FLIP ROBO

<u>Car Price Prediction</u>

Submitted By-
Akanksha Amarnani

Acknowledgement

I would like to thank Almighty for giving me the confidence to pursue this project. Further, the concepts from DataTrained Academy guided me to complete the project.
In addition I would like to thank my mentor from Flip Robo Technology, Ms Khushboo Garg for clarifying my doubts and queries.

The references used for the completion of this project are-

- Car Price Prediction using Machine Learning Techniques; International Burch University, Sarajevo, Bosnia and Herzegovina, Enis Gegic, Becir Isakovic, Dino Keco, Zerina Masetic and Jasmin Kevric, TEM Journal. Volume 8, Issue 1, Pages 113-118, ISSN 2217-8309, DOI: 10.18421/TEM81-16, February 2019.
- CS 229 Project Report: Predicting Used Car Prices; Kshitij Kumbar, Pranav Gadre and Varun Nayak
- Predicting Car Price using Machine Learning, Tarique Akhtar, Nov 23, 2020, Towards Data Science

INTRODUCTION

Business Problem Framing-

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a buyer, it is also a dilemma to price a used car appropriately. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model.

Conceptual Background of the Domain Problem-

The domain related concepts which help us in a better understanding are-

- Web Scraping- Web scraping data of used cars from different websites
- Exploratory Data Analysis (EDA)- By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.
- Splitting the data- The dataset is split into train and test sample using the train test split
- Modeling- We apply Linear Regression, Random Forest Regressor and Ada Boost Regressor models for prediction of the prices for the house
- Regularization- Models are regularized and the parameters are hypertuned to enhance the efficiency of the models

Review of Literature-

Machine learning is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to extract useful information from data. Machine learning methods are appropriate in big data since attempting to manually process vast volumes of data would be impossible without the support of machines. Machine learning in computer science attempts to solve problems algorithmically rather than purely mathematically. Therefore, it is based on creating algorithms that permit the machine to learn. However, there are two general groups in machine learning which are supervised and unsupervised. Supervised is where the program gets trained on pre-determined set to be able to predict when a new data is given. Unsupervised is where the program tries to find the relationship and the hidden pattern between the data

The performance of the model build will be measured upon predicting house prices since the prediction in many regression algorithms relies not only on a specific feature but on an unknown number of attributes that result in the value to be predicted. Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in India. The data used in the experiment will be handled by using a combination of pre-processing methods to improve the prediction accuracy

Motivation for the Problem Undertaken-

The purchase and sale of used cars is considered an unpredictable, cumbersome process. Being a data analyst, it seems to be my responsibility to add the element of prediction in every unpredictable scenario, to solve the cumbersome unsure process into a more reliable, dependent matter. Therefore, the project motivated me to go further and predict the unpredictable. Further, every project has a lot to offer as well. The project and its attributes imparted a lot of knowledge about the vehicle sector, its dependable and the various criteria which varies the prices of various used cars

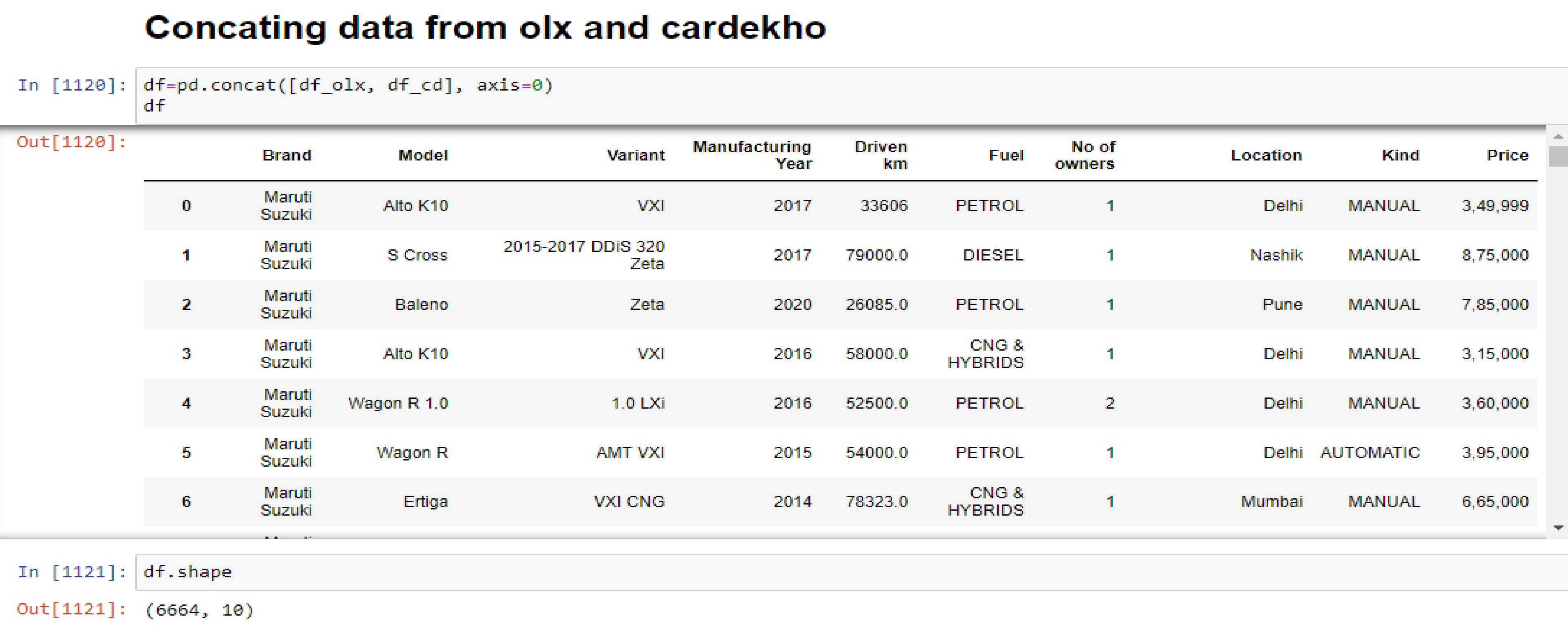
Data Collection

The data is scraped from 2 websites-

1. olx.com
2. Cardekho.com

From olx.com, the used car data is scraped based on the brands
From cardekho.com, the used car data is scraped based on their location

6664 used car data is scraped from both the websites.



Concating data from olx and cardekho

```
In [1120]: df=pd.concat([df_olx, df_cd], axis=0)
           df
```

Out[1120]:

	Brand	Model	Variant	Manufacturing Year	Driven km	Fuel	No of owners	Location	Kind	Price
0	Maruti Suzuki	Alto K10	VXI	2017	33606	PETROL	1	Delhi	MANUAL	3,49,999
1	Maruti Suzuki	S Cross	2015-2017 DDiS 320 Zeta	2017	79000.0	DIESEL	1	Nashik	MANUAL	8,75,000
2	Maruti Suzuki	Baleno	Zeta	2020	26085.0	PETROL	1	Pune	MANUAL	7,85,000
3	Maruti Suzuki	Alto K10	VXI	2016	58000.0	CNG & HYBRIDS	1	Delhi	MANUAL	3,15,000
4	Maruti Suzuki	Wagon R 1.0	1.0 LXi	2016	52500.0	PETROL	2	Delhi	MANUAL	3,60,000
5	Maruti Suzuki	Wagon R	AMT VXI	2015	54000.0	PETROL	1	Delhi	AUTOMATIC	3,95,000
6	Maruti Suzuki	Ertiga	VXI CNG	2014	78323.0	CNG & HYBRIDS	1	Mumbai	MANUAL	6,65,000

```
In [1121]: df.shape

Out[1121]: (6664, 10)
```

ANALYTICAL PROBLEM FRAMING

<u>EDA Steps and Visualization</u>

- The web scraped data is saved onto an excel sheet
- The excel datasheet is extracted and saved in a dataframe
- The shape of the dataframe is checked-

  There are 6664 rows and 10 columns
- **The columns are as follows-**
  - Brand
  - Model
  - Variant
  - Manufacturing Year
  - Driven km
  - Fuel
  - No of owners
  - Location
  - Kind
  - Price

The data type of each column is-

- Brand - object
- Model - object
- Variant - object
- Manufacturing Year - float64
- Driven km - object
- Fuel- object
- No of owners - object
- Location - object
- Kind - object
- Price – object

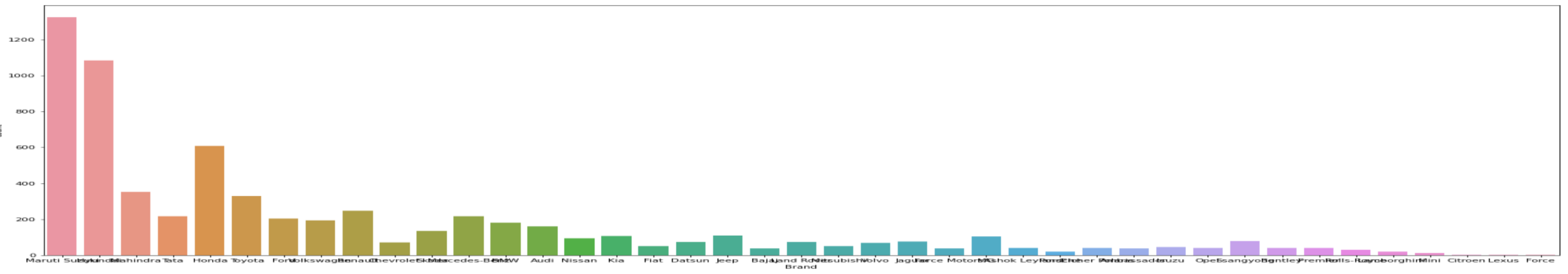
The null values are checked. The whitespaces, and dashes ('—') are replaced by null values-

- Variant - 179
- Driven km - 22
- Fuel- 32
- No of owners - 5298
- Kind - 92

As the No of owners is mostly consists of null values, the column is safe to be dropped

The data visualization, value counts encoding and imputation of null values for each column

Brand

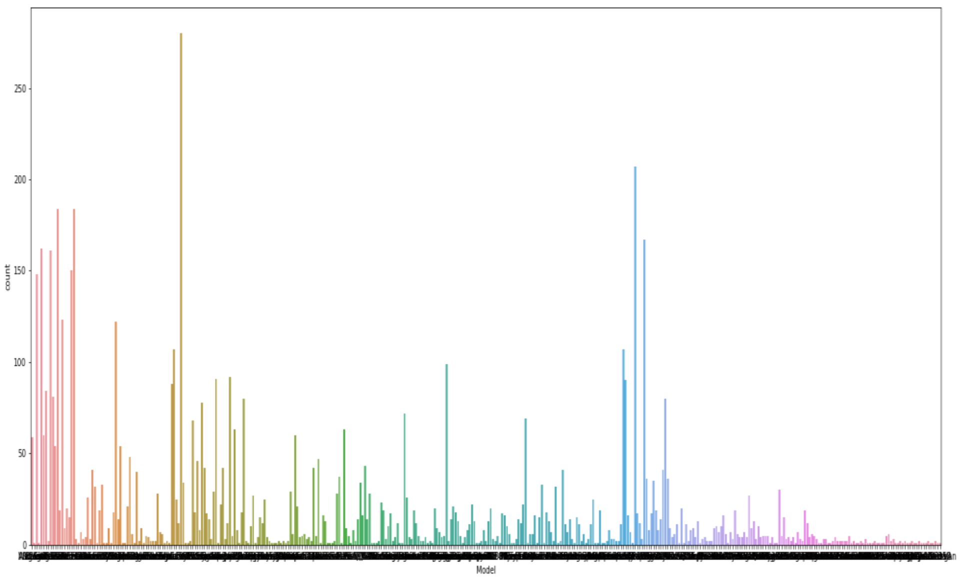


- 1325 used cars are Maruti Suzuki
- 1085 used cars are Hyundai
- 609 used cars are Honda
- 352 used cars are Mahindra
- 329 used cars are Toyota
- 247 used cars are Renault
- 218 used cars are Tata
- 217 used cars are Mercedes-Benz
- 204 used cars are Ford
- 194 used cars are Volkswagen
- 182 used cars are BMW
- 161 used cars are Audi
- 135 used cars are Skoda
- 109 used cars are Jeep
- 106 used cars are Kia
- 105 used cars are MG
- 95 used cars are Nissan
- 78 used cars are Ssangyong
- 76 used cars are Jaguar
- 74 used cars are Land Rover
- 73 used cars are Datsun
- 72 used cars are Chevrolet
- 69 used cars are Volvo
- 50 used cars are Fiat
- 50 used cars are Mitsubishi
- 45 used cars are Isuzu
- 40 used cars are Opel
- 40 used cars are Premier
- 40 used cars are Ashok Leyland
- 40 used cars are Bentley
- 40 used cars are Eicher Polaris
- 39 used cars are Bajaj
- 39 used cars are Ambassador
- 39 used cars are Force Motors
- 31 used cars are Rolls-Royce
- 21 used cars are Porsche
- 19 used cars are Lamborghini
- 13 used cars are Mini
- 1 used car is Citroen
- 1 used car is Lexus
- 1 used car is Force

$\downarrow$

$\boxed{\text{Encoding object data in numeric using Label Encoder}}$

Model

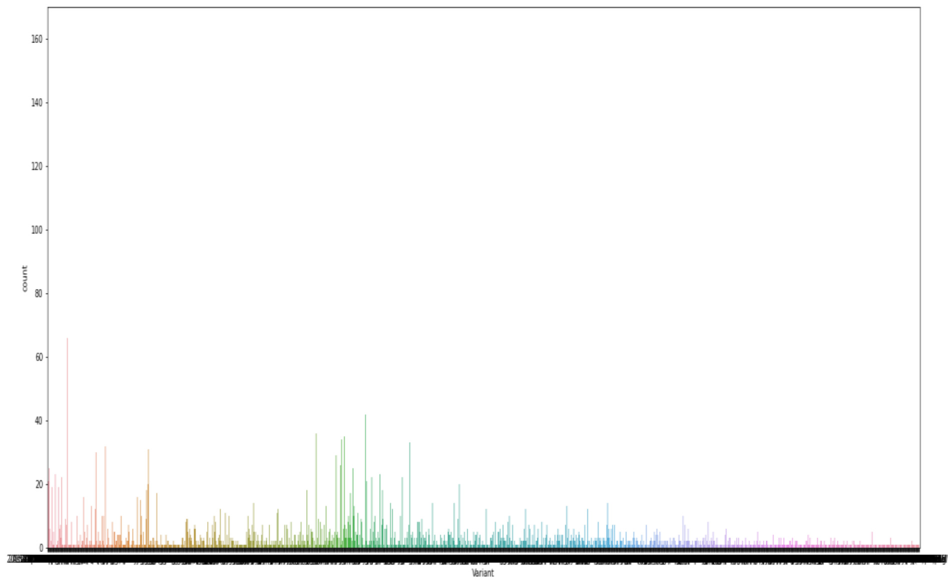


- Majority of the used cars are of the model City, i20, Swift, Creta and Grand i10

↓

Encoding object data in numeric using Label Encoder

Variant



- Majority of the used cars are of the model variant VXI, LXI, Sportz and VDI

↓

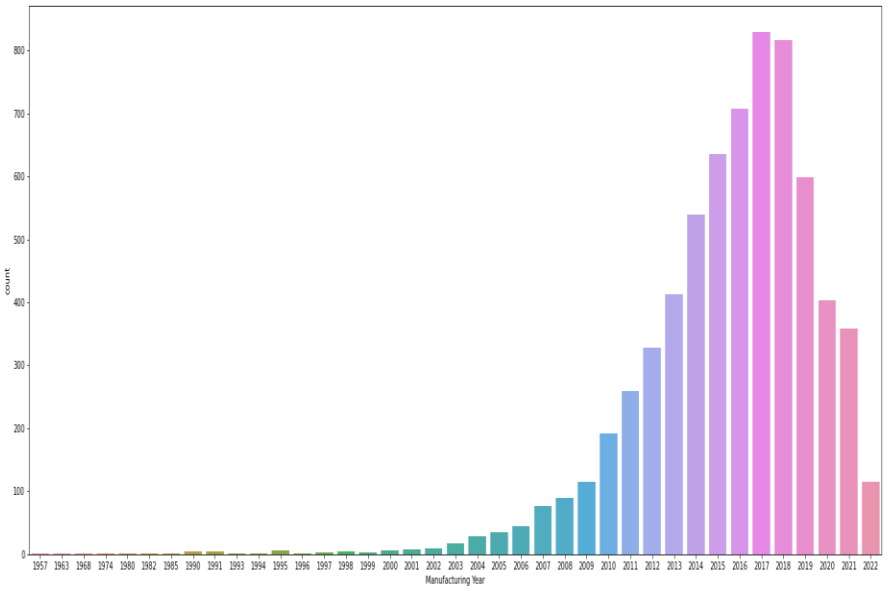
Encoding object data in numeric using Label Encoder

↓

Replacing wrongly encoded null value with null value

Manufacturing Year

Converting Manufacturing Year to int data type

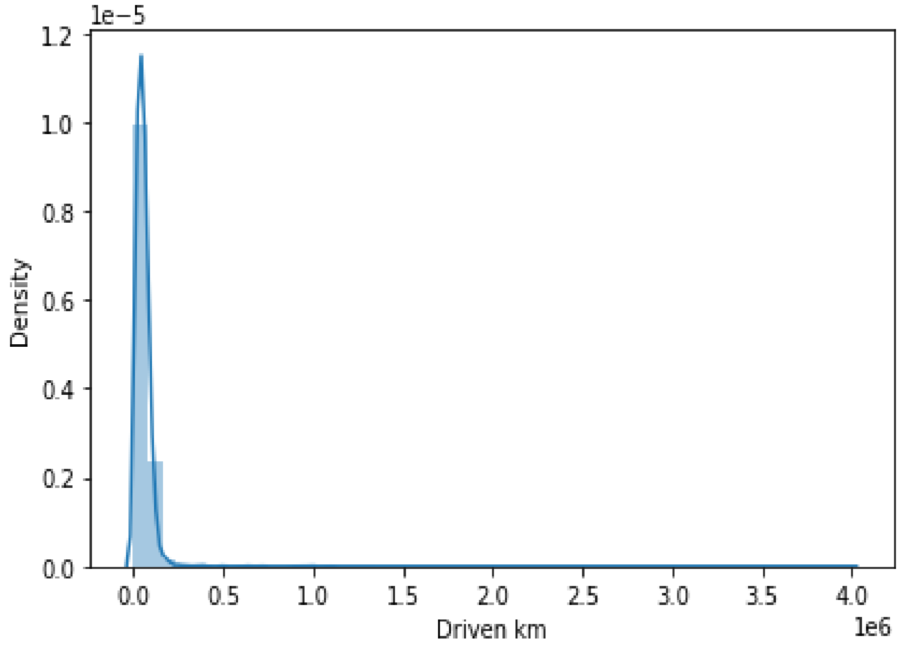


- Majority of the used cars are manufactured between 2010-2020

The data is skewed and will be transformed later

Driven km

Converting Driven km to float data type

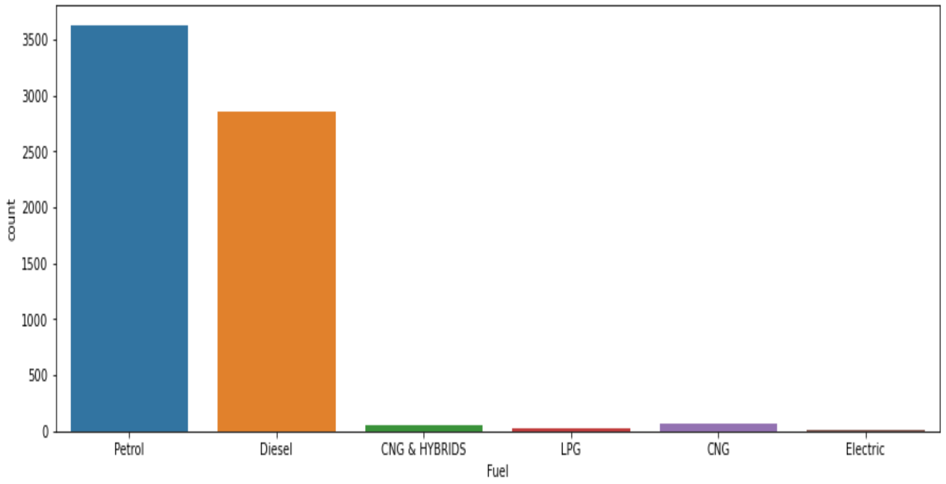


- 92 used cars have driven for 70000km

The data is skewed and will be transformed later

Fuel

Replacing Redundant data

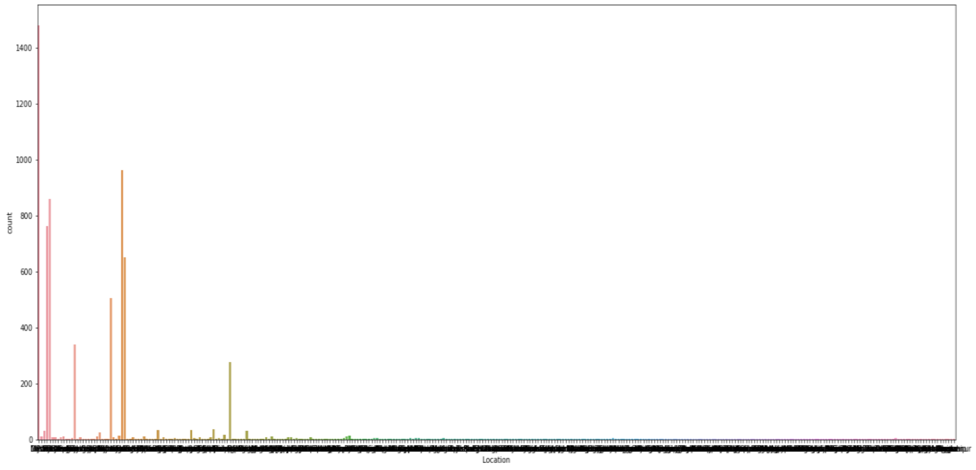


- 3626 used cars require Petrol as fuel
- 2858 used cars require Diesel as fuel
- 63 used cars require CNG as fuel
- 50 used cars require CNG & HYBRIDS as fuel
- 19 used cars require LPG as fuel
- 16 used cars require Electric as fuel

Encoding object data in numeric using Label Encoder

Replacing wrongly encoded null value with null value

Location

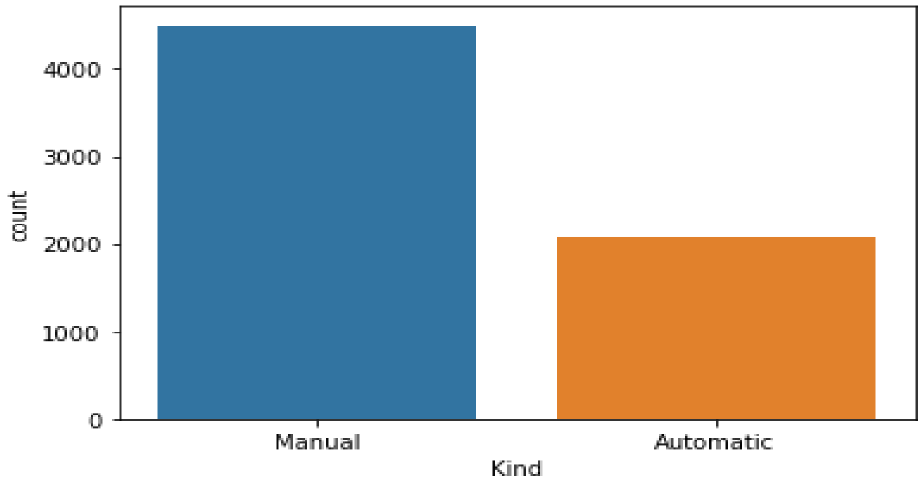


- Majority of the used cars are available at Delhi, Bengaluru, Hyderabad, Mumbai, Chennai and Kolkata

Encoding object data in numeric using Label Encoder

Kind

Replacing Redundant data



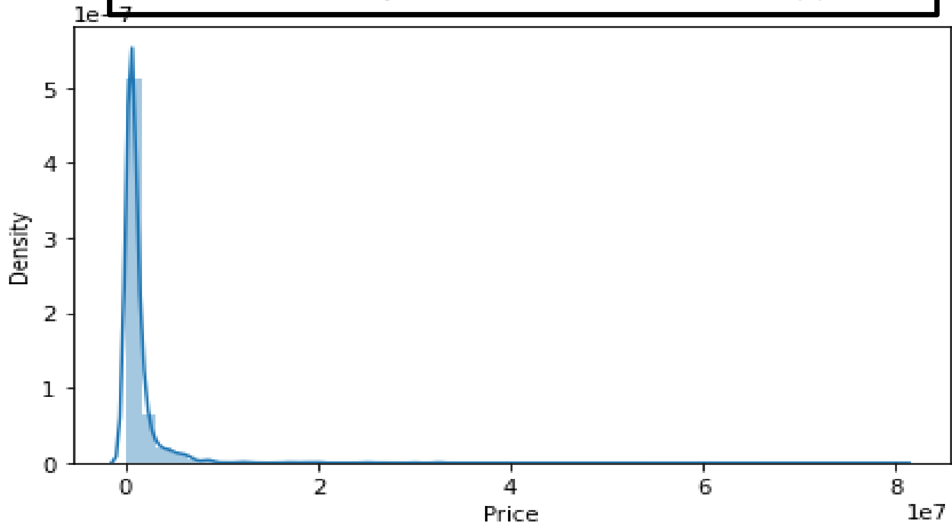
- 4493 used cars are manual
- 2079 used cars are automatic

Encoding object data in numeric using Label Encoder

Replacing wrongly encoded null value with null value

Price

Converting Price to float data type

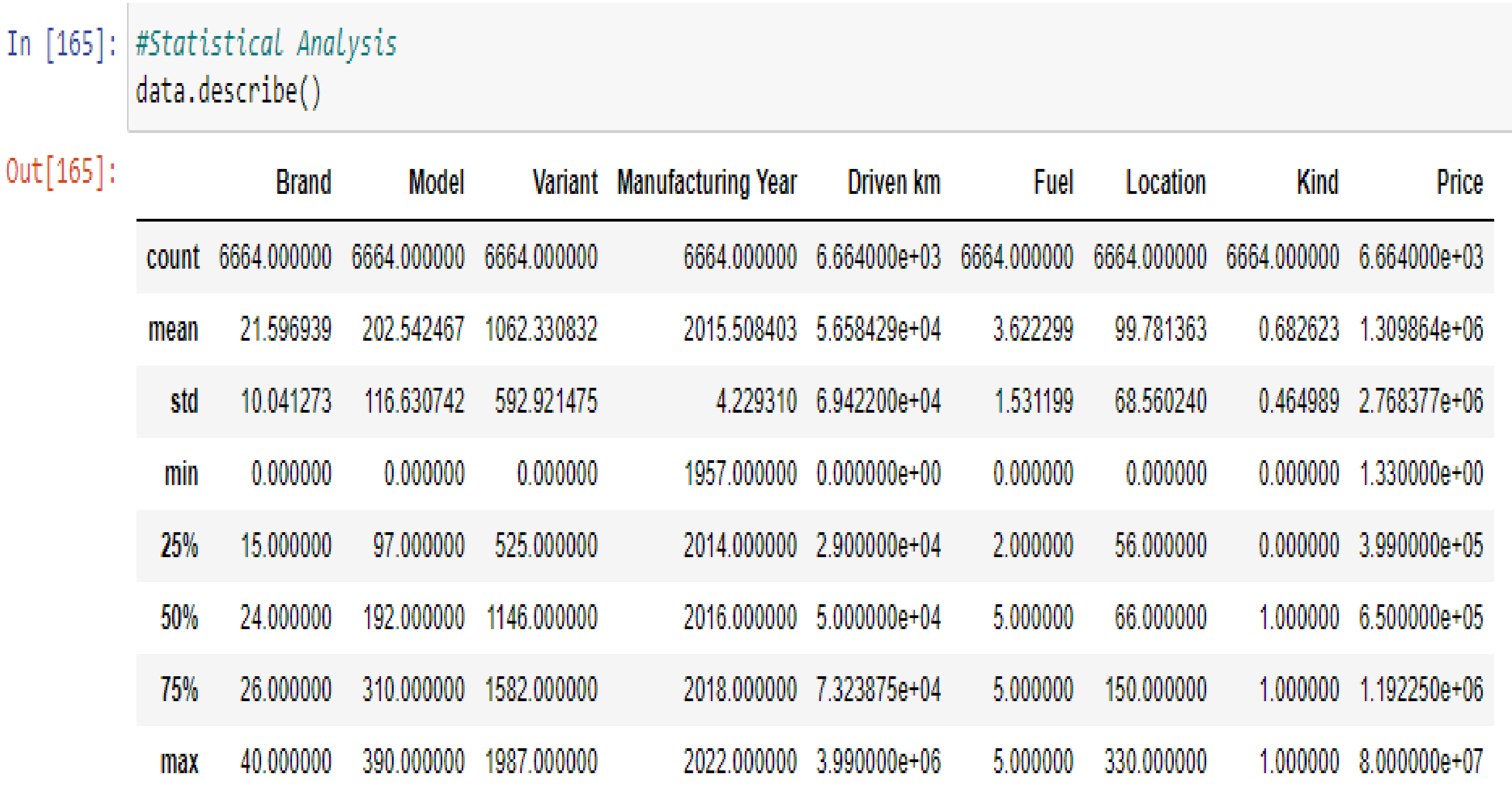


- Majority of the used cars are being sold at a range of 4-6 Lakhs

The data is skewed and will be transformed later

- The null values in 'Variant ‘, ‘Driven km’, ‘Fuel’ and 'Kind ‘ are imputed using KNN Imputer.

- Statistical analysis using describe method-

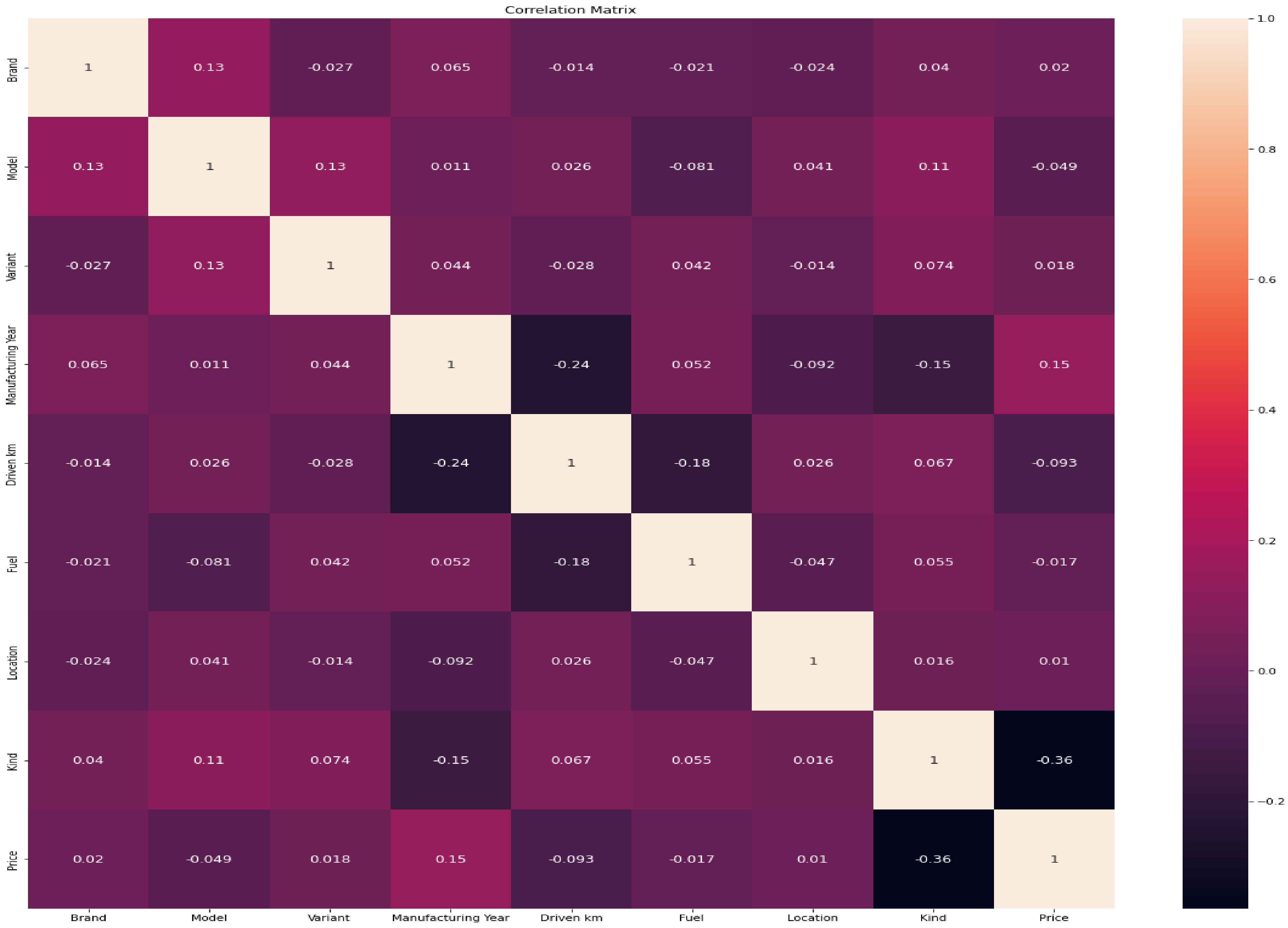


```
In [165]: #Statistical Analysis
          data.describe()
```

Out[165]:

	Brand	Model	Variant	Manufacturing Year	Driven km	Fuel	Location	Kind	Price
count	6664.000000	6664.000000	6664.000000	6664.000000	6.664000e+03	6664.000000	6664.000000	6664.000000	6.664000e+03
mean	21.596939	202.542467	1062.330832	2015.508403	5.658429e+04	3.622299	99.781363	0.682623	1.309864e+06
std	10.041273	116.630742	592.921475	4.229310	6.942200e+04	1.531199	68.560240	0.464989	2.768377e+06
min	0.000000	0.000000	0.000000	1957.000000	0.000000e+00	0.000000	0.000000	0.000000	1.330000e+00
25%	15.000000	97.000000	525.000000	2014.000000	2.900000e+04	2.000000	56.000000	0.000000	3.990000e+05
50%	24.000000	192.000000	1146.000000	2016.000000	5.000000e+04	5.000000	66.000000	1.000000	6.500000e+05
75%	26.000000	310.000000	1582.000000	2018.000000	7.323875e+04	5.000000	150.000000	1.000000	1.192250e+06
max	40.000000	390.000000	1987.000000	2022.000000	3.990000e+06	5.000000	330.000000	1.000000	8.000000e+07

The Correlation Matrix using heatmap

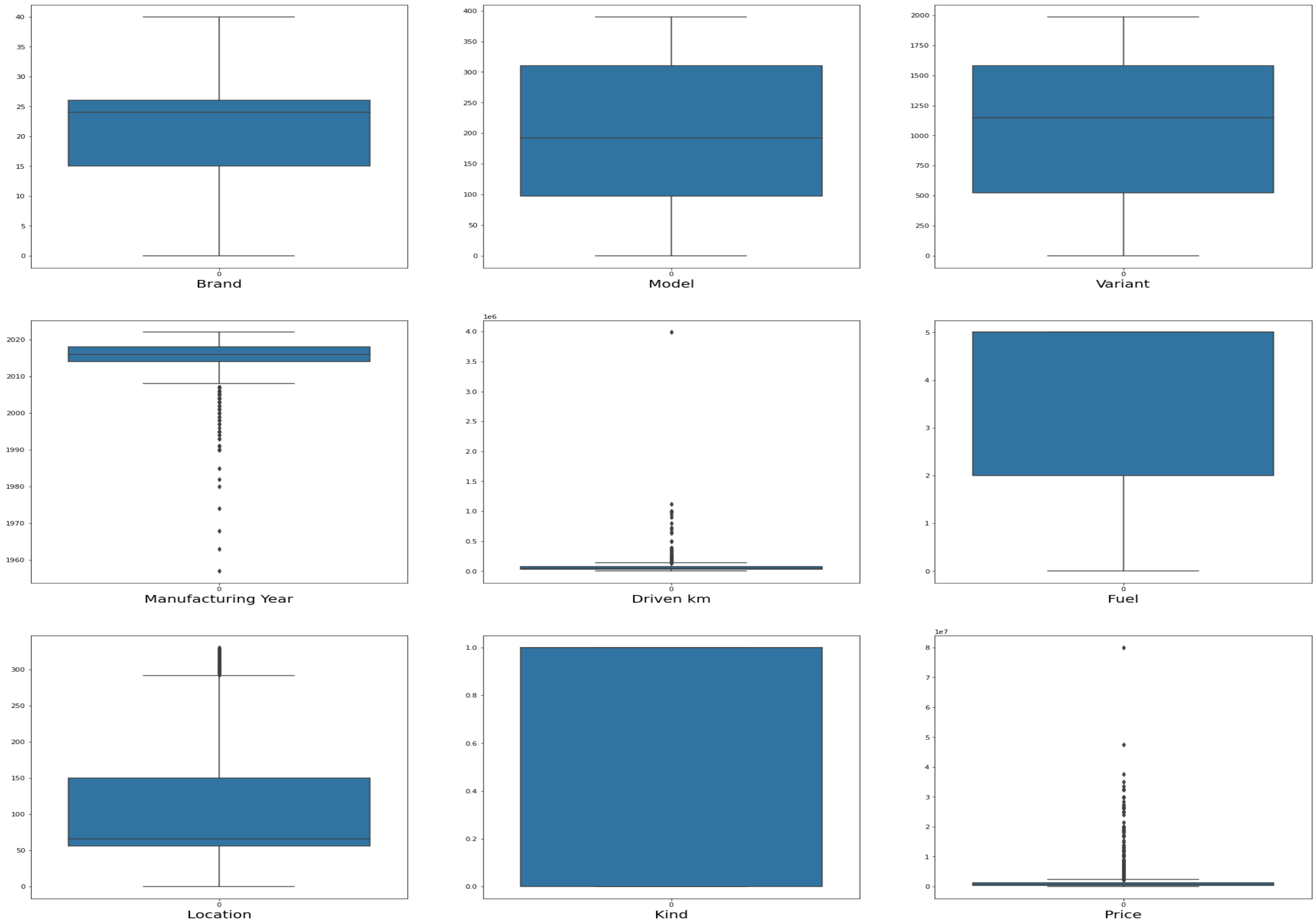


	Brand	Model	Variant	Manufacturing Year	Driven km	Fuel	Location	Kind	Price
Brand	1	0.13	-0.027	0.065	-0.014	-0.021	-0.024	0.04	0.02
Model	0.13	1	0.13	0.011	0.026	-0.081	0.041	0.11	-0.049
Variant	-0.027	0.13	1	0.044	-0.028	0.042	-0.014	0.074	0.018
Manufacturing Year	0.065	0.011	0.044	1	-0.24	0.052	-0.092	-0.15	0.15
Driven km	-0.014	0.026	-0.028	-0.24	1	-0.18	0.026	0.067	-0.093
Fuel	-0.021	-0.081	0.042	0.052	-0.18	1	-0.047	0.055	-0.017
Location	-0.024	0.041	-0.014	-0.092	0.026	-0.047	1	0.016	0.01
Kind	0.04	0.11	0.074	-0.15	0.067	0.055	0.016	1	-0.36
Price	0.02	-0.049	0.018	0.15	-0.093	-0.017	0.01	-0.36	1

- Correlation between the columns and the label ‘Price’ using corr method-
    - Price : 1.000000
    - Manufacturing Year : 0.147913
    - Brand : 0.019558
    - Variant : 0.017863
    - Location : 0.010105
    - Fuel : -0.017235
    - Model : -0.049014
    - Driven km : -0.093212
    - Kind : -0.364608

- Manufacturing Year is 14.79% positively correlated to ‘Price’
- Brand is 1.95% positively correlated to ‘Price’
- Variant is 1.78% positively correlated to ‘Price’
- Location is 1% positively correlated to ‘Price’
- Fuel is 1.72% negatively correlated to ‘Price’
- Model is 4.94% negatively correlated to ‘Price’
- Driven km is 9.32% negatively correlated to ‘Price’
- Kind is 36.46% negatively correlated to ‘Price’

- Visualizing outliers using boxplot method-



- Removing outliers using zscore method-

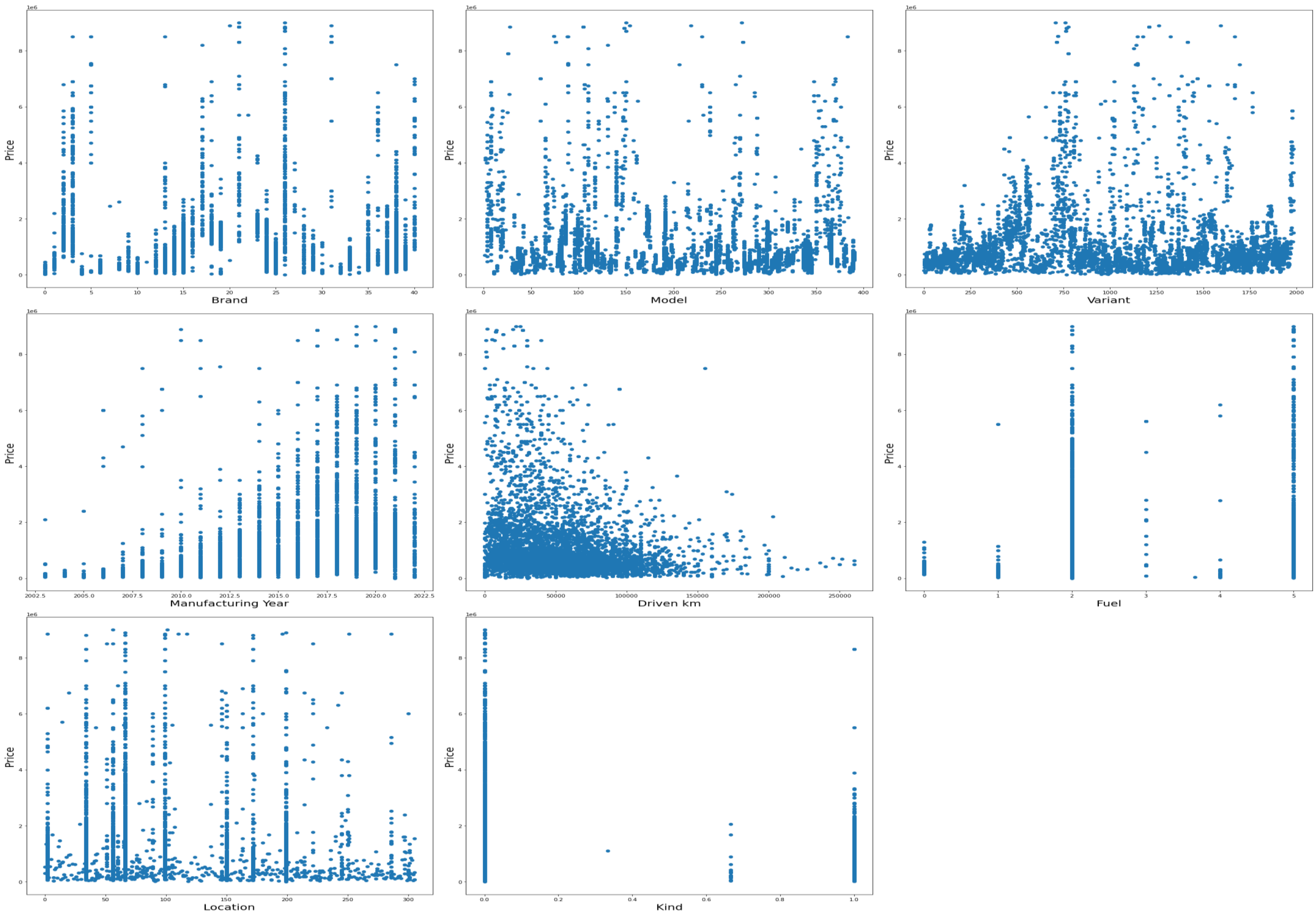
On removing the outliers the data loss is 3%, which is acceptable, hence outliers are removed

- The dataset is divided into x(features) and y (label)-

The x contains all the features other than the label ‘Price’

The y contains only the label ‘Price’

- Visualizing relationship between features and label-

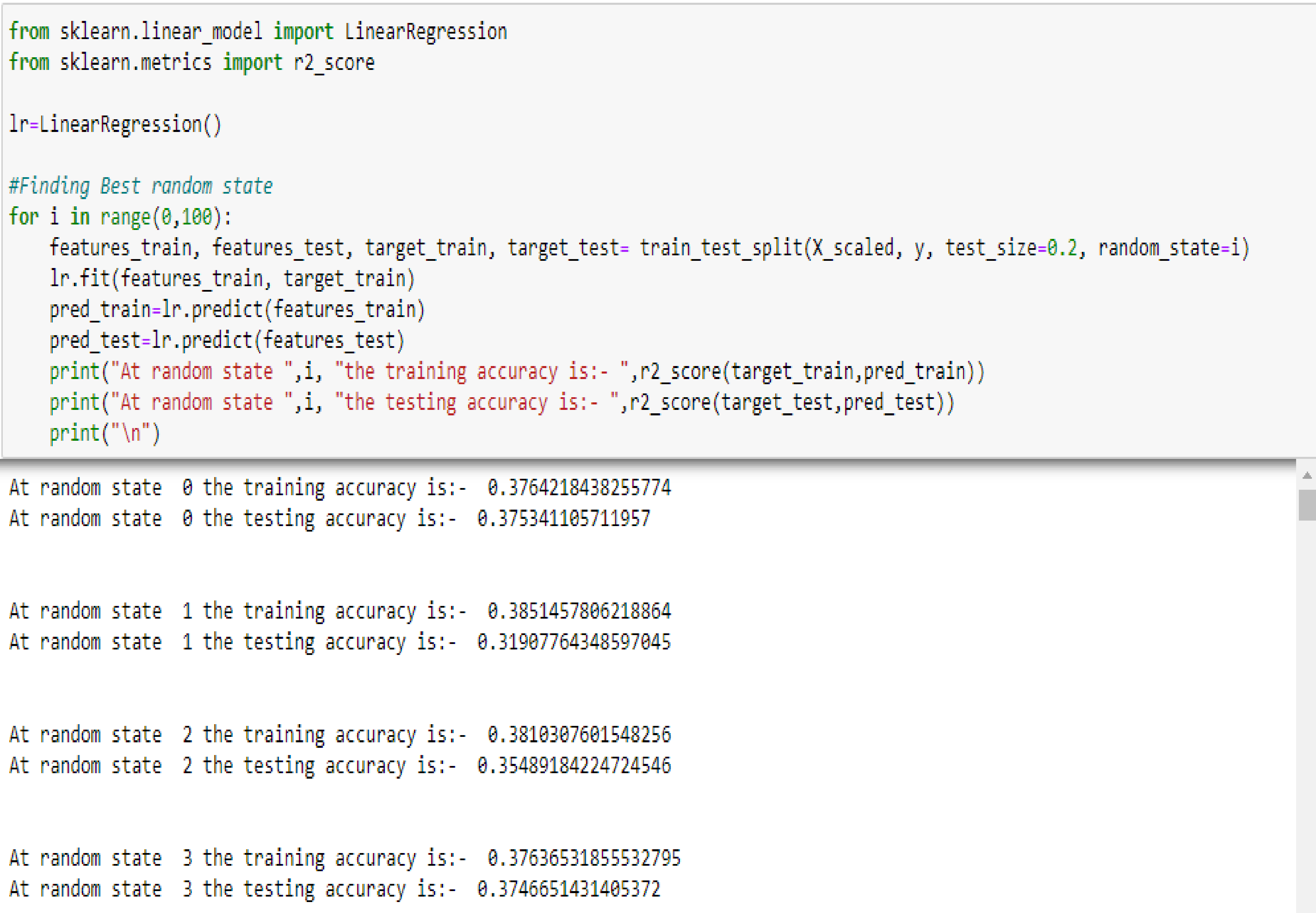


- The skewness observed in graphical analysis was confirmed by using the skew method-
    - Driven km : 1.082297
    - Location : 0.801142
    - Model : 0.025468
    - Brand : 0.023501
    - Variant : -0.213326
    - Fuel : -0.277616
    - Manufacturing Year : -0.760945
    - Kind : -0.814154
- This skewness was removed using the power transformer
- The x(features) were scaled using the Standard Scaler
- The dataset was divided into train and test set using train test split and the best random state was found to be 62

Sofware Requirements-

- Jupyter Notebook – Interface for the program
- Pandas – for datafram working
- Numpy – to deal with null data
- matplotlib.pyplot – for data visualization
- Seaborn - for data visualization
- Warnings- to omit warnings
- sklearn.preprocessing – to import LabelEncoder, powertransform
- Label Encoder- to encode object data to numeric data
- sklearn.impute- to import KNNImputer
- KNNImputer- to fill in the null values with meaningful data
- scipy.stats – to import zscore
- Zscore- to remove outliers
- sklearn.feature_selection- to import SelectPercentile and chi2
- SelectPercentile- to select best features
- Chi2-chi2 retrives p-values which help in filtering the best features
- power_transform- to remove the skewness in the data
- sklearn.model_selection- to import train_test_split
- train_test_split- to spit dataset into train and test samples
- sklearn.linear_model- to import LinearRegression
- LinearRegression- to use LinearRegression model
- sklearn.metrics- to import r2_score
- R2 score- to check for the efficiency of the model
- sklearn.model_selection- to import cross_val_score
- cross_val_score- to check for overfitting and underfitting
- sklearn.model_selection- to import GridSearchCV
- GridSearchCV to enhance the working of the model by manipulating the parameters
- from sklearn.linear_model-to import Lasso
- Lasso- to regularize the linear regression model
- sklearn.ensemble – to import RandomForestRegressor, AdaBoostRegressor
- RandomForestRegressor- to use Random Forest Regressor model
- AdaBoostRegressor- to use AdaBoostRegressor

Train Test Split



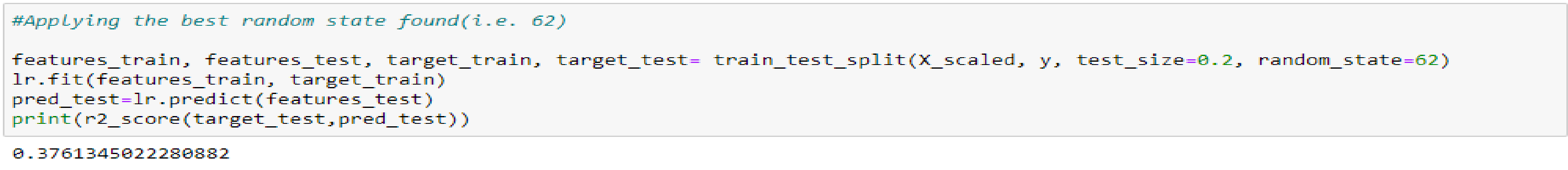
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

lr=LinearRegression()

#Finding Best random state
for i in range(0,100):
    features_train, features_test, target_train, target_test= train_test_split(X_scaled, y, test_size=0.2, random_state=i)
    lr.fit(features_train, target_train)
    pred_train=lr.predict(features_train)
    pred_test=lr.predict(features_test)
    print("At random state ",i, "the training accuracy is:- ",r2_score(target_train,pred_train))
    print("At random state ",i, "the testing accuracy is:- ",r2_score(target_test,pred_test))
    print("\n")
```

```
At random state  0 the training accuracy is:-  0.3764218438255774
At random state  0 the testing accuracy is:-  0.375341105711957



At random state  1 the training accuracy is:-  0.3851457806218864
At random state  1 the testing accuracy is:-  0.31907764348597045



At random state  2 the training accuracy is:-  0.3810307601548256
At random state  2 the testing accuracy is:-  0.35489184224724546



At random state  3 the training accuracy is:-  0.37636531855532795
At random state  3 the testing accuracy is:-  0.3746651431405372
```

Model/s Development and Evaluation

The train and test data were applied on different models as follows

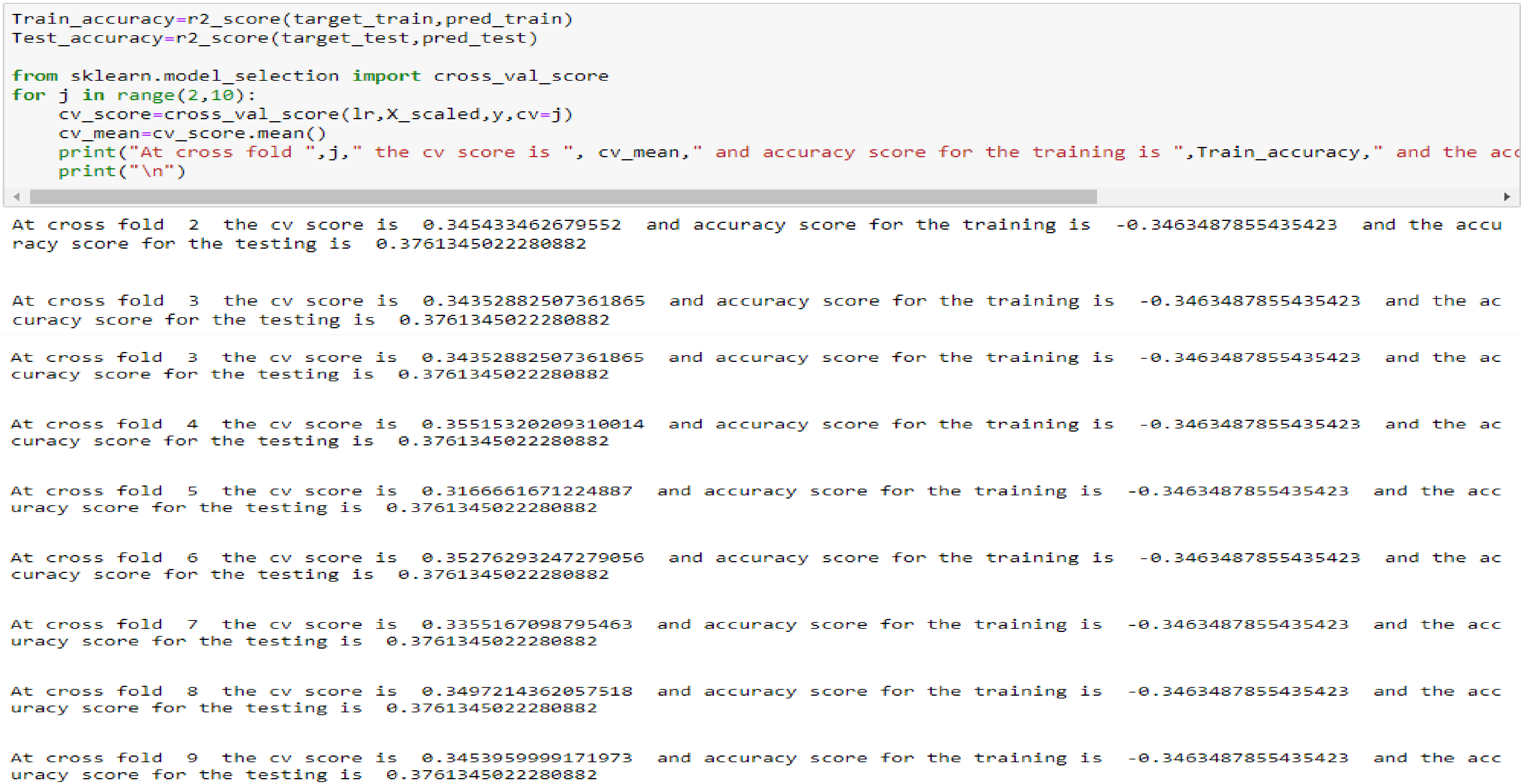
Linear Regression Model



```
#Applying the best random state found(i.e. 62)

features_train, features_test, target_train, target_test= train_test_split(X_scaled, y, test_size=0.2, random_state=62)
lr.fit(features_train, target_train)
pred_test=lr.predict(features_test)
print(r2_score(target_test,pred_test))
```

```
0.3761345022280882
```

Cross Validation of the model



```
Train_accuracy=r2_score(target_train,pred_train)
Test_accuracy=r2_score(target_test,pred_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(lr,X_scaled,y,cv=j)
    cv_mean=cv_score.mean()
    print("At cross fold ",j," the cv score is ", cv_mean," and accuracy score for the training is ",Train_accuracy," and the acc
    print("\n")
```

```
At cross fold  2  the cv score is  0.345433462679552  and accuracy score for the training is  -0.3463487855435423  and the accu
racy score for the testing is  0.3761345022280882


At cross fold  3  the cv score is  0.34352882507361865  and accuracy score for the training is  -0.3463487855435423  and the ac
curacy score for the testing is  0.3761345022280882
```

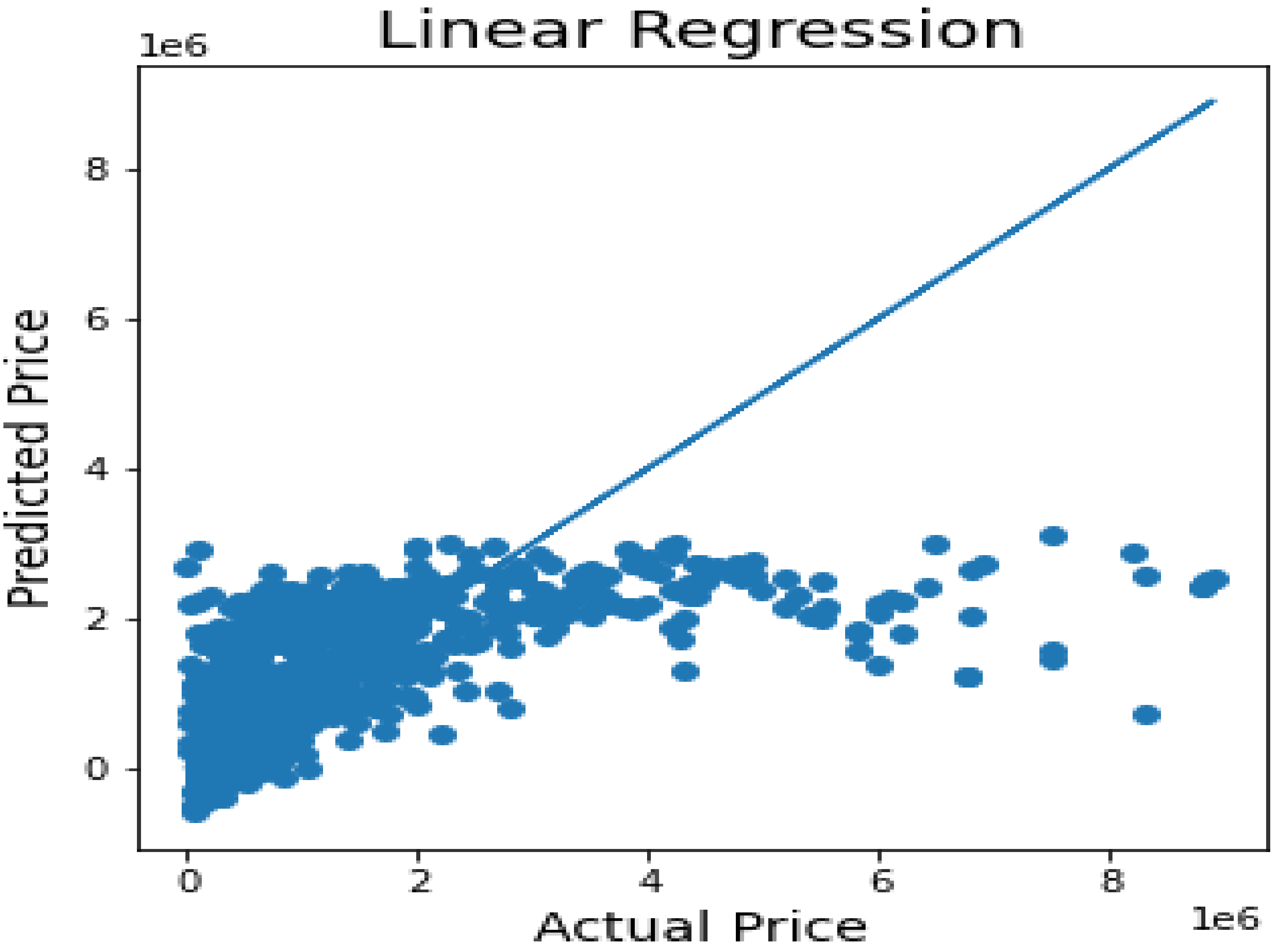
```
At cross fold  3  the cv score is  0.34352882507361865  and accuracy score for the training is  -0.3463487855435423  and the ac
curacy score for the testing is  0.3761345022280882
```

```
At cross fold  4  the cv score is  0.35515320209310014  and accuracy score for the training is  -0.3463487855435423  and the ac
curacy score for the testing is  0.3761345022280882


At cross fold  5  the cv score is  0.3166661671224887  and accuracy score for the training is  -0.3463487855435423  and the acc
uracy score for the testing is  0.3761345022280882


At cross fold  6  the cv score is  0.35276293247279056  and accuracy score for the training is  -0.3463487855435423  and the ac
curacy score for the testing is  0.3761345022280882


At cross fold  7  the cv score is  0.3355167098795463  and accuracy score for the training is  -0.3463487855435423  and the acc
uracy score for the testing is  0.3761345022280882


At cross fold  8  the cv score is  0.3497214362057518  and accuracy score for the training is  -0.3463487855435423  and the acc
uracy score for the testing is  0.3761345022280882


At cross fold  9  the cv score is  0.3453959999171973  and accuracy score for the training is  -0.3463487855435423  and the acc
uracy score for the testing is  0.3761345022280882
```

- The **R2 score** for target test and pred_test(data predicted on features_test) is **37.61%**

- Upon cross-validation it was observed that the number of folds did not have such impact on the accuracy and cv score. So cv=9 is selected. Here we have handled the problem of the overfitting and the underfitting by checking the training and testing score

- The graph between Actual Price and Predicted Price depicts the best fit line which passes through maximum of the points, hence suggesting that the model works well-



Linear Regression
1e6
8
6
4
2
0
Predicted Price
0
2
4
6
8
1e6
Actual Price

Regularization

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Lasso
parameters={'alpha':[.0001, .001, .01,.1, 1, 10], 'random_state':list(range(0,10))}
ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(features_train,target_train)
print(clf.best_params_)
```

```
{'alpha': 10, 'random_state': 0}
```

```
ls=Lasso(alpha=10, random_state=0)
ls.fit(features_train,target_train)
ls.score(features_train, target_train)
pred_ls=ls.predict(features_test)

lss=r2_score(target_test,pred_ls)
lss
```

```
0.3761342266507416
```

```
cv_score=cross_val_score(ls,X_scaled,y,cv=4)
cv_mean=cv_score.mean()
cv_mean
```

```
0.3551551832404276
```

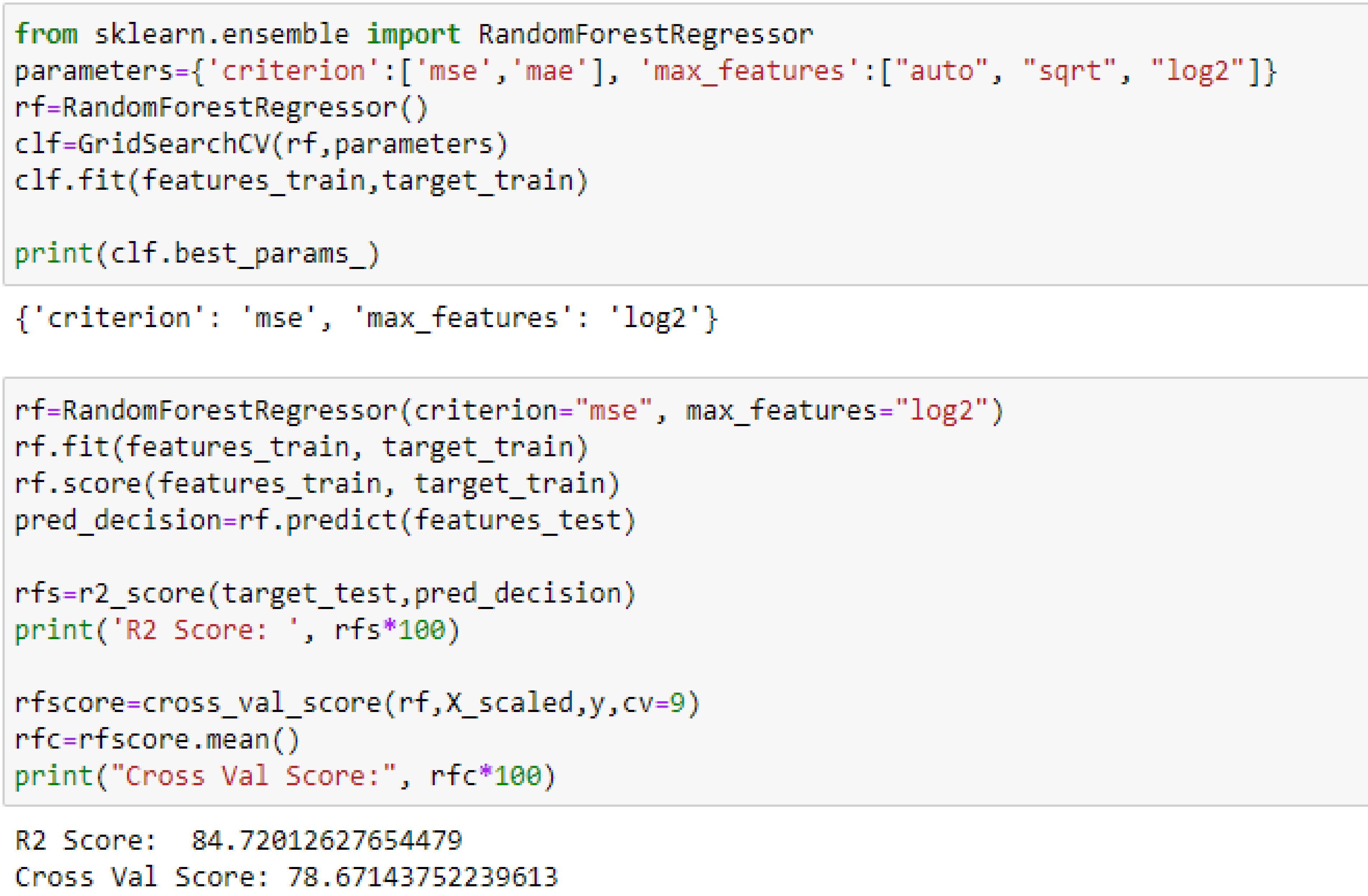
- The Linear Regression Model is Lasso regularized with the aid of GridSearchCV

- The best parameters for alpha and random_state are found as follows-
    - **alpha: 10**
    - **random_state: 0**

- Applying the above found best parameters on Lasso regularized Linear Regression Model, the following was obtained-

    - **R2 score for target_test and pred_test(data predicted on features_test) – 37.61%**
    - **CV score- 35.51%**

Random Forest Regressor Model



```
from sklearn.ensemble import RandomForestRegressor
parameters={'criterion':['mse','mae'], 'max_features':["auto", "sqrt", "log2"]}
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(features_train,target_train)

print(clf.best_params_)
```

```
{'criterion': 'mse', 'max_features': 'log2'}
```

```
rf=RandomForestRegressor(criterion="mse", max_features="log2")
rf.fit(features_train, target_train)
rf.score(features_train, target_train)
pred_decision=rf.predict(features_test)

rfs=r2_score(target_test,pred_decision)
print('R2 Score: ', rfs*100)

rfscore=cross_val_score(rf,X_scaled,y,cv=9)
rfc=rfscore.mean()
print("Cross Val Score:", rfc*100)
```

```
R2 Score:  84.72012627654479
Cross Val Score: 78.67143752239613
```

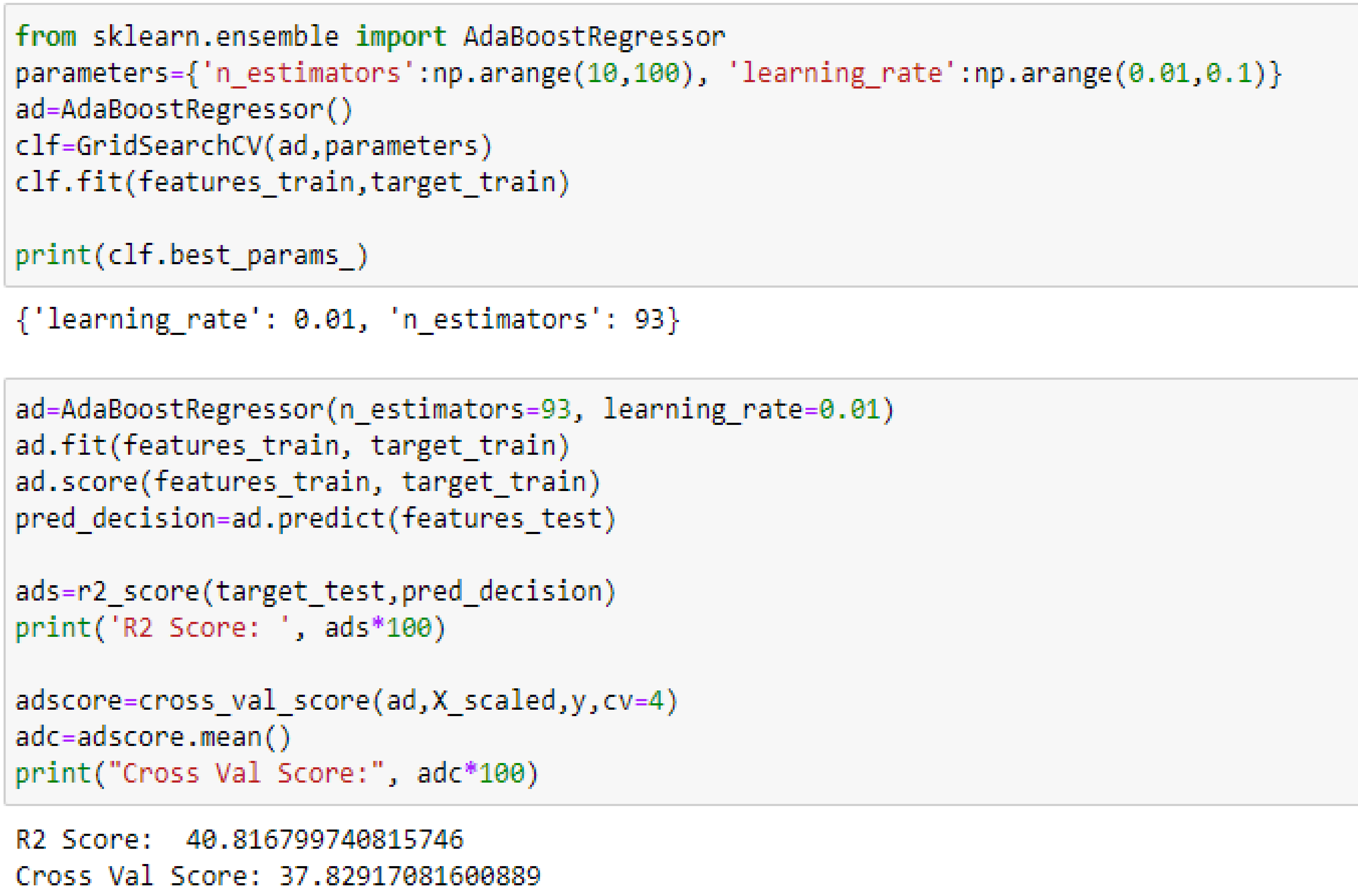
- Random Forest Regressor Model is hyperparameter tuned using GridSearchCV

- The best parameters for criterion and max_features are found as follows-
    - **criterion: mse**
    - **max_features: log2**

- Applying the above found best parameters on Random Forest Regressor Model, the following was obtained-

    - **R2 score for target_test and pred_test (data predicted on features_test) –84.72%**
    - **CV score- 78.67%**

Ada Boost Regressor Model



```
from sklearn.ensemble import AdaBoostRegressor
parameters={'n_estimators':np.arange(10,100), 'learning_rate':np.arange(0.01,0.1)}
ad=AdaBoostRegressor()
clf=GridSearchCV(ad,parameters)
clf.fit(features_train,target_train)

print(clf.best_params_)
```

```
{'learning_rate': 0.01, 'n_estimators': 93}
```

```
ad=AdaBoostRegressor(n_estimators=93, learning_rate=0.01)
ad.fit(features_train, target_train)
ad.score(features_train, target_train)
pred_decision=ad.predict(features_test)

ads=r2_score(target_test,pred_decision)
print('R2 Score: ', ads*100)

adscore=cross_val_score(ad,X_scaled,y,cv=4)
adc=adscore.mean()
print("Cross Val Score:", adc*100)
```

```
R2 Score:  40.816799740815746
Cross Val Score: 37.82917081600889
```

- Ada Boost Regressor Model is hyperparameter tuned using GridSearchCV

- The best parameters for n_estimators and learning_rate are found as follows-
    - **learning_rate: 0.01**
    - **n_estimators: 93**

- Applying the above found best parameters on Ada Boost Regressor Model, the following was obtained-

    - **R2 score for target_test and pred_test(data predicted on features_test) –40.82%**
    - **CV score- 37.83%**

Model	R2 score for target_test and pred_test (data predicted on features_test)	CV score
Linear Regression Model	37.61%	35.51%
Random Forest Regressor Model	84.72%	78.67%
AdaBoost Regressor Model	40.82%	37.83%

- The R2 score of Random Forest Regressor is 84.72% and CV score of Random Forest Regressor is 78.67%. This is the best working model and is finalized
- This model can be used to predict the selling price of used cars.

CONCLUSION

- It is essential to scrape diverse data to allow the model to be applicable for versatile inputs
- The EDA analysis of the data is essential as it helps to understand the relationship between the target and features as well as omit out unwanted columns, thereby taking care of overfitting scenario
- It is necessary to have the data encoded properly as well as the null values must be handled with care in order to avoid mis-interpretation of data
- The models should be used properly, as their regularization /hyperparamter tuning is highly advisable for the best outcome.
- The project imparted key knowledge about the car retail market and how beneficial it is to estimate a proper price of the car to enhance the company's returns
- The limitation of the solution is that it predicts the sale price of the car but does not classify it to be purchasable or not. The limitation can be handled, that post the regression model, the findings can be subjected to classification models to classify the used cars to be purchased and not to be purchased