Assignment on

CAR PRICE PREDICTION USING MACHINE LEARNING

CA-1

Submitted By:

VISHAL SHARMA
Registration Id: 12101719
Roll No: RQ1E35B51

Submitted To:

DR Niharika Thakur

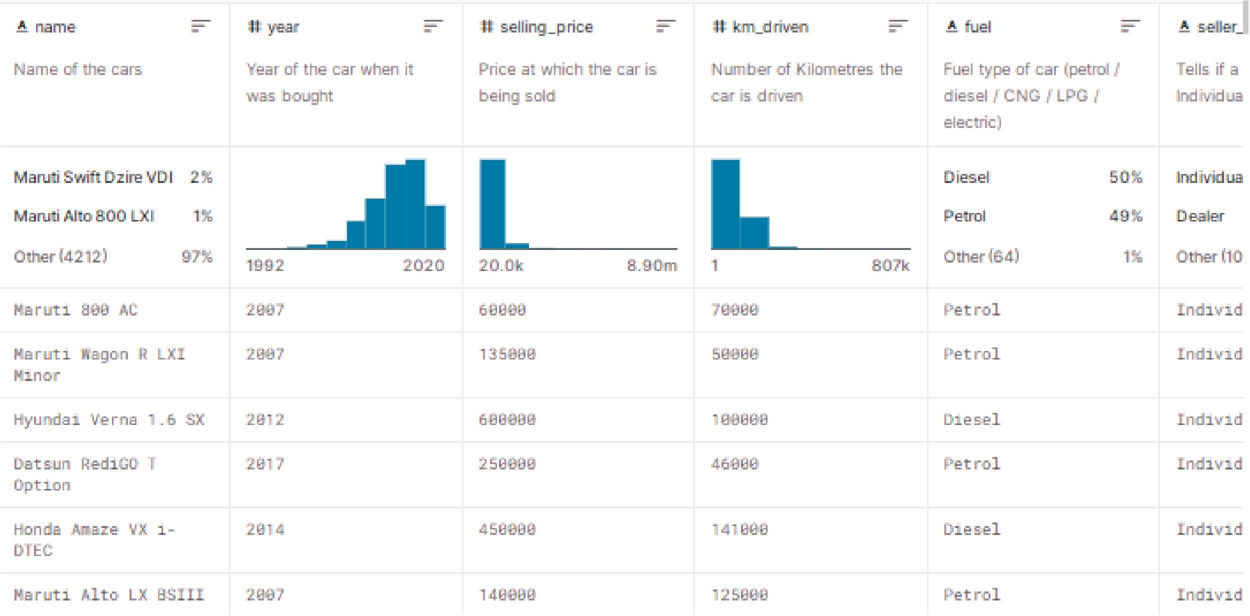
MITTAL SCHOOL OF BUISNESS
LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

INTRODUCTION

Dataset Description:

This dataset contains information about used cars. This data can be used for a lot of purposes such as price prediction to exemplify the use of linear regression in Machine Learning. The columns in the given dataset are as follows:

1. name
2. year
3. selling_price
4. km_driven
5. fuel
6. seller_type
7. transmission
8. Owner

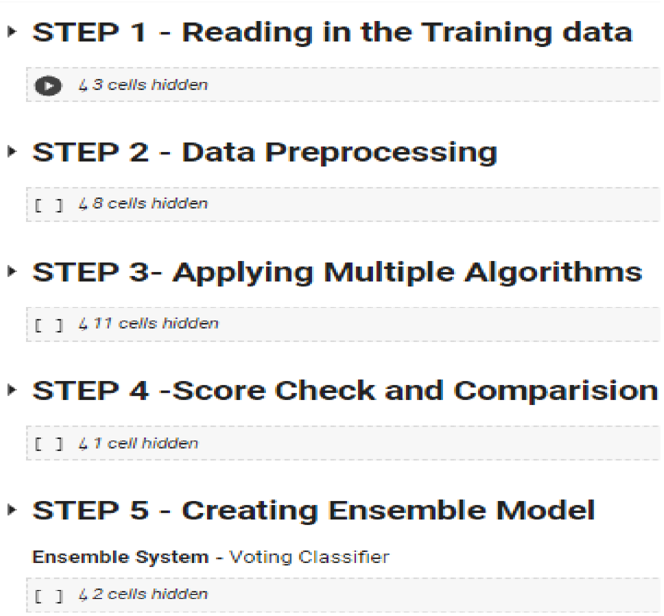


name	year	selling_price	km_driven	fuel	seller_
Name of the cars	Year of the car when it was bought	Price at which the car is being sold	Number of Kilometres the car is driven	Fuel type of car (petrol / diesel / CNG / LPG / electric)	Tells if a Individua
Maruti Swift Dzire VDI 2% Maruti Alto 800 LXI 1% Other (4212) 97%	1992 2020	20.0k 8.90m	1 807k	Diesel 50% Petrol 49% Other (64) 1%	Individua Dealer Other (10
Maruti 800 AC	2007	60000	70000	Petrol	Individ
Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individ
Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individ
Datsun RediGO T Option	2017	250000	46000	Petrol	Individ
Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individ
Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individ

Dataset Source :

https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho

METHODOLOGY



STEP 1 - Reading in the Training data

↳ 3 cells hidden

STEP 2 - Data Preprocessing

[ ] ↳ 8 cells hidden

STEP 3- Applying Multiple Algorithms

[ ] ↳ 11 cells hidden

STEP 4 -Score Check and Comparision

[ ] ↳ 1 cell hidden

STEP 5 - Creating Ensemble Model

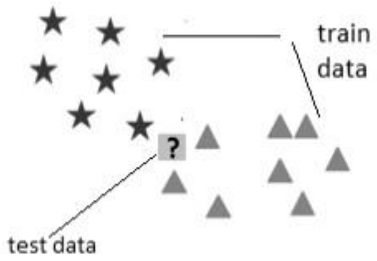
Ensemble System - Voting Classifier

[ ] ↳ 2 cells hidden

DATA PREPROCESSING

In data preprocessing, few targeted columns chosen for removal of NULL and unused values of in dataset for further preventing incorrect train of models after splitting.

Data is separated in 7:3 ratio for training and testing. Where 70% gone for training and 30% for testing. Label Encoding takes place for converting the labels into a numeric form to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how these labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.



red and green points are two classes in training data

ALGORITHM USED

Prediction models are generated using these machine learning algorithms -

- Linear Regression
- k-means clustering
- Decision Tree

GitHub Link

https://github.com/vishalimpinge7696/Data_Modelling

Problem Statement:

The Problem

The prices of new cars in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But due to the increased price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase (Pal, Arora and Palakurthy, 2018). There is a need for a used car price prediction system to effectively determine the worthiness of the car using a variety of features. Even though there are websites that offers this service, their prediction method may not be the best. Besides, different models and systems may contribute on predicting power for a used car’s actual market value. It is important to know their actual market value while both buying and selling.

The Client

To be able to predict used cars market value can help both buyers and sellers.

Used car sellers (dealers): They are one of the biggest target group that can be interested in results of this study. If used car sellers better understand what makes a car desirable, what the important features are for a used car, then they may consider this knowledge and offer a better service.

Online pricing services: There are websites that offers an estimate value of a car. They may have a good prediction model. However, having a second model may help them to give a better prediction to their users. Therefore, the model developed in this study may help online web services that tells a used car’s market value.

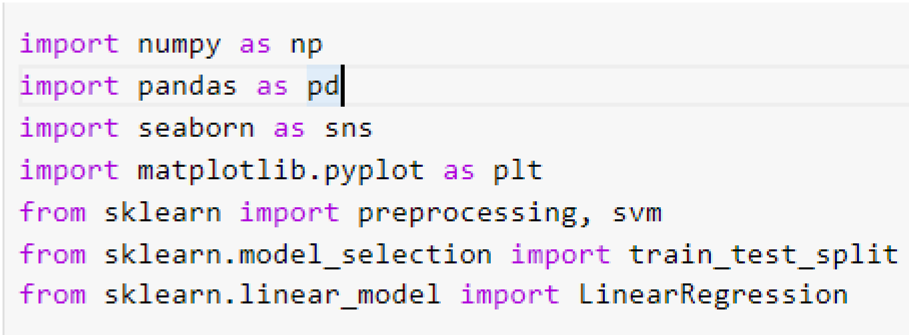
Individuals: There are lots of individuals who are interested in the used car market at some points in their life because they wanted to sell their car or buy a used car. In this process, it’s a big corner to pay too much or sell less then it’s market value.

Data Exploration:

1-LINEAR REGRESSION

Descriptive analysis step by step:

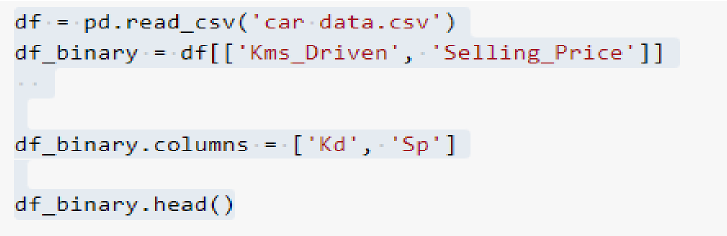
Step 1: Importing all the required libraries



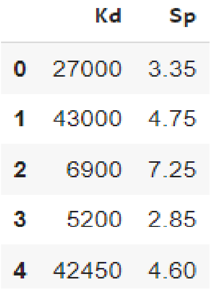
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Step 2: Reading the dataset

In this the data set is uploaded to the google colab from where it is available and then processed with further steps

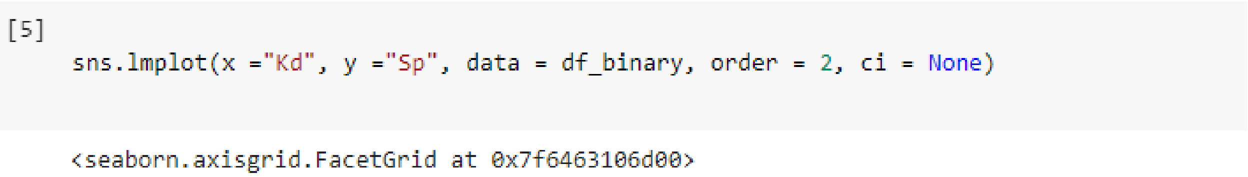


```
df = pd.read_csv('car data.csv')
df_binary = df[['Kms_Driven', 'Selling_Price']]


df_binary.columns = ['Kd', 'Sp']

df_binary.head()
```



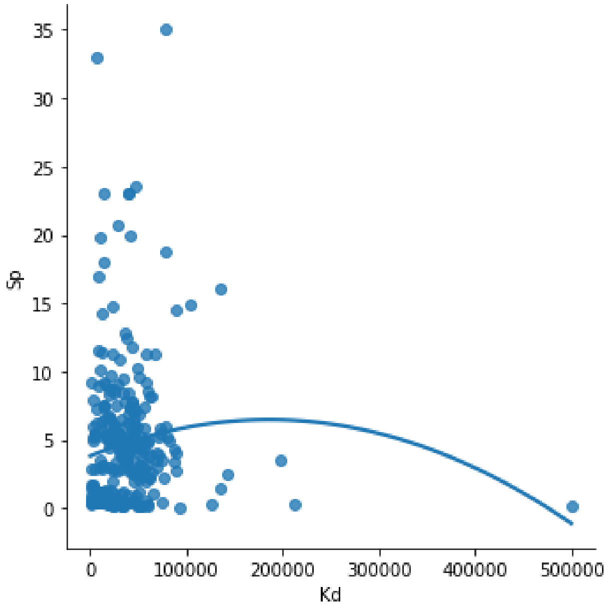
	Kd	Sp
0	27000	3.35
1	43000	4.75
2	6900	7.25
3	5200	2.85
4	42450	4.60

Step 3: Exploring the data scatter by Scatter plot

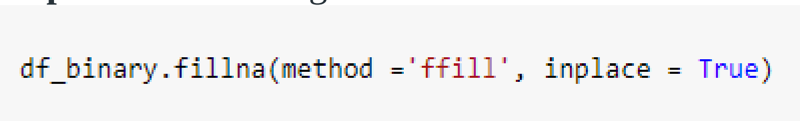


```
[5]
     sns.lmplot(x ="Kd", y ="Sp", data = df_binary, order = 2, ci = None)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f6463106d00>
```

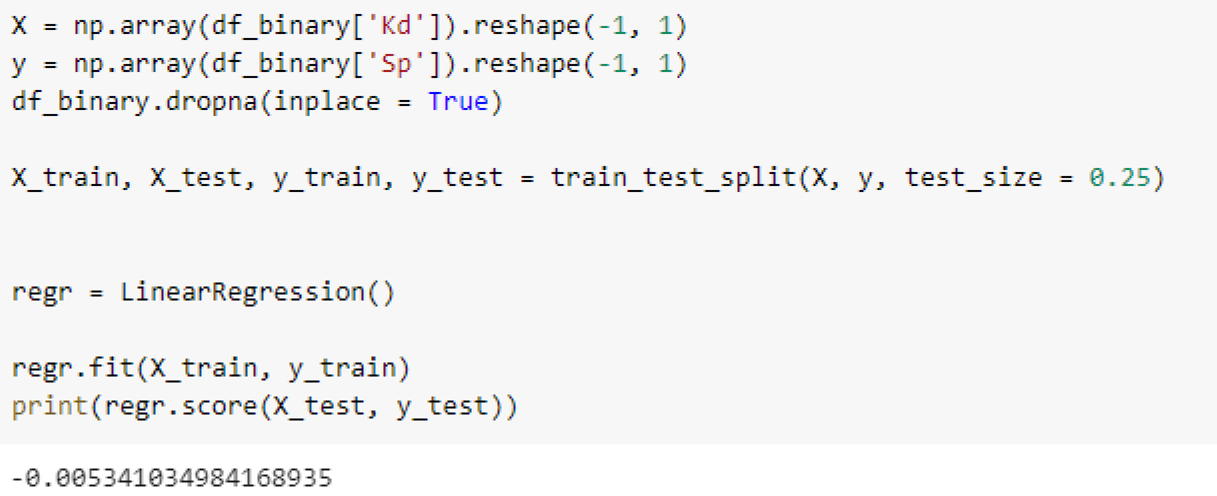


Step 4: Data cleaning



```
df_binary.fillna(method ='ffill', inplace = True)
```

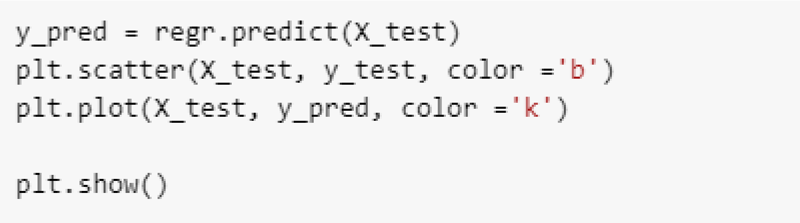
Step 5: Training our model



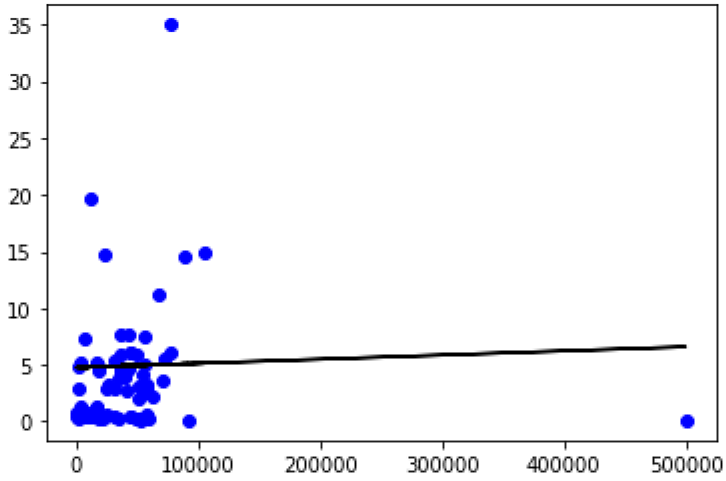
```
X = np.array(df_binary['Kd']).reshape(-1, 1)
y = np.array(df_binary['Sp']).reshape(-1, 1)
df_binary.dropna(inplace = True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)


regr = LinearRegression()

regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
-0.005341034984168935
```

Step 6: Exploring our results

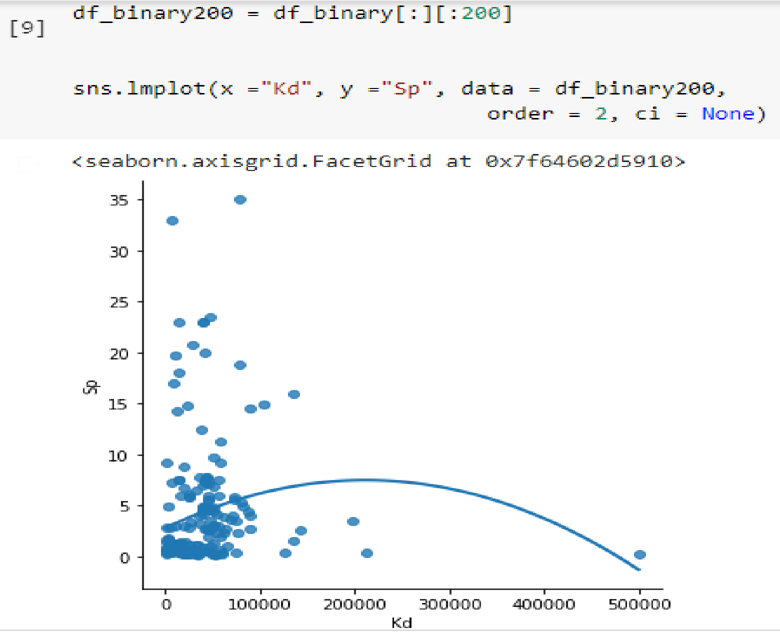


```
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')

plt.show()
```



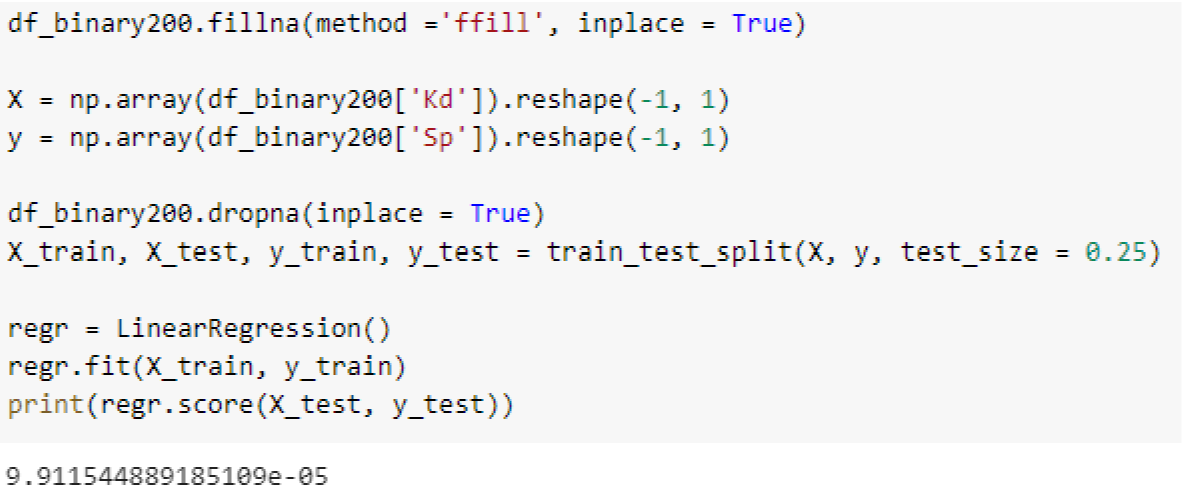
The low accuracy score of our model suggests that our regressive model has not fit very well with the existing data. This suggests that our data is not suitable for linear regression. But sometimes, a dataset may accept a linear regressor if we consider only a part of it. Let us check for that possibility.

Step 7: Working with a smaller dataset



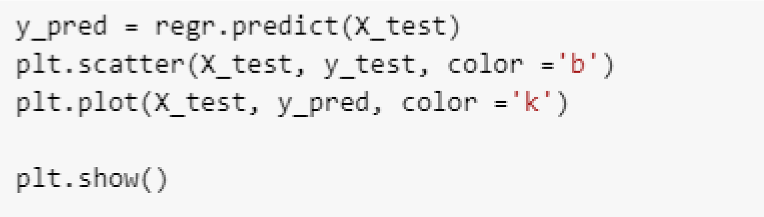
```
[9]  df_binary200 = df_binary[:][:200]

     sns.lmplot(x ="Kd", y ="Sp", data = df_binary200,
                                order = 2, ci = None)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f64602d5910>
```

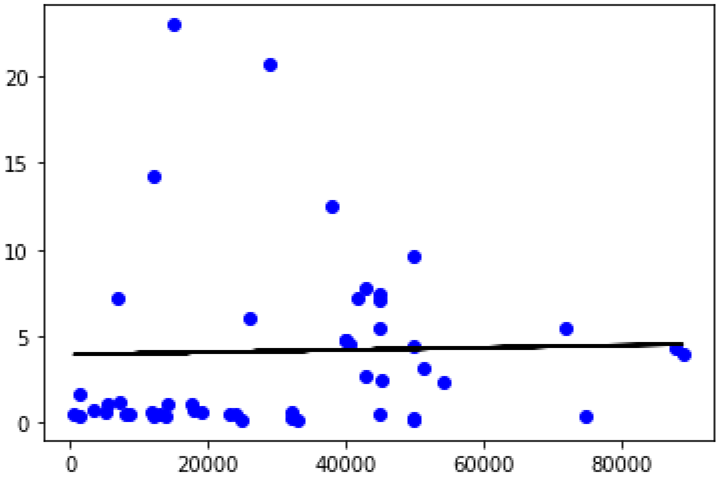


```
df_binary200.fillna(method ='ffill', inplace = True)

X = np.array(df_binary200['Kd']).reshape(-1, 1)
y = np.array(df_binary200['Sp']).reshape(-1, 1)

df_binary200.dropna(inplace = True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
9.911544889185109e-05
```

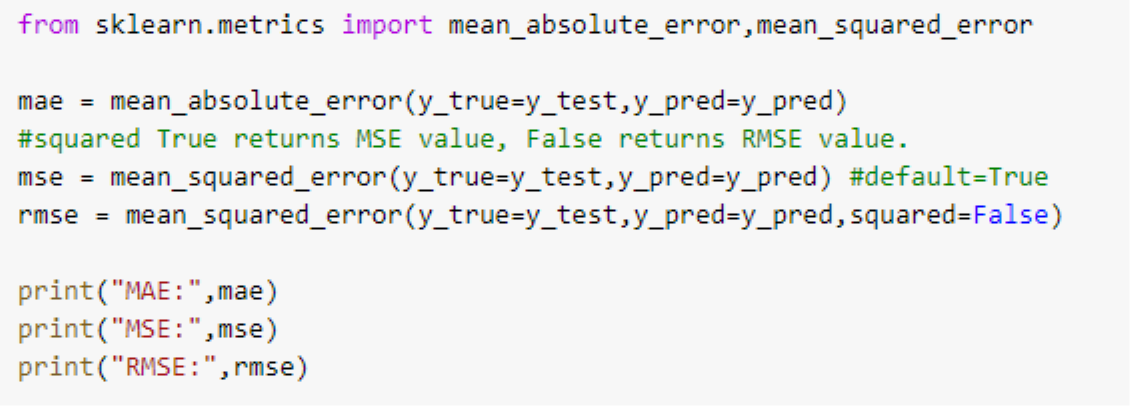


```
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')

plt.show()
```

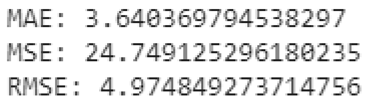


Step 8: Evaluation Metrics For Regression

At last, we check the performance of the Linear Regression model with help of evaluation metrics. For Regression algorithms we widely use mean_absolute_error, and mean_squared_error metrics to check the model performance.



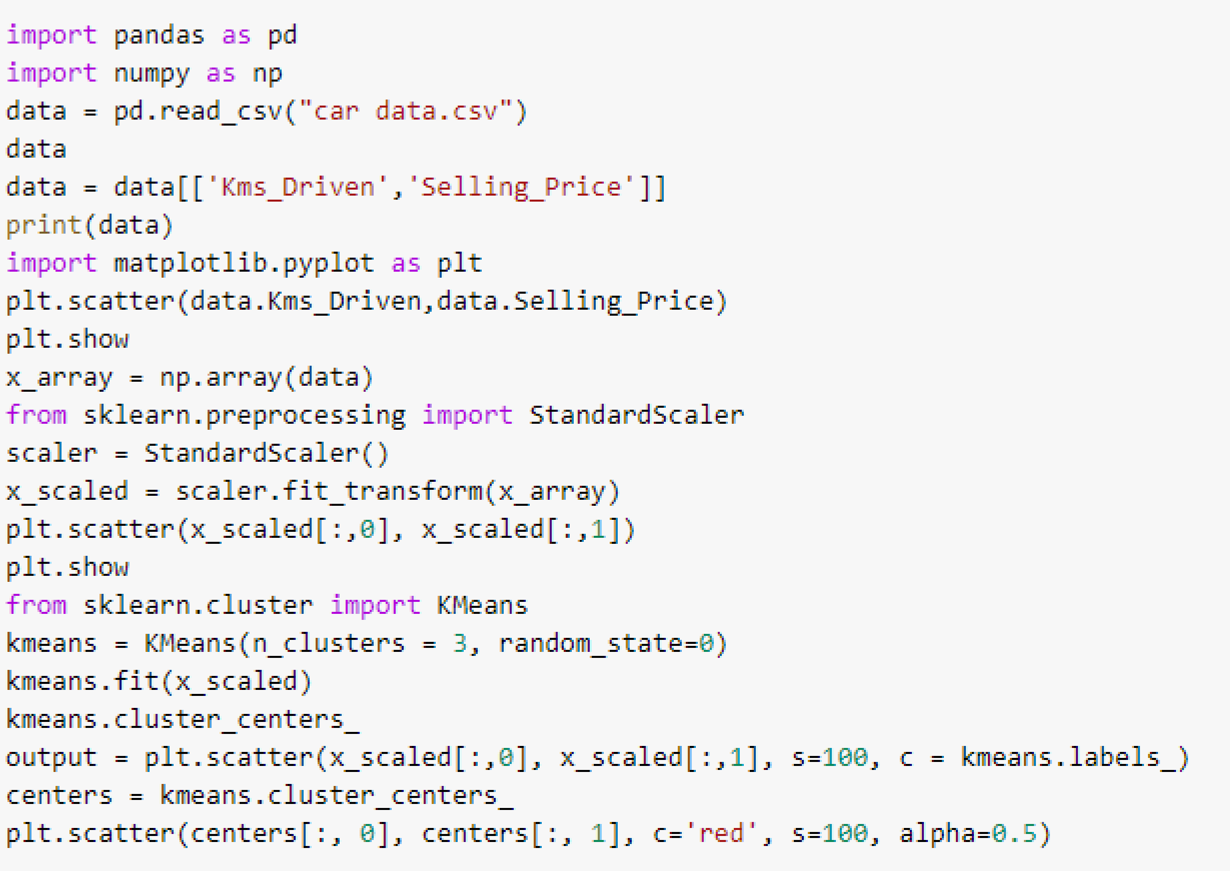
```
from sklearn.metrics import mean_absolute_error,mean_squared_error

mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
#squared True returns MSE value, False returns RMSE value.
mse = mean_squared_error(y_true=y_test,y_pred=y_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,squared=False)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```



```
MAE: 3.640369794538297
MSE: 24.749125296180235
RMSE: 4.974849273714756
```

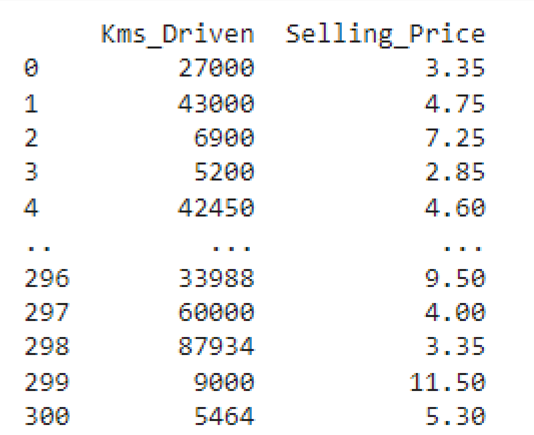
2- K-MEANS CLUSTERING

The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data. This can be used to confirm business assumptions about what types of groups exist or to identify unknown groups in complex data sets.

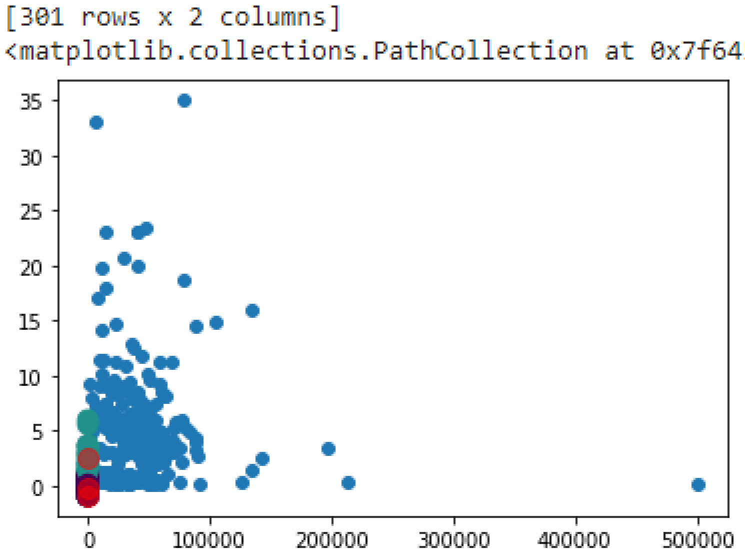


```
import pandas as pd
import numpy as np
data = pd.read_csv("car data.csv")
data
data = data[['Kms_Driven','Selling_Price']]
print(data)
import matplotlib.pyplot as plt
plt.scatter(data.Kms_Driven,data.Selling_Price)
plt.show
x_array = np.array(data)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x_array)
plt.scatter(x_scaled[:,0], x_scaled[:,1])
plt.show
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3, random_state=0)
kmeans.fit(x_scaled)
kmeans.cluster_centers_
output = plt.scatter(x_scaled[:,0], x_scaled[:,1], s=100, c = kmeans.labels_)
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=100, alpha=0.5)
```

OUTPUT



```
     Kms_Driven  Selling_Price
0         27000           3.35
1         43000           4.75
2          6900           7.25
3          5200           2.85
4         42450           4.60
..          ...            ...
296       33988           9.50
297       60000           4.00
298       87934           3.35
299        9000          11.50
300        5464           5.30
```

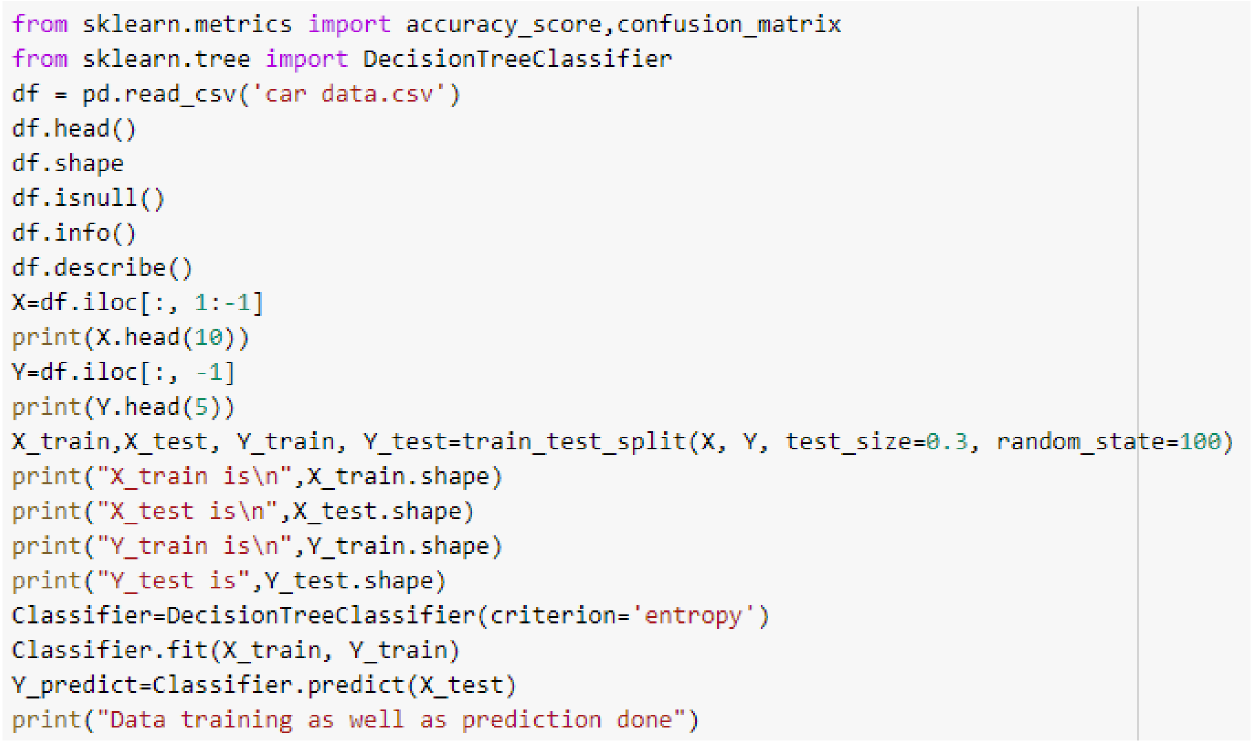


```
[301 rows x 2 columns]
<matplotlib.collections.PathCollection at 0x7f645d8c5790>
```

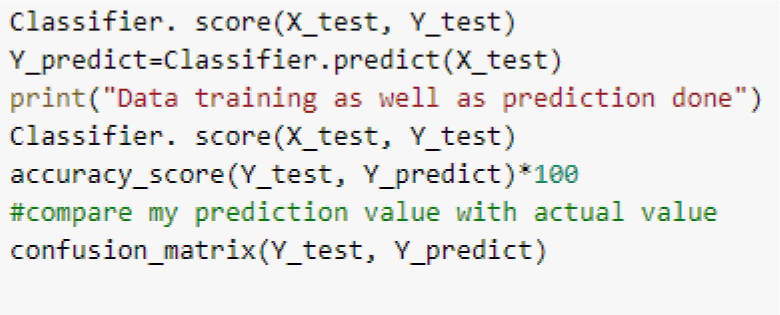
3- DECISION TREE

Decision trees are used as an approach in machine learning **to structure the algorithm**. A decision tree algorithm will be used to split dataset features through a cost function. The decision tree is grown before being optimised to remove branches that may use irrelevant features, a process called pruning.

CODE FOR DECISION TREE

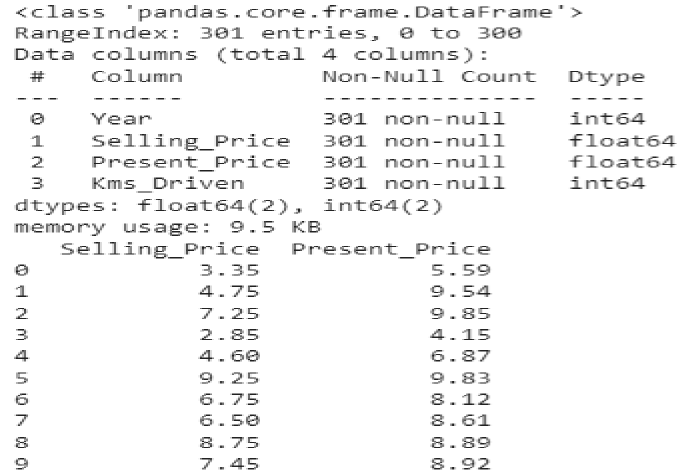


```
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.tree import DecisionTreeClassifier
df = pd.read_csv('car data.csv')
df.head()
df.shape
df.isnull()
df.info()
df.describe()
X=df.iloc[:, 1:-1]
print(X.head(10))
Y=df.iloc[:, -1]
print(Y.head(5))
X_train,X_test, Y_train, Y_test=train_test_split(X, Y, test_size=0.3, random_state=100)
print("X_train is\n",X_train.shape)
print("X_test is\n",X_test.shape)
print("Y_train is\n",Y_train.shape)
print("Y_test is",Y_test.shape)
Classifier=DecisionTreeClassifier(criterion='entropy')
Classifier.fit(X_train, Y_train)
Y_predict=Classifier.predict(X_test)
print("Data training as well as prediction done")
```

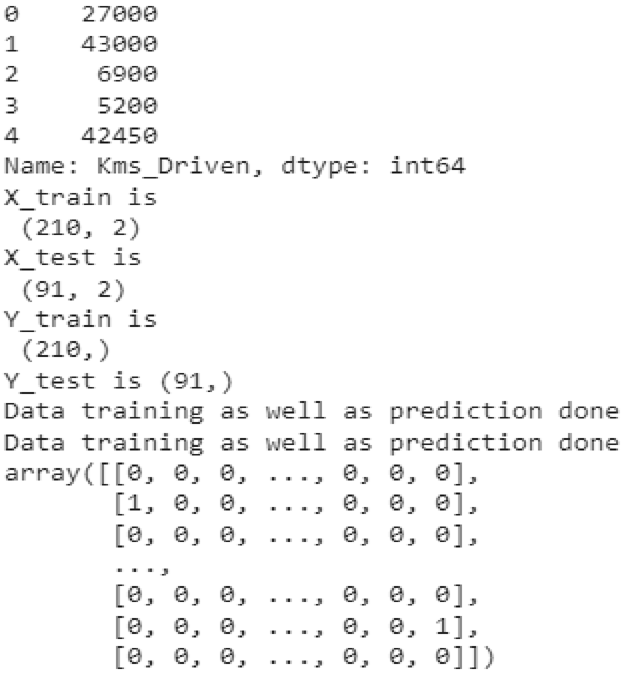


```
Classifier. score(X_test, Y_test)
Y_predict=Classifier.predict(X_test)
print("Data training as well as prediction done")
Classifier. score(X_test, Y_test)
accuracy_score(Y_test, Y_predict)*100
#compare my prediction value with actual value
confusion_matrix(Y_test, Y_predict)
```

OUTPUT



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Year           301 non-null    int64
 1   Selling_Price  301 non-null    float64
 2   Present_Price  301 non-null    float64
 3   Kms_Driven     301 non-null    int64
dtypes: float64(2), int64(2)
memory usage: 9.5 KB
   Selling_Price  Present_Price
0           3.35           5.59
1           4.75           9.54
2           7.25           9.85
3           2.85           4.15
4           4.60           6.87
5           9.25           9.83
6           6.75           8.12
7           6.50           8.61
8           8.75           8.89
9           7.45           8.92
```



```
0    27000
1    43000
2     6900
3     5200
4    42450
Name: Kms_Driven, dtype: int64
X_train is
 (210, 2)
X_test is
 (91, 2)
Y_train is
 (210,)
Y_test is (91,)
Data training as well as prediction done
Data training as well as prediction done
array([[0, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 0]])
```

Conclusion

By performing different models, it was aimed to get different perspectives and eventually compared their performance. With this study, it purpose was to predict prices of used cars by using a dataset that has 13 predictors and 380962 observations. With the help of the data visualizations and exploratory data analysis, the dataset was uncovered and features were explored deeply. The relation between features were examined. At the last stage, predictive models were applied to predict price of cars in an order: random forest, linear regression, ridge regression, lasso, KNN, XGBoost.

By considering all four metrics from table 15, it can be concluded that random forest the best model for the prediction for used car prices. Random Forest as a regression model gave the best MAE, MSE and RMSE values (Table 14). According to random forest, here are the most important features: year, odometer, make, drive, fuel, manufacturer, cylinders. These features provide 3960.11 RMSE just by using seven listed features.

Limitations of the Study and Suggestion for Further Studies

This study used different models in order to predict used car prices. However, there was a relatively small dataset for making a strong inference because number of observations was only 380962. Gathering more data can yield more robust predictions. Secondly, there could be more features that can be good predictors. For example, here are some variables that might improve the model: number of doors, gas/mile (per gallon), color, mechanical and cosmetic reconditioning time, used-to-new ratio, appraisal-to-trade ratio.

Another point that that has room to improvement is that data cleaning process can be dome more rigorously with the help of more technical information. For example, instead of using ‘ffill’ method, there might be indicators that helps to fill missing values more meaningfully.