(X_train, Y_train)
predicted= model.predict(X_test)
acc_nb = round(model.score(X_test, Y_test) * 100, 2)
print("Naive Bayes accuracy is:", acc_nb)

##SVM
svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_test, Y_test) * 100, 2)
print("SVM accuracy is:", acc_svc)

##KNN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_test, Y_test) * 100, 2)
print("KNN accuracy is:",acc_knn)
```

*Output:*

```
                sepal length  sepal width  petal length  petal width     class
sepal length        1.000000    -0.109369      0.871754     0.817954  0.782561
sepal width        -0.109369     1.000000     -0.420516    -0.356544 -0.419446
petal length        0.871754    -0.420516      1.000000     0.962757  0.949043
petal width         0.817954    -0.356544      0.962757     1.000000  0.956464
class               0.782561    -0.419446      0.949043     0.956464  1.000000
Naive Bayes accuracy is: 93.33
SVM accuracy is: 96.67
KNN accuracy is: 93.33

Process finished with exit code 0
```

*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)

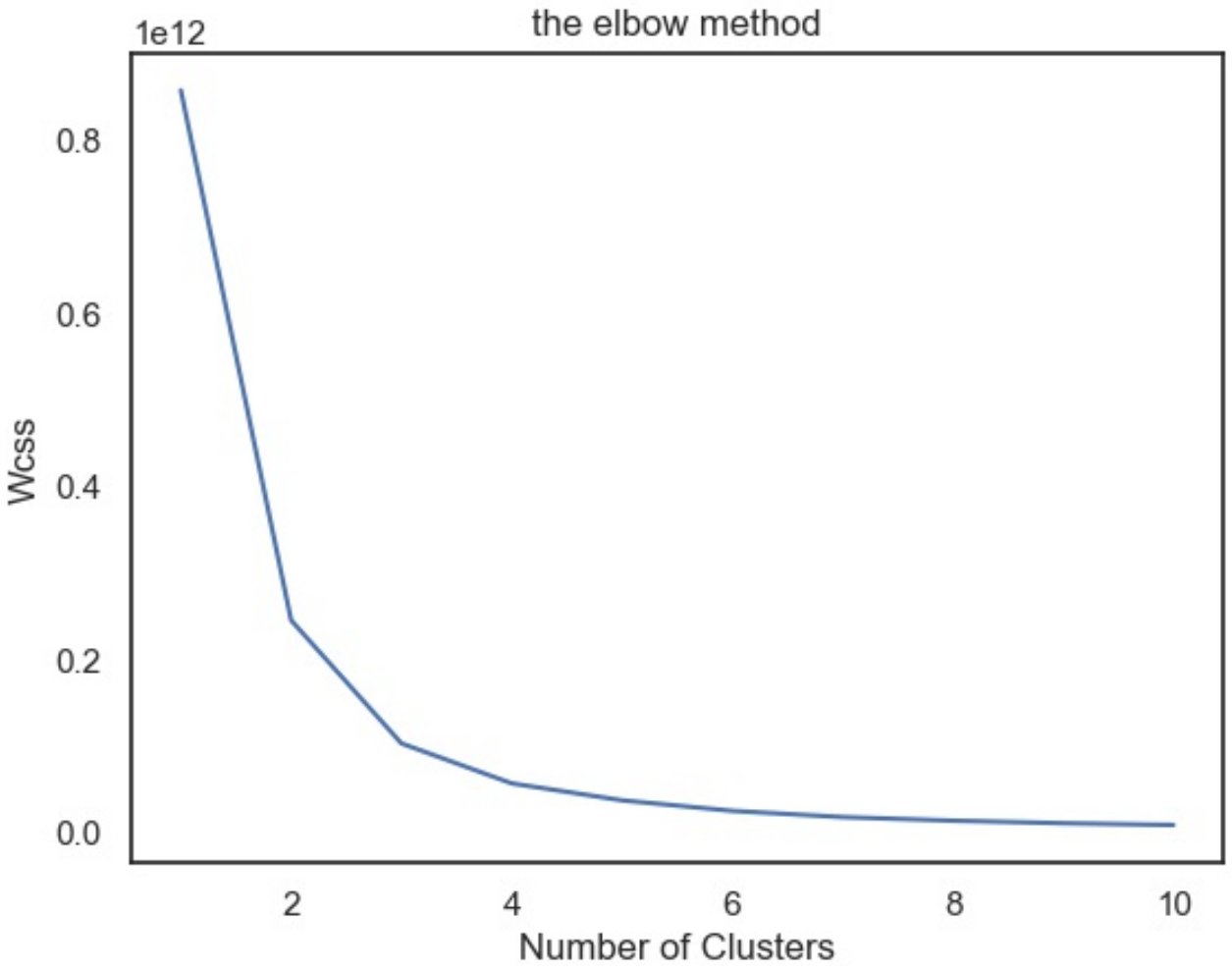
**Observations:** For the diamonds dataset, using the elbow method, the optimal K is taken as 3.

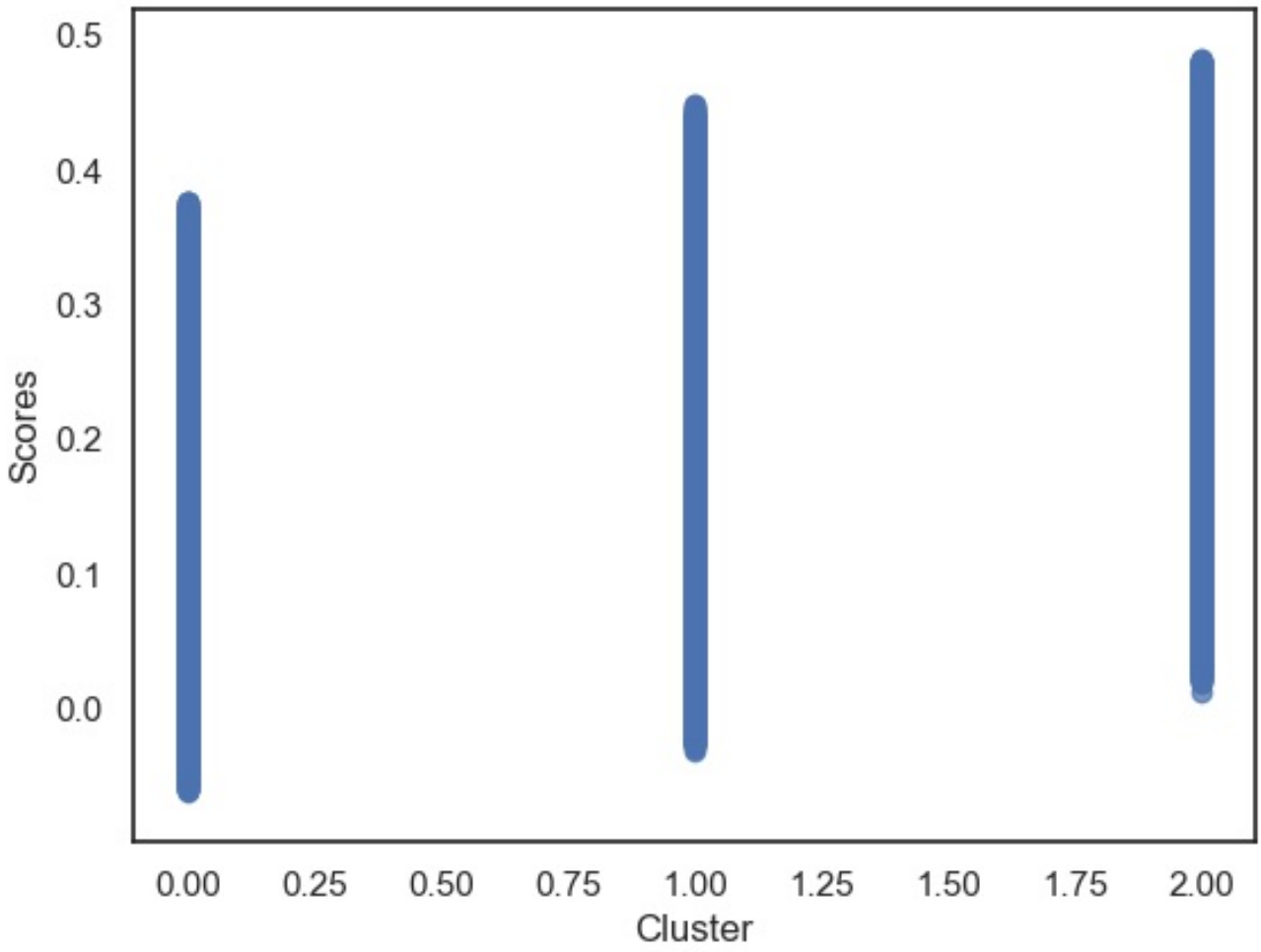
*Code:*

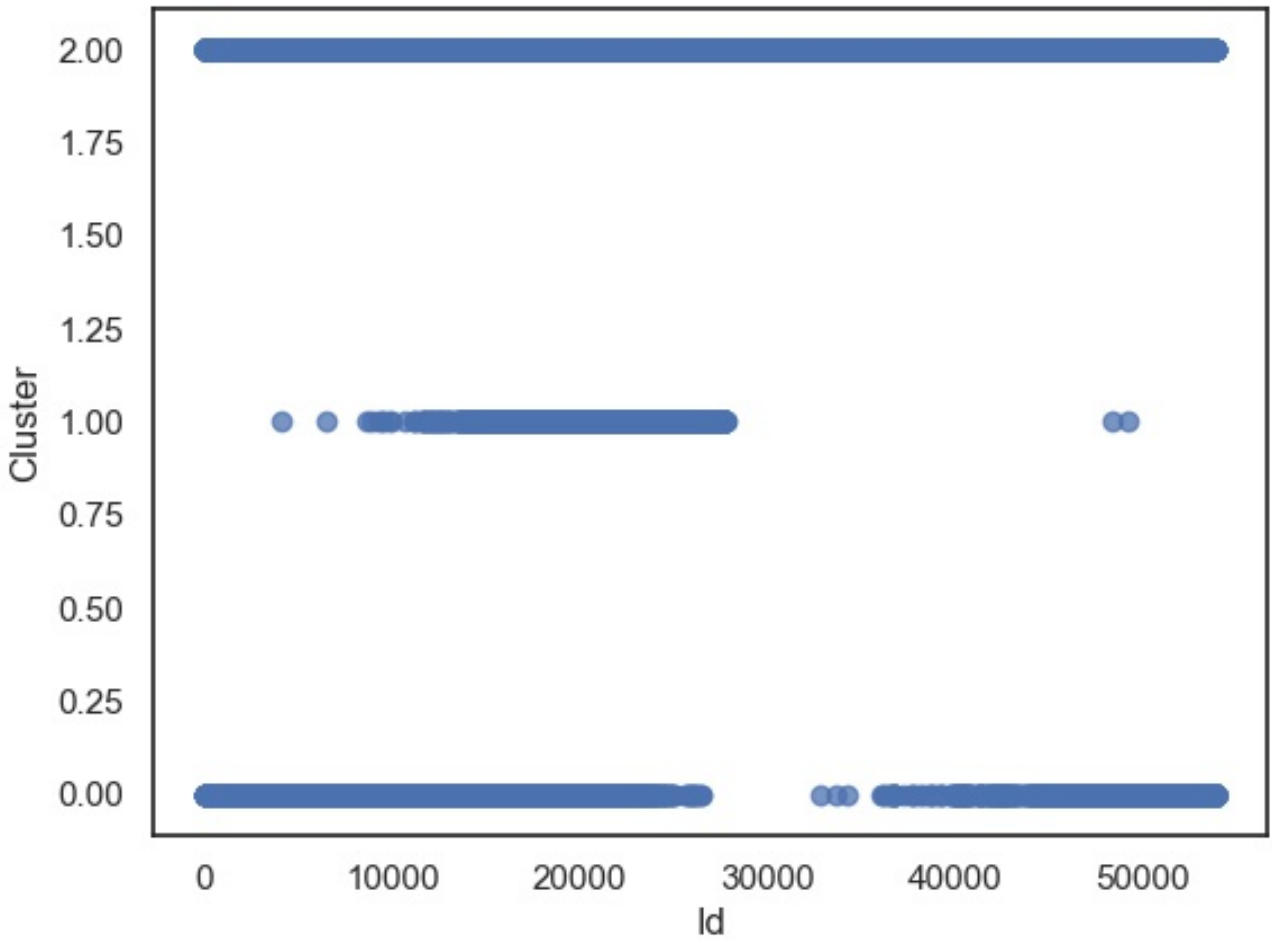
```
train = pd.read_csv('./diamonds.csv')
categorical_data = train.select_dtypes(exclude=[np.number])
categorical_features = list(categorical_data.columns)
le = LabelEncoder()
for i in categorical_features:
    train[i] = le.fit_transform(train[i])
data = train.select_dtypes(include=[np.number]).interpolate().fillna(train.select_dtypes(include=[np.number]).interpolate().mean(axis=0))
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(data)
X_train = scaler.transform(data)

from sklearn.cluster import KMeans
nclusters = 3
seed = 0
km = KMeans(n_clusters=nclusters, random_state=seed)
km.fit(X_train)
y_cluster = km.predict(X_train)

from sklearn import metrics
score = metrics.silhouette_score(X_train, y_cluster)
scores = metrics.silhouette_samples(X_train, y_cluster)
print("Silhoutte score",score)

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(data)
    cluster_an = kmeans.predict(data)
    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()

plt.scatter(y_cluster, scores, alpha=.75,
            color='b')
plt.xlabel('Cluster')
plt.ylabel('Scores')
```

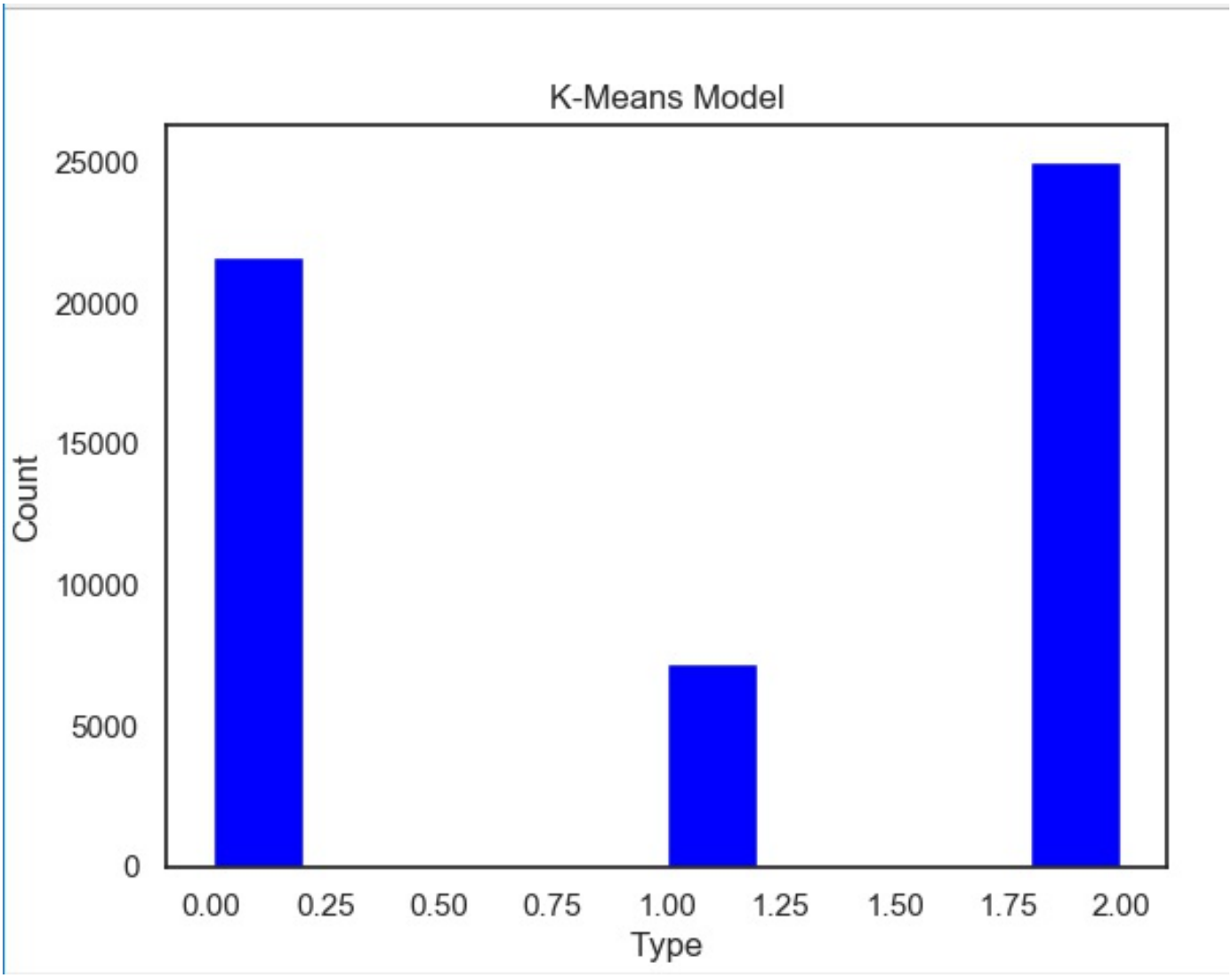
*Output:*

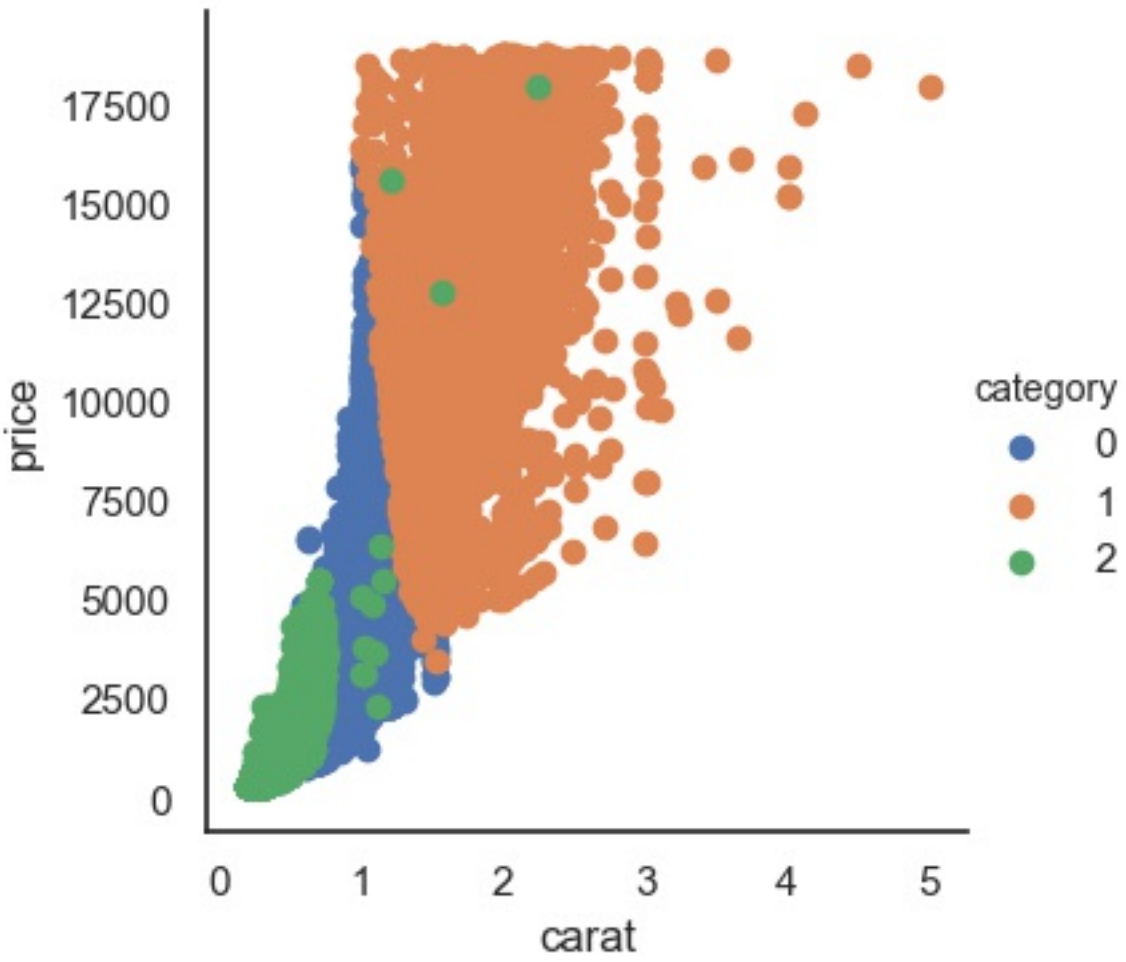
```
Silhoutte score 0.23810135856664913

Process finished with exit code 0
```

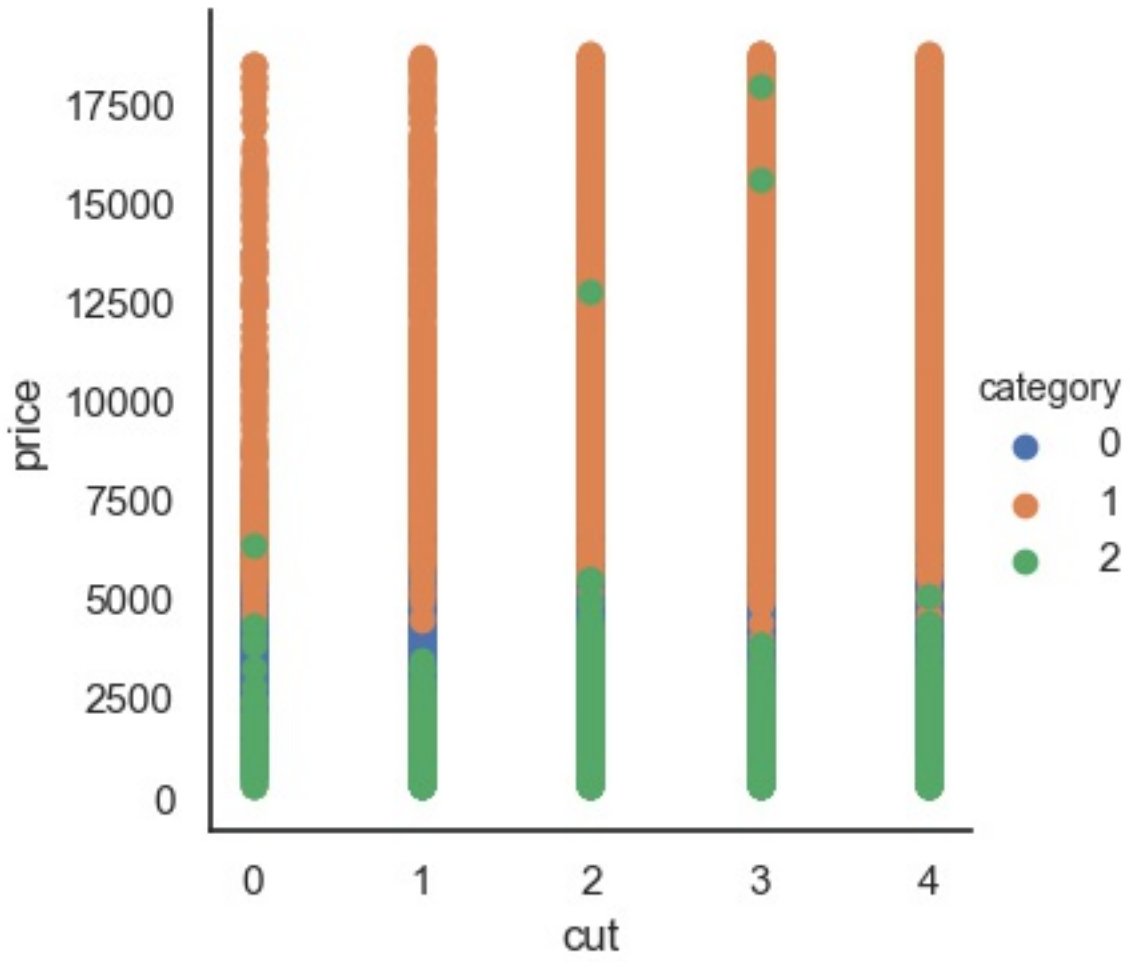



0.5
0.4
0.3
0.2
0.1
0.0
Scores
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Cluster


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cluster
0
10000
20000
30000
40000
50000
Id


K-Means Model
25000
20000
15000
10000
5000
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Type


price
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
carat
category
0
1
2




*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 K-means clustering and plotted them.