SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Program Offered	**M. Tech /*AI***
Course Title	Machine Learning-Supervised Regression.
Name of the Project	Mini Project-2
Members	**GROUP-5** BHARATH K M (PA2212049010019) DIWAHAR A K (PA2212049010012) RAGHUNATH M (PA2212049010050)

Assignment Question	CRISP DM	Refer Section.
1. Read the data coefficients. Load the csv file and set the first column as index	Business Understanding Data Understanding Data Preparation	Business Understanding - Section 4 Reading the data with first column as index – Section 5.1 Data Understanding- Section 5.2 Data Preparation Numerical Variable- Section 5.3 Categorical Variables- Section 5.4 Null Value Treatment- Section 5.5.1 Date Conversion- Section 5.5.2 Duplicate Removal- Section 5.5.3 Outlier Analysis- Section 5.5.4 Insignificant Feature Analysis- Section 6.1 Numerical Analysis- Section 6.3.1 Categorical Analysis- Section 6.3.2 Encoding- Section 6.4 Feature Transformation- Section 6.6 Feature Scaling- Section 6.7
2. Build a full model and interpret the beta coefficients	Modelling	Test- Train Split- Section 6.11 Linear Regression (all features)- Section 7.1 Model Summary- Section 7.2 Linear Equation- Section 7.3 Model Prediction- Section 7.4
3. What is the impact of fuel type of cars on the selling price?	Model Evaluation	Impact of Fuel Type with selling price- Section 7.5
4. Does the model significantly explain variation in the target variable? Justify your answer	Model Evaluation	Model Evaluation- Section 7.6 Impact Analysis of each Feature- Section 7.7

Table of Contents

1 Purpose

The purpose of this document to report on the analysis performed to predict the selling price of the used vehicle.

2 Scope

The scope of this document to build the model based on Linear Regression Model. The scope covers the following area to predict the selling price of the used vehicle.

a. Data understanding
b. Data Preparation
c. Model Building
d. Model Evaluation

3 Environment Preparation for Data Analysis

3.1 Tools Selection

As we are 3 members in Group 5 and all we work remotely, we were using the google Collaboratory to have better interactions between us. The below is the Google Colab link we were using for this assignment.

https://colab.research.google.com/drive/16Pr0hfBCN-5qYzcJfqc3eX3a2RxX_DMF?usp=sharing

3.2 Dataset and accessing of dataset.

To have the common working among us, we have placed our data set in the github in the following path so that anyone of us can access the dataset directly through colab.

https://github.com/akdiwahar/dataset/raw/main/SRM/MLSR/CT2/download.csv

3.3 Importing the python libraries.

```
# 'Pandas'
import pandas as pd

# 'Numpy'
import numpy as np

# 'SciPy'
from scipy.stats import norm

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt

# 'Statsmodels' is used to build and analyze various statistical models
import statsmodels
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor as vif
from statsmodels.formula.api import ols
from statsmodels.tools.eval_measures import rmse

# sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import SequentialFeatureSelector as SFS
from sklearn.preprocessing import RobustScaler

# to set the digits after decimal place
pd.options.display.float_format = '{:.5f}'.format

# suppress warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

We have used the libraries as in the snapshot above for the purpose of

a. Handling of dataset.
b. Virtualization of variables
c. Model building
d. Model evaluation

We have also defined the global variable whether to perform the following by setting the global variable to True/False.

1. Outlier Treatment
2. Feature Scaling
3. Feature Transformation
4. Elimination of Feature having Multicollinearity (Based on VIF).

```
[2] #Global Variables
    executeOutliers=False
    doScaling=True
    doTransformation=True
    multicollenarityFeatureElimination=False # Based on VIF.
```

In this assessment, we are not going to remove outliers as it removes more observation. Refer section 5.5.4 for more details

Since the expectation of this assessment to have full model with all features, we are not going to eliminate any feature.

4 Business Understanding

India's used-vehicle industry is currently transitioning from an unorganized setup - where transactions happen via roadside garage mechanics, small brokers and between car owners - to an organized system with more players entering the market.

CarDekho.com is India's leading car search venture that helps users buy cars that are right for them. Its website and app carry rich automotive content such as expert reviews, detailed specs and prices, comparisons as well as videos and pictures of all car brands and models available in India. The company has tie-ups with many auto manufacturers, more than 4000 car dealers and numerous financial institutions to facilitate the purchase of vehicles.

The expectation of CarDekho.com to quote the selling price for the used vehicle request received to them based on the Machine Learning technique. The company has the details of their past resale details of the used vehicle. The dataset used here is the dataset of CarDekho.

5 Data Understanding

5.1 Collect initial data

We have collected the "download.csv" contains information about sold used vehicle details. The dataset is loaded into tool using the pandas method read_csv().

```
[4] resale_vehicle_df=pd.read_csv("https://github.com/akdiwahar/dataset/raw/main/SRM/MLSR/CT2/download.csv",index_col=0)

[5] resale_vehicle_df.head(5)
```

Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
ritz	2014	3.35000	5.59000	27000	Petrol	Dealer	Manual	0
sx4	2013	4.75000	9.54000	43000	Diesel	Dealer	Manual	0
ciaz	2017	7.25000	9.85000	6900	Petrol	Dealer	Manual	0
wagon r	2011	2.85000	4.15000	5200	Petrol	Dealer	Manual	0
swift	2014	4.60000	6.87000	42450	Diesel	Dealer	Manual	0

The data is loaded to dataframe variable "resale_vehicle_df"

5.2 Describe the data

Data in the dataset has the below information.

1. Car_Name: Name of the Vehicle. <Descriptive Data>

<u>Independent Variable/Features</u>

2. YearThis: year in which the car was bought. <Numerical Data>
3. Present_Price: current ex-showroom price of the car (in lakhs). <Numerical Data>
4. Kms_Driven: distance completed by the car in km. <Numerical Data>
5. Fuel_Type: fuel type of the car. <Categorical Data>
6. Seller_Type: defines whether the seller is a dealer or an individual. <Categorical Data>
7. Transmission: defines whether the car is manual or automatic. <Categorical Data>
8. Owner: defines the number of owners the car has previously had. <Categorical Data>

Response Variable/Target Variable/Dependent Variable.

9. Selling_Price: price the owner wants to sell the car at (in lakhs) (response variable)

5.2.1 Dataset

As per the provided dataset,

- We have received 301 records.
- We have received the parameters as stated

5.2.2 Initial data Analysis

```
[6] resale_vehicle_df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 301 entries, ritz to brio
Data columns (total 8 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
 4   Fuel_Type      301 non-null    object
 5   Seller_Type    301 non-null    object
 6   Transmission   301 non-null    object
 7   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(3)
memory usage: 21.2+ KB
```

The info() method on dataframe gives the details of

a. Number of variables.
b. Datatype of each variable
c. Number of Non-Null record for each variable.
d. Number of records in the dataset.

As per the data loaded, we have the following observation or inferences

a. We have received 301 Observations.
b. We have received 8 parameter/variable.
c. We have used index as Car Name.
d. We have Selling Price which is the predicting variable (target variable) .
e. We have no null entries

5.3 Five Point Summary of Numerical Data.

The describe method() on dataframe gives the five point summary(min, max, mean, median(50%) and standard deviation).

```
[7] resale_vehicle_df.describe()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.00000	301.00000	301.00000	301.00000	301.00000
mean	2013.62791	4.66130	7.62847	36947.20598	0.04319
std	2.89155	5.08281	8.64412	38886.88388	0.24791
min	2003.00000	0.10000	0.32000	500.00000	0.00000
25%	2012.00000	0.90000	1.20000	15000.00000	0.00000
50%	2014.00000	3.60000	6.40000	32000.00000	0.00000
75%	2016.00000	6.00000	9.90000	48767.00000	0.00000
max	2018.00000	35.00000	92.60000	500000.00000	3.00000

5.4 Summarize observations for categorical variables

The describe method() with exclude=np.number options provides the details of categorial parameters.

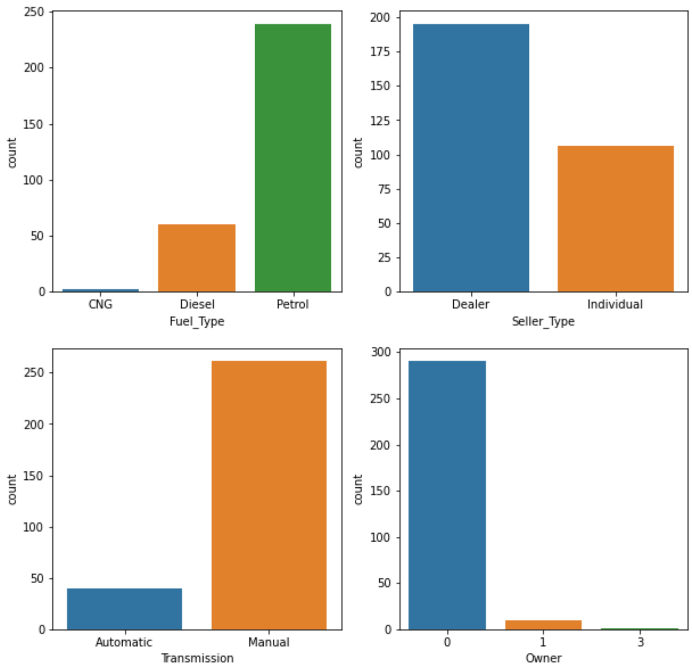
```
[10] resale_vehicle_df.describe(exclude=np.number)
```

	Fuel_Type	Seller_Type	Transmission	Owner
count	301	301	301	301
unique	3	2	2	3
top	Petrol	Dealer	Manual	0
freq	239	195	261	290

```
[15] for eachFeature in resale_vechile_df.select_dtypes(exclude=np.number).columns:
       print(eachFeature,":", list(resale_vechile_df[eachFeature].unique()))

Fuel_Type : ['Petrol', 'Diesel', 'CNG']
Seller_Type : ['Dealer', 'Individual']
Transmission : ['Manual', 'Automatic']
Owner : [0, 1, 3]
```

```
[16] pltcount=1
     plt.figure(figsize=(10,10))
     for eachFeature in resale_vechile_df.select_dtypes(exclude=np.number).columns:
       data=resale_vechile_df[eachFeature].value_counts().to_frame().reset_index()
       data.columns=[eachFeature,"count"]
       plt.subplot(2,2,pltcount)
       pltcount+=1
       sns.barplot(x=data[eachFeature],y=data["count"])
```



Inference:

1. There are 4 categorical variables in the dataset as below and the possible values on each variable.
    a. Fuel_Type : ['Petrol', 'Diesel', 'CNG']
    b. Seller_Type : ['Dealer', 'Individual']
    c. Transmission : ['Manual', 'Automatic']
    d. Owner : [0, 1, 3]
2. As seen from the visualization, there values in each variable is not equally distributed.

5.5 Check for defect in dataset

5.5.1 Missing Values and null value check.

The null values or missing value is identified with the method isnull() when applied on the dataframe. sum() method on top of null() gives the summation of null values.

Null Check

```
[14] resale_vehicle_df.isnull().sum()

Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

Inference:

1. There is no null values in the dataset.

5.5.2 Date to Age Conversion

Year column has the year on which the used vehicle was sold. Normally, it is the better approach, we always convert the date format to age.

Feature - Year Treatment

```
[15] resale_vehicle_df["Age"]=2022-resale_vehicle_df["Year"]
     resale_vehicle_df.drop("Year",axis=1,inplace=True)
```

```
[16] resale_vehicle_df.head(2)
```

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
Car_Name								
ritz	3.35000	5.59000	27000	Petrol	Dealer	Manual	0	8
sx4	4.75000	9.54000	43000	Diesel	Dealer	Manual	0	9

5.5.3 Duplicate removal Treatment

Duplicate Removal

```
[17] resale_vehicle_df[resale_vehicle_df.duplicated(keep=False)]
```

Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
ertiga	7.75000	10.79000	43000	Diesel	Dealer	Manual	0	6
ertiga	7.75000	10.79000	43000	Diesel	Dealer	Manual	0	6
fortuner	23.00000	30.61000	40000	Diesel	Dealer	Automatic	0	7
fortuner	23.00000	30.61000	40000	Diesel	Dealer	Automatic	0	7

```
[18] resale_vehicle_df=resale_vehicle_df.drop_duplicates()
```

We see two observations are duplicate. We have removed the duplicates.

5.5.4 Outliers

The below code visualizes the outlier of each parameter in the given dataset using box plot.

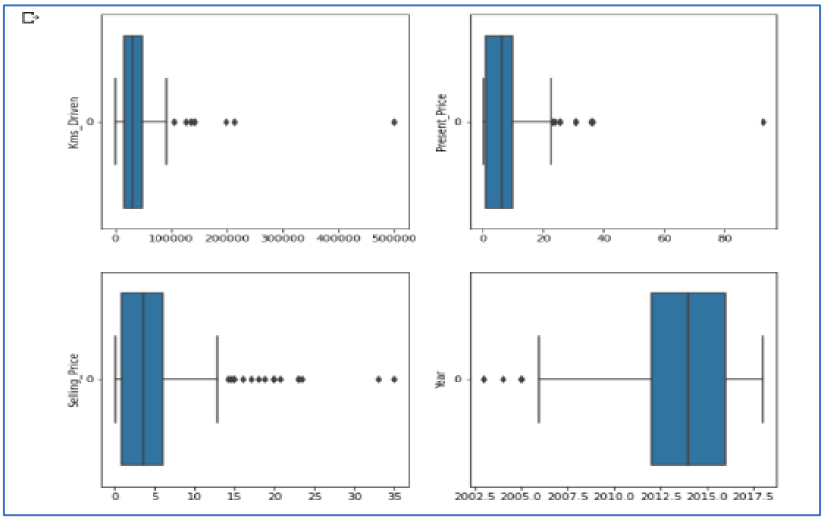
```
[19] columns=resale_vehicle_df.select_dtypes(include=np.number).columns

     pltcount=len(columns)
     plt.figure(figsize=(10,10))
     for eachFeature in columns:

       plt.subplot(2,2,pltcount)
       pltcount-=1
       plt.ylabel(eachFeature)
       sns.boxplot( data=resale_vehicle_df[eachFeature] ,orient="h")

     plt.show()
```

The output of the above code snippet is as below



Inference:

1. The parameter Kms_Driven, Present_Price, Selling_Price and Age has Outliers.

Outlier Treatment

The below function helps to remove the outlier data based on IQR.

```
[97] def outliers(df):
    indexesToRevome=[]
    index_name=df.index.name
    df=df.reset_index()
    columns=df.select_dtypes(include=np.number).columns

    for eachCol in columns:
      print("Processing ",eachCol)
      q1=df[eachCol].quantile(0.25)
      q3=df[eachCol].quantile(0.75)
      IQR=q3-q1
      whisherLeft=q1-IQR*1.5
      whisherRight=q3+IQR*1.5
      indexes=df.index[(whisherLeft>df[eachCol]) | (whisherRight<df[eachCol] )].tolist()
      if (len(indexes)>0):
        print("Index in outlier for ",eachCol," is ", indexes, "\n")
        indexesToRevome=indexesToRevome + indexes
    #noOfRecords=dfturnout[dfturnout[eachCol]>whisherRight and dfturnout[eachCol]<whisherLeft].size()
    #print(eachCol,q1,q3,whisherLeft,whisherRight, noOfRecords)
    #Unique entries
    indexesToRevome= list(set(indexesToRevome))
    print("Finalized index to remove from source :", indexesToRevome)

    df=df.drop(index=indexesToRevome)
    df=df.set_index(index_name)
    return df,indexesToRevome
```

Inference: When we treat the outliers, we see below 38 observations are identified as outliers(iloc rows).

```
{157, 158, 162, 164, 37, 166, 39, 40, 170, 173, 45, 50, 51, 52,
53, 179, 54, 59, 189, 62, 63, 64, 66, 196, 69, 72, 77, 79, 80,
82, 84, 85, 86, 92, 93, 96, 97, 251}
```

Note: We have included the outliers in this assessment. Also, we see a better model compared with outlier than without outliers. Also, we see very uneven distribution of features Age, Present_Price, Selling_Price and Kms_Driven when we remove outlier. i.e multiple peak when plotting the density plot.

6 Data Preparation (Exploratory Data Analysis)

6.1 Ignore insignificant feature.

Car Name is the insignificant parameter for model building. As this parameter is used as an row index, no special treatment is needed.

6.2 Split Numerical and Categorical Dataset

We are splitting the data into numerical and categorical dataset. The data is stored in vehicle_num and vehicle_cat.

```
[25] vehicle_num=resale_vehicle_df.select_dtypes(include=[np.number])
     vehicle_cat=resale_vehicle_df.select_dtypes(exclude=[np.number])
```

```
[26] vehicle_cat.head(2)
```

Car_Name	Fuel_Type	Seller_Type	Transmission	Owner
ritz	Petrol	Dealer	Manual	0
sx4	Diesel	Dealer	Manual	0

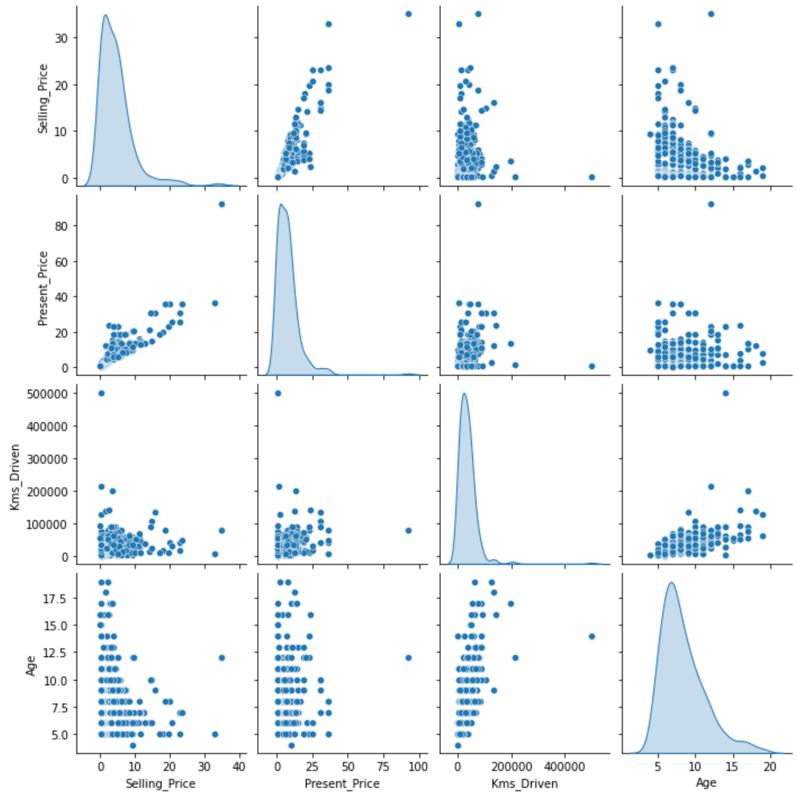
```
[27] vehicle_num.head(2)
```

Car_Name	Selling_Price	Present_Price	Kms_Driven	Age
ritz	3.35000	5.59000	27000	8
sx4	4.75000	9.54000	43000	9

6.3 Distribution of Variables

6.3.1 Numerical Parameters

```
car_n=vehicle_num.reset_index().drop("Car_Name",axis=1)
sns.pairplot(car_n.reindex(), diag_kind='kde')
plt.show()
```

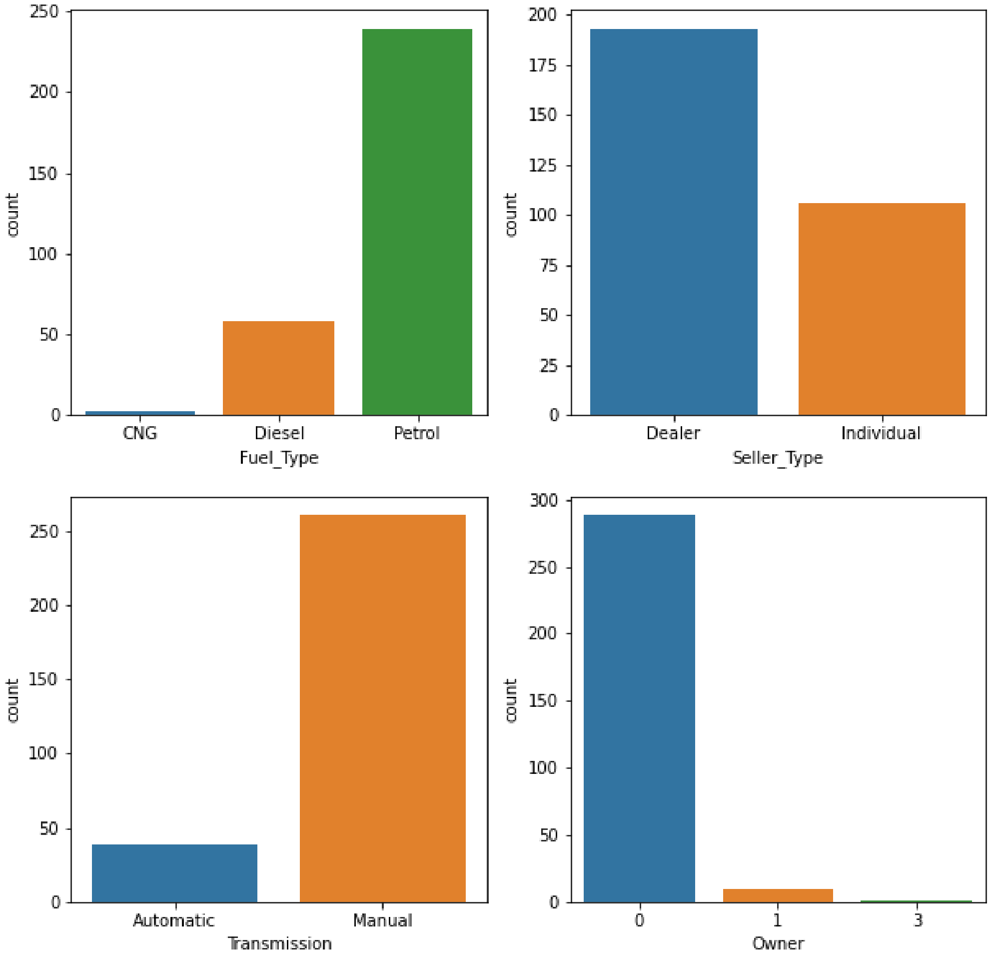


Inference:

1. We could see the pattern between Present_Price and selling_price
2. We could see the distribution for the parameter Selling_price, Present_Price and Kms_Driven are right skewed

6.3.2 Categorical Parameters

```
[223] pltcount=1
      plt.figure(figsize=(10,10))
      for eachFeature in vehicle_cat.columns:
        data=vehicle_cat[eachFeature].value_counts().to_frame().reset_index()
        data.columns=[eachFeature,"count"]
        plt.subplot(2,2,pltcount)
        pltcount+=1
        sns.barplot(x=data[eachFeature],y=data["count"])
```



Inference:

1. We could see the uneven distribution for the categorical parameter Fuel_Type, Seller_Type, Transmission and Owner.

6.4 Data Encoding for Categorical Dataset

From the dataset, it is observed that we have four categorical variable

a. Fuel_Type
b. Seller_Type
c. Transmission
d. Owner.

```
[30] vehicle_cat=pd.get_dummies(vehicle_cat,  drop_first=True, columns=["Transmission","Seller_Type"])

[31] vehicle_cat["Owner"]=vehicle_cat["Owner"].astype("uint8")

[32] vehicle_cat=pd.get_dummies(vehicle_cat, columns=["Fuel_Type"])

[33] #Fuel-Type Gas has only 2 observation, hence do not want to remove
     vehicle_cat.drop("Fuel_Type_Petrol",axis=1,inplace=True)
```

We have done the encoding as below by executing the above code.

<table>
<tr><td>Categorical Variable</td><td>Encoding done.</td></tr>
<tr><td>Owner</td><td>Already data has 0,1,2,3… As it is ordinal data, we keep the entry as it is and converted to integer</td></tr>
<tr><td>Transmission</td><td>Dummy Encoding. After encoding we have the parameter
<code>Transmission_Manual. 1-> Manual , 0 -> Automatic</code></td></tr>
<tr><td>Seller_Type</td><td>Dummy Encoding. After encoding we have the parameter
<code>Seller_Type_Individual. 0-> Dealer, 1->Individual</code></td></tr>
<tr><td>Fuel_Type</td><td>Dummy Encoding. But we removed Petrol instead of Gas Fuel Type.
<table>
<tr><td>Fuel_Type_CNG</td><td>Fuel_Type_Diesel</td><td>Fuel_Type</td></tr>
<tr><td>0</td><td>0</td><td>Petrol</td></tr>
<tr><td>0</td><td>1</td><td>Diesel</td></tr>
<tr><td>1</td><td>0</td><td>CNG</td></tr>
</table>
</td></tr>
</table>

6.5 Merge Numerical and Categorial Data

We finalize the numerical and categorical data into the one dataset vehicle.

Merge Numerical and Categorical

```
[35] vehicle=pd.concat([vehicle_num,vehicle_cat],axis=1)
```

6.6 Feature Transformation.

```
skewKurt= pd.DataFrame()
doTransformation=True
if (doTransformation):
  vehicle_before_Transformation=vehicle.copy()
  skewKurt["Skew_Before"]=vehicle.skew()
  skewKurt["Kurt_Before"]=vehicle.kurt()

  for eachCol in vehicle.columns:
    vehicle[eachCol]=np.power(vehicle[eachCol],1/3)

  skewKurt["Skew_After_cubic-rt"]=vehicle.skew()
  skewKurt["Kurt_After_cubic-rt"]=vehicle.kurt()
else:
  skewKurt["Skew"]=vehicle.skew()
  skewKurt["Kurt"]=vehicle.kurt()
  print("Transformation Not Enabled")
skewKurt
```

	Skew_Before	Kurt_Before	Skew_After_cubic-rt	Kurt_After_cubic-rt
Selling_Price	2.53652	9.48209	0.28263	-0.26446
Present_Price	4.18689	33.19508	0.33905	0.12346
Kms_Driven	6.41813	68.13042	0.36239	2.33334
Age	1.23688	1.50724	0.71008	0.11682
Owner	7.59060	72.82124	5.09532	24.87865
Transmission_Manual	-2.20577	2.88468	-2.20577	2.88468
Seller_Type_Individual	0.61133	-1.63727	0.61133	-1.63727
Fuel_Type_CNG	12.16511	146.97300	12.16511	146.97300
Fuel_Type_Diesel	1.55566	0.42287	1.55566	0.42287

Inference:

1. We have done cubic root for the features.
2. After the transformation, we skew and kurt reduced on numerical variable.
3. Effect of Transformation to Categorical variable is not seen. Since $1^3 = 1^{1/3} = 1$ and $0^3 = 0^{1/3} = 0$
   This is due to imbalanced data on the categorical variable

6.7 Feature Scaling.

Since the numerical data are in different scale, we are doing here with Robust scaling. The robust scaler subtracts feature values by their median and then divides by its IQR. It is best scaling when we have outliers.

Feature Scaling

```
[44] scaler = RobustScaler()

     vehicle_before_scaling=vehicle.copy()
     if (doScaling):
       vehicle_scaled=scaler.fit_transform(vehicle)
       vehicle_scaled = pd.DataFrame(vehicle_scaled, index=vehicle.index, columns=vehicle.columns)
       vehicle=vehicle_scaled
     else:
       print("Scaling Not Enabled")
     vehicle.head(5)
```

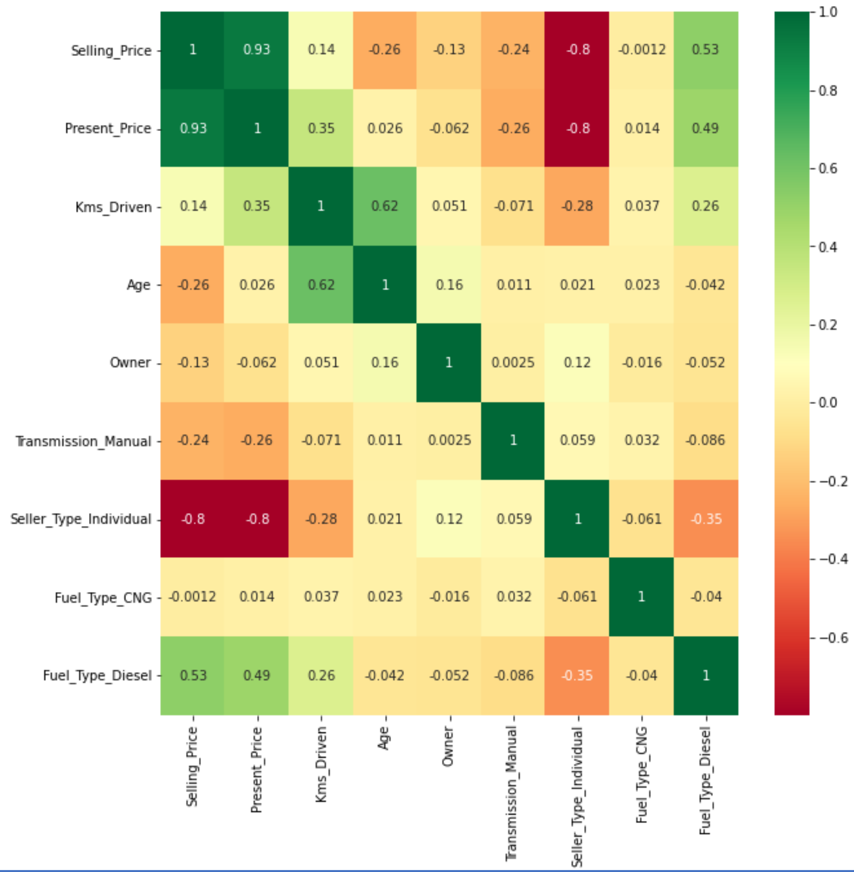
Car_Name	Selling_Price	Present_Price	Kms_Driven	Age	Owner	Transmission_Manual	Seller_Type_Individual	Fuel_Type_CNG	Fuel_Type_Diesel
ritz	-0.02695	-0.04852	-0.14687	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
sx4	0.18530	0.27191	0.27609	0.23742	0.00000	0.00000	0.00000	0.00000	1.00000
ciaz	0.47770	0.29295	-1.06792	-0.85980	0.00000	0.00000	0.00000	0.00000	0.00000
wagon r	-0.11714	-0.20381	-1.21184	0.66401	0.00000	0.00000	0.00000	0.00000	0.00000
swift	0.16475	0.06837	0.26348	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000

6.8 Correlation between parameters

We used the below code for generating the cross correlation across the feature including the target feature(selling price).

```
[45] plt.figure(figsize=(10,10))
     sns.heatmap(vehicle.corr(),annot=True,cmap="RdYlGn")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f77d7fffeb0>
```



Inference:

1. We could see a good correlation of selling price with current price(0.93) and seller_type_individual(-0.8)
2. We could see the correlation with Fuel_Type_Diesel(>0.5), Kms_Driven(0.14), Transmission_Manual(-0.24) , Owner(-0.13) and age(0.26) against selling price.

6.9 Split the data frame to Dependent and Independent Feature.

Dependent and Independent Features

```
[43] X=vehicle.drop("Selling_Price",axis=1)
     y=vehicle["Selling_Price"]
```

We store them the independent feature in X and dependent feature in y.

6.10 Adding Constant

Since we are using the stats model, the const co-efficient(y-intercept) to be added in the dataset.

```
[48] if (multicollenarityFeatureElimination):
       X1=X[features]
     else:
       X1=X
     X1 = sm.add_constant(X1)
```

```
[49] X1.head()
```

Car_Name	const	Present_Price	Kms_Driven	Age	Owner	Transmission_Manual	Seller_Type_Individual	Fuel_Type_CNG	Fuel_Type_Diesel
ritz	1.00000	-0.04852	-0.14687	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
sx4	1.00000	0.27191	0.27609	0.23742	0.00000	0.00000	0.00000	0.00000	1.00000
ciaz	1.00000	0.29295	-1.06792	-0.85980	0.00000	0.00000	0.00000	0.00000	0.00000
wagon r	1.00000	-0.20381	-1.21184	0.66401	0.00000	0.00000	0.00000	0.00000	0.00000
swift	1.00000	0.06837	0.26348	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000

6.11 Train Test Split

The package sklearn.model_selection provides the function train_test_split() to split the data into train set and test set. The trainset is used to train the model and testset is used to evaluate the model.

The below code converts the dependent feature(X) and independent variable(y) into training set (80%) and test set (20%).

Test-Train Split

```
[ ] #splitting train and test data
    X_train,X_test,Y_train,Y_test=train_test_split(X1,y,test_size=0.2,random_state=6)
```

Since the random_state is marked with the static value of 6, we get the same split of train and test data for any time execution of train_test_split() method.

7 Model Building

7.1 Model Building – Full Model with all Features

Model Building with Complete Features

```
[50] lin_reg_model=sm.OLS(Y_train,X_train)
     model = lin_reg_model.fit()
```

7.2 Model Summary

```
[52] model.summary()
```

OLS Regression Results

Dep. Variable:	Selling_Price	**R-squared:**	0.963
Model:	OLS	**Adj. R-squared:**	0.962
Method:	Least Squares	**F-statistic:**	746.6
Date:	Sun, 11 Dec 2022	**Prob (F-statistic):**	6.75e-160
Time:	05:36:00	**Log-Likelihood:**	157.17
No. Observations:	239	**AIC:**	-296.3
Df Residuals:	230	**BIC:**	-265.1
Df Model:	8		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.0496	0.013	3.769	0.000	0.024	0.076
Present_Price	0.8696	0.026	33.408	0.000	0.818	0.921
Kms_Driven	-0.0430	0.015	-2.923	0.004	-0.072	-0.014
Age	-0.2366	0.017	-13.799	0.000	-0.270	-0.203
Owner	-0.0991	0.044	-2.245	0.026	-0.186	-0.012
Transmission_Manual	-0.0035	0.025	-0.140	0.889	-0.053	0.046
Seller_Type_Individual	-0.1666	0.030	-5.486	0.000	-0.226	-0.107
Fuel_Type_CNG	-0.0939	0.129	-0.730	0.466	-0.348	0.160
Fuel_Type_Diesel	0.1421	0.024	5.859	0.000	0.094	0.190

Omnibus:	9.141	**Durbin-Watson:**	2.061
Prob(Omnibus):	0.010	**Jarque-Bera (JB):**	9.875
Skew:	-0.373	**Prob(JB):**	0.00717
Kurtosis:	3.660	**Cond. No.**	17.6

Inference:

1. Prob(F-stat) <0.05 indicates the model is significant.
2. R-squared is 0.963 is close to 1. The model can predict more accurate.
3. R-squared-adj is 0.962 is close to 1. The model can predict more accurate.
4. Degree of Freedom of the model is 8.
5. No of Residuals is 239.
6. Durbin-Watson is near to 2 indicates no auto-correlation
7. Condition No <100 indicates no multi-collinearity between the dependent variables.
8. Skew is -0.3 indicates not much skewness in the dataset.
9. Kurt is 3.6 indicate lyptokurtic. It is preferred to be <3.
10. P(JB)<0.05 indicates the data is not normally distributed.
11. Parameter Fuel_Type_CNG, Transmission_Manual are not significant which has p-value> 0.05.

7.3 Linear Equation

The equation of the model is given by

Selling Price= (0.04959) * const + (0.8696) * Present_Price + (-0.043) * Kms_Driven + (-0.23655) * Age + (-0.09915) * Owner + (-0.00353) * Transmission_Manual + (-0.16664) * Seller_Type_Individual + (-0.09393) * Fuel_Type_CNG + (0.14212) * Fuel_Type_Diesel

Note: the parameter needs to be transformed and scaled before applying to the model.

Beta coefficient of the model is as below

```
[59] model.params

const                     0.04959
Present_Price             0.86960
Kms_Driven               -0.04300
Age                      -0.23655
Owner                    -0.09915
Transmission_Manual      -0.00353
Seller_Type_Individual   -0.16664
Fuel_Type_CNG            -0.09393
Fuel_Type_Diesel          0.14212
dtype: float64
```

7.4 Model Prediction

```
[58] Y_predict=model.predict(X_test)
```

Model is predicted with Test data with the predict method on the model.

7.5 Impact of Fuel Type with selling price.

There are three fuel Type in the given data. They are

1. Petrol
2. Diesel
3. CNG

We have 2 records of CNG fuel Type and 58 records of Diesel fuel Type out of 299 observations.

The model is build considering the Petrol as base for the used vehicle and the selling price is impacted based on the Diesel or Gas fuel type.

As per the model, the impact of the Fuel Type is as below

```
(-0.09393) * Fuel_Type_CNG + (0.14212) * Fuel_Type_Diesel
```

```
[58] model.pvalues

const                    0.00021
Present_Price            0.00000
Kms_Driven               0.00381
Age                      0.00000
Owner                    0.02572
Transmission_Manual      0.88891
Seller_Type_Individual   0.00000
Fuel_Type_CNG            0.46619
Fuel_Type_Diesel         0.00000
dtype: float64
```

```
[59] model.params

const                     0.04959
Present_Price             0.86960
Kms_Driven               -0.04300
Age                      -0.23655
Owner                    -0.09915
Transmission_Manual      -0.00353
Seller_Type_Individual   -0.16664
Fuel_Type_CNG            -0.09393
Fuel_Type_Diesel          0.14212
dtype: float64
```

When the vehicle is identified as CNG, then the selling price is reduced by 0.09393 per unit change in Fuel_Type_CNG. Since Fuel_Type_CNG is categorical, we always get either 0 or 1. On the other hand, the p-value of Fuel_Type_CNG> 0.05, indicated the parameter/feature(CNG) is not significant.

When the vehicle is identified as Diesel, then the selling price is increased by 0.14212 per unit change in Fuel_Type_Diesel. Since Fuel_Type_ Diesel is categorical, we always get either 0 or 1. On the other hand, the p-value of Fuel_Type_ Diesel < 0.05, indicated the parameter is significant.

As per the coefficient received from the OLS model,

1. when the fuel type is Diesel, selling price is increased by 0.14212 for each change in Fuel_Type_Diesel .

2. When the fuel type is CNG, the selling price is decreased by 0.09393 for each change in Fuel_Type_CNG

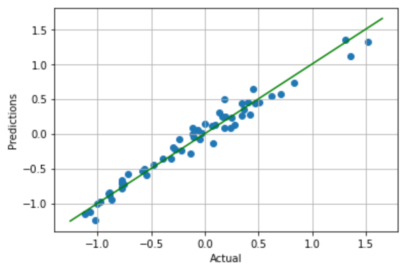
3. When the fuel type is Petrol, there is no impact on selling price as the model assumption is that Fuel Type Petrol is considered by default and selling prices varies based on other fuel Type.

7.6 Model Evaluation

```
rmseval=rmse(Y_test,Y_predict)
print("RMSE:",rmseval)

plt.scatter(Y_test,Y_predict)
left, right = plt.xlim()
bottom, top = plt.ylim()
xpoints = ypoints = np.linspace(left, right)
plt.plot(xpoints, ypoints, color="g", label="xlim")
plt.grid()
plt.xlabel('Actual')
plt.ylabel('Predictions')
plt.show()
```

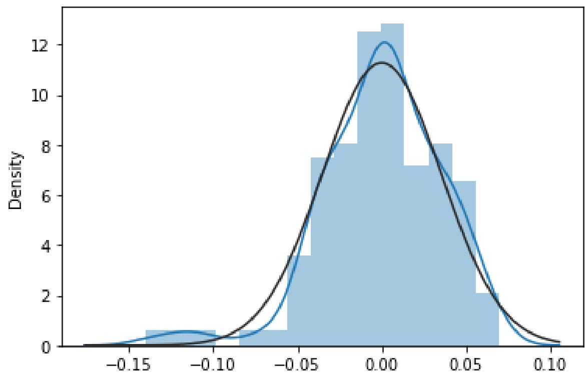
```
RMSE: 0.11060612208969292
```



Impression: When we see the scatter plot which is plotted between the actual selling price and predicted selling price, we see the plot around 45 degree line(Green) indicates the predications are more accurate.

```
[265]
    residual=model.resid
    norm.fit(residual)
    sns.distplot(residual,fit=norm)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1a1db4e280>
```



Impression: We could see the residuals are fit almost normally distributed. This indicates we extracted the most of the patterns from the dependent variable.

7.7 Regress with each Feature with Selling Price – Impact of each feature and its combination with Selling Price in the model.

```
[73] model.pvalues.to_frame().sort_values(by=0,ascending=True)
```

	0
Present_Price	0.00000
Age	0.00000
Fuel_Type_Diesel	0.00000
Seller_Type_Individual	0.00000
const	0.00021
Kms_Driven	0.00381
Owner	0.02572
Fuel_Type_CNG	0.46619
Transmission_Manual	0.88891

As per the model, based on the pvalues, we could observe that

1. Transmission_Manual and Fuel_Type_CNG > 0.05 indicates the feature is not significant.
2. Present_Price, Age, Fuel_Type_Diesel, Seller_Type_Individual, Kms_Driven and owner are significant features.

```
[76] model.params.to_frame().sort_values(by=0, ascending=False)
```

	0
Present_Price	0.86960
Fuel_Type_Diesel	0.14212
const	0.04959
Transmission_Manual	-0.00353
Kms_Driven	-0.04300
Fuel_Type_CNG	-0.09393
Owner	-0.09915
Seller_Type_Individual	-0.16664
Age	-0.23655

As per the model, based of the coefficient we could observe that

1. The present price and Fuel_Type_Diesel increases the selling price.
    a. For each unit change in Present_Price the selling price is increased by the factor of 0.8696042811265295
    b. For each unit change in Fuel_Type_Diesel the selling price is increased by the factor of 0.142123190517839
2. The other Feature reduces the selling prices when it varies.
    a. For each unit change in Transmission_Manual the selling price is decreased by factor of 0.00352844765943396
    b. For each unit change in Kms_Driven the selling price is decreased by factor of 0.04300254119495564
    c. For each unit change in Fuel_Type_CNG the selling price is decreased by factor of 0.09393401257871288

d. For each unit change in Owner the selling price is decreased by factor of 0.09914884369041846
e. For each unit change in Seller_Type_Individual the selling price is decreased by factor of 0.16663861101564342
f. For each unit change in Age the selling price is decreased by -0.23655334537187575

To confirm the impact, Let us regress the each features or combination against selling price. In order to build the regression of dependent and independent feature, we use the below code and we executed them with iteration.

```
#Iteration 1
feature_regress=list(X.columns)
regress=pd.DataFrame()
selected_Feature=[]
for eachFeature in feature_regress:
  final_selected_feature=selected_Feature.copy()
  final_selected_feature.append(eachFeature)
  lin_reg_model=sm.OLS(Y_train,X_train[final_selected_feature])
  model2 = lin_reg_model.fit()
  y_predict=model2.predict(X_test[final_selected_feature])
  rmseval=rmse(Y_test,y_predict)
  regress["selling price ~ "+str(final_selected_feature)]=pd.DataFrame([model2.rsquared, model2.rsquared_adj,model2.f_pvalue,rmseval])

regress=regress.T
regress.columns=["rsquared","rsquared_adj","model p","rmse"]
regress.sort_values(by=["rmse","rsquared_adj"])
```

The above code snippet regress each feature with the selling price. i.e (Selling Price ~ each feature). In case we need to regress selling price with more than 1 feature, the same code is used with feature added in the selected_Features.

<table>
<tr><th>Iteration</th><th>Outcome/Result</th><th>Inference</th></tr>
<tr><td>1</td><td>
<table>
<tr><th></th><th>rsquared</th><th>rsquared_adj</th><th>model p</th><th>rmse</th></tr>
<tr><td>selling price ~ ['Present_Price']</td><td>0.87006</td><td>0.86952</td><td>0.00000</td><td>0.21964</td></tr>
<tr><td>selling price ~ ['Seller_Type_Individual']</td><td>0.46395</td><td>0.46170</td><td>0.00000</td><td>0.42423</td></tr>
<tr><td>selling price ~ ['Fuel_Type_Diesel']</td><td>0.19194</td><td>0.18854</td><td>0.00000</td><td>0.58680</td></tr>
<tr><td>selling price ~ ['Transmission_Manual']</td><td>0.01923</td><td>0.01511</td><td>0.03176</td><td>0.60705</td></tr>
<tr><td>selling price ~ ['Age']</td><td>0.07148</td><td>0.06758</td><td>0.00003</td><td>0.61629</td></tr>
<tr><td>selling price ~ ['Owner']</td><td>0.01506</td><td>0.01093</td><td>0.05761</td><td>0.62080</td></tr>
<tr><td>selling price ~ ['Kms_Driven']</td><td>0.01677</td><td>0.01264</td><td>0.04506</td><td>0.62239</td></tr>
<tr><td>selling price ~ ['Fuel_Type_CNG']</td><td>0.00009</td><td>-0.00411</td><td>0.88062</td><td>0.63494</td></tr>
</table>
</td><td>Present_Price gives the higher R^2_{adj} values compared with the other feature. Also the RMSE is around 0.2.

Present_Price shows the higher impact on the selling price.

We select the Present_Price</td></tr>
<tr><td>2</td><td>
<table>
<tr><th></th><th>rsquared</th><th>rsquared_adj</th><th>model p(f-stat)</th><th>rmse</th></tr>
<tr><td>selling price ~ ['Present_Price', 'Age']</td><td>0.94923</td><td>0.94880</td><td>0.00000</td><td>0.12633</td></tr>
<tr><td>selling price ~ ['Present_Price', 'Kms_Driven']</td><td>0.90950</td><td>0.90873</td><td>0.00000</td><td>0.17404</td></tr>
<tr><td>selling price ~ ['Present_Price', 'Fuel_Type_Diesel']</td><td>0.87837</td><td>0.87734</td><td>0.00000</td><td>0.21482</td></tr>
<tr><td>selling price ~ ['Present_Price', 'Seller_Type_Individual']</td><td>0.87107</td><td>0.86999</td><td>0.00000</td><td>0.21806</td></tr>
<tr><td>selling price ~ ['Present_Price', 'Transmission_Manual']</td><td>0.87017</td><td>0.86907</td><td>0.00000</td><td>0.22134</td></tr>
<tr><td>selling price ~ ['Present_Price', 'Owner']</td><td>0.87565</td><td>0.87460</td><td>0.00000</td><td>0.22217</td></tr>
<tr><td>selling price ~ ['Present_Price', 'Fuel_Type_CNG']</td><td>0.87061</td><td>0.86952</td><td>0.00000</td><td>0.22271</td></tr>
</table>
</td><td>Age along with Present_Price improves the R^2_{adj} values compared with the other feature in combination with Present_Price. Improvement in R^2_{adj} from</td></tr>
</table>

0.86852 to 0.94880 when Age is included.

We select the **Present_Price, Age**

3

	rsquared	rsquared_adj	f_pvalue	rmse
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel']	0.95546	0.95489	0.00000	0.11615
selling price ~ ['Present_Price', 'Age', 'Kms_Driven']	0.94981	0.94917	0.00000	0.12554
selling price ~ ['Present_Price', 'Age', 'Seller_Type_Individual']	0.94940	0.94876	0.00000	0.12610
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_CNG']	0.94926	0.94861	0.00000	0.12637
selling price ~ ['Present_Price', 'Age', 'Transmission_Manual']	0.94923	0.94859	0.00000	0.12643
selling price ~ ['Present_Price', 'Age', 'Owner']	0.95013	0.94950	0.00000	0.13061

Fuel_Type_Diesel along with Present_Price,Age improves the R^2_{adj} values compared with the other feature in combination with Present_Price, Age.

Improvement in R^2_{adj} from 0.94880 to 0.95489 when Fuel_Type_Diesel is included.

We select the **Present_Price, Age, Fuel_Type_Diesel**

4

	rsquared	rsquared_adj	f_pvalue	rmse
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual']	0.95825	0.95754	0.00000	0.11172
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Kms_Driven']	0.95642	0.95568	0.00000	0.11385
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Fuel_Type_CNG']	0.95548	0.95472	0.00000	0.11617
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Transmission_Manual']	0.95569	0.95493	0.00000	0.11820
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Owner']	0.95680	0.95606	0.00000	0.12136

Improvement in R^2_{adj} from 0.95489 to 0.95754 when Seller_Type_Individual is included.

We select the **Present_Price, Age, Fuel_Type_Diesel, Seller_Type_Individual**

5

	rsquared	rsquared_adj	f_pvalue	rmse
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Kms_Driven']	0.95998	0.95913	0.00000	0.10712
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Transmission_Manual']	0.95832	0.95743	0.00000	0.11014
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Fuel_Type_CNG']	0.95827	0.95737	0.00000	0.11169
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Owner']	0.95893	0.95806	0.00000	0.11565

Improvement in R^2_{adj} from 0.95754 to 0.95913 when Kms_Driven is included.

We select the **Present_Price, Age, Fuel_Type_Diesel, Seller_Type_Individual, Kms_Driven**

6

	rsquared	rsquared_adj	f_pvalue	rmse
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Kms_Driven', 'Transmission_Manual']	0.96006	0.95904	0.00000	0.10539
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Kms_Driven', 'Fuel_Type_CNG']	0.96000	0.95897	0.00000	0.10720
selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Kms_Driven', 'Owner']	0.96072	0.95970	0.00000	0.11106

Improvement in R^2_{adj} from 0.95913 to 0.95970 when Owner is included and adding value to the model

The other feature (Transmission and Fuel_Type_CNG) reduces the R^2_{adj} which affects the model.

<table>
<tr><td></td><td></td><td>We select the Present_Price, Age, Fuel_Type_Diesel, Seller_Type_Individual, Kms_Driven,Owner</td></tr>
<tr><td>7</td><td><table><tr><th></th><th>rsquared</th><th>rsquared_adj</th><th>f_pvalue</th><th>rmse</th></tr><tr><td>selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Kms_Driven', 'Owner', 'Transmission_Manual']</td><td>0.96079</td><td>0.95961</td><td>0.00000</td><td>0.10941</td></tr><tr><td>selling price ~ ['Present_Price', 'Age', 'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Kms_Driven', 'Owner', 'Fuel_Type_CNG']</td><td>0.96074</td><td>0.95955</td><td>0.00000</td><td>0.11118</td></tr></table></td><td>Decline in R^2_{adj} when further features is added to model, though R^2 is improved.

This implies, Transmission_Manual and Fuel_Type_CNG is not significant to the model.</td></tr>
</table>

Impression:

1. Features Present_Price(Most Significant), Age, Fuel_Type_Diesel, Seller_Type_Individual, Kms_Driven,Owner(Least Significant) are significant in predicting the selling price.
2. Transmission_Manual and Fuel_Type_CNG is not significant to the model.
3. Impact(as stated in this section earlier):
    a. For each unit change in Present_Price the selling price is increased by factor of 0.8696042811265295
    b. For each unit change in Fuel_Type_Diesel the selling price is increased by factor of 0.142123190517839
    c. For each unit change in Transmission_Manual the selling price is decreased by factor of 0.00352844765943396
    d. For each unit change in Kms_Driven the selling price is decreased by factor of 0.04300254119495564
    e. For each unit change in Fuel_Type_CNG the selling price is decreased by factor of 0.09393401257871288
    f. For each unit change in Owner the selling price is decreased by factor of 0.09914884369041846
    g. For each unit change in Seller_Type_Individual the selling price is decreased by factor of 0.16663861101564342
    h. For each unit change in Age the selling price is decreased by factor of 0.23655334537187575