USED CAR PRICE PREDICTION PROJECT

Submitted by:

TAMALI SAHA

FlipRobo SME:

GULSHANA CHAUDHARY

ACKNOWLEDGMENT

I would like to express my special gratitude to Flip Robo Technologies team, who has given me this opportunity to deal with this dataset during my internship. It helped me to improve my analyzation skills. I want to express my gratitude to Ms. Gulshana Chaudhary (SME, Flip Robo) as she has helped me to get out of all the difficulties I faced while doing the project. I also want to give huge thanks to entire DataTrained team.

Bibliography:

Reference used in this project:

1. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
2. Andrew Ng Notes on Machine Learning (GitHub).
3. Different projects on Github and Kaggle.
4. towardsdatascience, Analytics Vidya's different papers on Data Science.
5. SCIKIT Learn Library Documentation.
6. Different conference papers on Recharchgate.
7. Gegic, Enis, Becir Isakovic, Dino Keco, Zerina Masetic, and Jasmin Kevric. "Car price prediction using machine learning techniques." TEM Journal 8, no. 1 (2019): 113.
8. Noor, Kanwal, and Sadaqat Jan. "Vehicle price prediction system using machine learning techniques." International Journal of Computer Applications 167, no. 9 (2017)
9. 'Used Carprice Prediction Complete Machine Larning Project', Tejashree Nawale, Published in Geek Culture.

Table of Contents

1. Introduction

1.1 Business Problem Framing

The manufacturer sets the price of a new car in the market, with some additional expenses paid by the government in the form of taxes. Customers who purchase a new car can be sure that their investment will be worthwhile. But because new cars are becoming more expensive and consumers can no longer afford to acquire them, used car sales are rising everywhere. Now, some cars are in high demand and are therefore expensive, while others are not and are therefore less expensive.

Therefore, a system that accurately assesses the value of the car utilising a range of features is urgently needed for used car price prediction. With the covid 19 impact in the market, we have seen lot of changes in the car market. The current system involves a procedure where a vendor chooses a price at random and the buyer is unaware of the car and its current market value. In actuality, neither the seller nor the price at which he ought to sell the car have any notion of the current value of the vehicle. In order to solve this issue, a model must be created that can accurately estimate a car's true price rather than just its price range.

1.2 Conceptual Background of the Domain Problem

Due to the numerous variables that affect a used automobile's market price, figuring out whether the quoted price of a used car is accurate is a difficult undertaking. The goal of this research is to create machine learning models that can precisely forecast a used car's price based on its attributes so that buyers can make educated decisions.

Because the price of a car typically depends on a variety of unique features and factors, accurate car price prediction requires specialist expertise. Usually, brand and model, age, horsepower, and mileage are the most important ones. Due to the frequent changes in fuel prices, the kind of gasoline used in the automobile and fuel consumption per mile have a significant impact on the price of a car. The price of a car will also be influenced by many aspects such as the outside colour, door count, transmission type, dimensions, safety, air conditioning, interior, and whether or not it includes navigation. We used a variety of methodologies and techniques in this study to increase the accuracy of the used car price prediction.

It is feasible to anticipate the real price of an automobile rather than just the price range of a car since regression algorithms give us a continuous value as an output rather than a categorised value.

1.3 Review of Literature

Enis Gegic et al. put forth the idea of predicting car prices using machine learning techniques.. By combining multiple machine learning methods including Support Vector Machine, Random Forest, and Artificial Neural Networks, the authors of this research suggested an ensemble model. To forecast the price of used automobiles in Bosnia and Herzegovina, they built this model using information from the website www.autopijaca.ba. Their model's accuracy is 87%.

Using machine learning techniques, Kanwal Noor and Sadaqat Jan suggested a system for predicting vehicle prices. In this study, a model to forecast car prices using multiple linear regression was suggested. By using the feature selection technique, they selected the feature that had the most impact and eliminated the others. The proposed model's prediction accuracy was nearly 98%.

In this article named 'Used Carprice Prediction Complete Machine Larning Project', Nawale will demonstrate how to train a model that will allow us to forecast automobile prices. We will put into practise the machine learning workflow that has so far discovered how to forecast the market price of a car using its qualities. The dataset we'll be using has data on numerous types of autos.

1.4 Motivation for the Problem Undertaken

The project is provided to me by Flip Robo Technologies as a part of the internship programme (Internship Batch No-31). This problem is a real world dataset. The exposure of this data gives me the opportunity to locate my skills in solving a real time problem. It is the primary motivation to solve this problem.

The data required for this project must be extracted from the internet and processed. It can be challenging to decide whether a used car is worth the asking price while viewing listings online. The goal of this research is to create machine learning models that can precisely forecast a used car's price based on its attributes so that buyers can make educated decisions. The model created in this study could be useful for online web businesses that estimate the market value of old cars.

This study aims to analyse and predicting price when using **Regression Model** like Linear Regression, Random Forest, Decision Tree, KNN, Gradient Boosting, Ada Boost Regression algorithms. Thus, the purpose of this study is to grow the knowledge of **Regression methods** in machine learning fields. Those are the different factors to undertaken the problem for study purpose.

2. Analytical Problem Framing

2.1 Mathematical/ Analytical Modelling of the Problem

The goal of this project is to predict used car price prediction. We make this dataset by web scraping from **CarDekho Website**. The algorithms are used with their own mathematical equation on background. Here the problem is in two phase.

One is: **Data Collection Phase-** We have to scrape at least 5000 used cars data from websites (Olx, cardekho, Cars24 etc.), need web scraping for this.

Another is: **Model Building Phase-** After collecting the data, we need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

This project have one big set of data. All the missing value are imputed. Then different steps of data pre-processing is performed over the dataset like data cleaning, data visualization, relationship between features with target. In model building. Final model is select based on R2 score, Mean absolute error, Root mean square error and Cross Validation score of different algorithms.

Here 7 different algorithm are used.

2.2 Data Sources and their formats

The data is used for both training and testing the model. It has 5467 rows and 25 columns. The model will train with the help of this dataset. It has 24 independent features and one dependent or target variable (Price_Rs.)

Later the dataset is divided into two parts, training and testing. After determine the proper model, the model is applied to predict the target variable for the test data.

```
print('No. of Rows :',data.shape[0])
print('No. of Columns :',data.shape[1])
pd.set_option('display.max_columns',None)
data.sample(n=6)
```

```
No. of Rows : 5467
No. of Columns : 25
```

1. To determine the data format, info() method is used. There are total 4 numarical columns among 25 columns.

2.3 Data Pre-processing Done:

2.3.1 Feature Engineering:

' --', 'null', 'NA', ' ' are present in the total dataset.

```
data.isin([' --','null','NA',' ']).sum().any()
```

```
True
```

```
data.replace(' --',np.nan, inplace = True)
data.replace('null',np.nan, inplace= True)
data.replace('NA',np.nan, inplace= True)
data.replace('  ', np.nan, inplace = True)
```

2.3.2 Change target variable (Price_Rs):

pdate is an object datatype. Need to convert in in date format.

```
data['Price_Rs'] = data['Price_Rs'].str.replace('Lakh','100000')
data['Price_Rs'] = data['Price_Rs'].str.replace('Crore','10000000')
data['Price_Rs']= data['Price_Rs'].str.replace(',','')
```

```
data[['a','b']] = data['Price_Rs'].str.split(expand=True)
```

```
data['a'] = data['a'].astype('float')
data['b'] = data['b'].astype('float')
data["Price_Rs."] = data['a'] * data['b']
```

2.3.3 Transferring Manufacturing_year into Age column:

Let's transfer the Manufacturing year column into corresponding age.

```
data['Car_Age'] = 2022 - data['Manufactring_year']
```

```
data.drop(columns=['Manufactring_year'], inplace = True)
```

2.3.4 Checking duplicate:

Let's check duplicate entry in the total dataset. There 1 row with completely duplicate values of all features.

```
data.duplicated().sum()
```

```
1
```

```
data.drop_duplicates(keep='last', inplace =True)
```

```
data.shape
```

```
(5466, 26)
```

Now save the dataset for future use.

```
data.to_excel('Final_Scraped_Details_Recheck.xlsx',index =False)
```

2.3.5 Checking Null:

```
Kilometers_driven        6
Fuel_Type                0
Ownership               14
Transmission             0
Insurance_Validity     502
Seats                    1
RTO                    499
Engine_displacement     17
Milage_kmpl            179
Torque_nm              191
Color                  633
Max_Torque             704
Engine_Type            752
No_of_cylinder         626
Turbo_charger         2821
Length_mm               87
Width_mm                88
Height_mm               89
Wheel_Base_mm          112
Front_Brake_Type       136
Steering_Type          702
Tyre_Type             2108
Price_Rs.                0
Brand_Name               0
Model_Name               0
Car_Age                  8
```

There are a huge set of null data in most of the features.

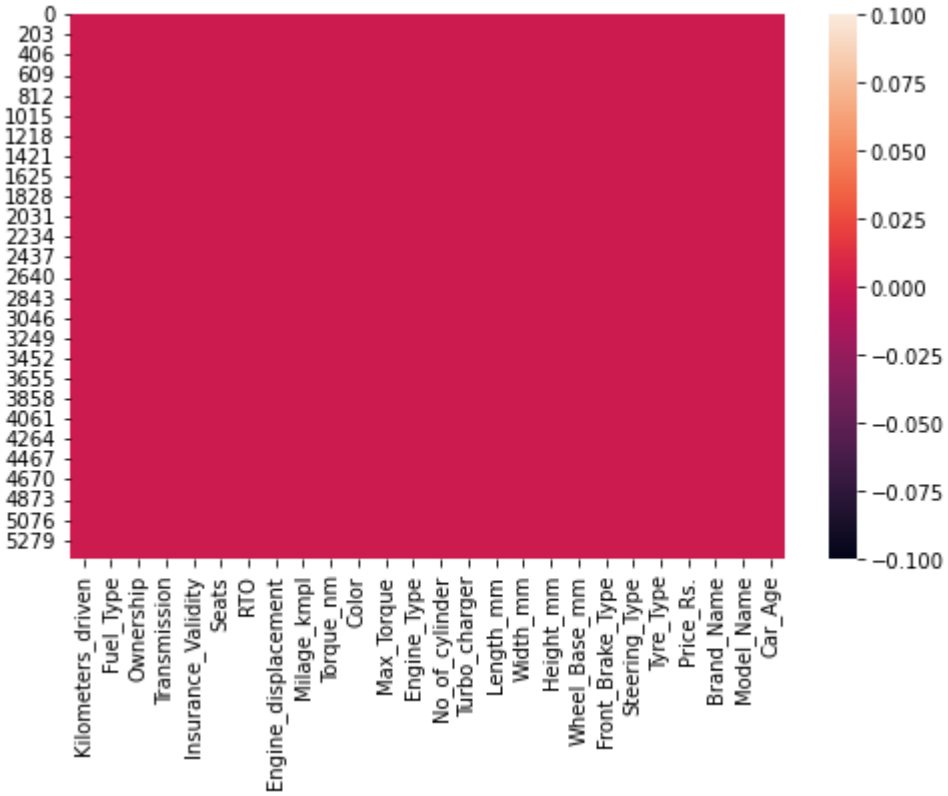
Now use describe() method to check the statistical features. Following observations are noted.

1. Null value is present.
2. Seems outliers are present as there is a difference between 75% and max of some features.
3. Minimum Engine_displacement is 0. Seems it is a error,
4. Minimum Length_mm is 4 where maximim 5453.
5. Minimum Car_age is 0 where maximum is 20 years.
6. Maximum Color of car is White.
7. In maximum cases Max_Torque of car is 200Nm@1750rpm.
8. In maximum cases Engine_Type is In-Line Engine

2.3.6 Null value imputation:

1. All of the Categorical variable can be imputate with mode.
2. All of the Numerical value can be imputate with Mean and Median. We can decide imputation method based on boxplot & Distplot.

After imputed all the null now let's move to the next steps.



2.3.7 Datatypes after converting into appropriate datatypes:

Skewness is present in all features including target. Let's remove the skewness except label as it is the target variable. Let's use Power Transformer to transform skewness in features.

```
data.columns.to_series().groupby(data.dtypes).groups
```

```
{float64: ['Kilometers_driven', 'Engine_displacement', 'Milage_kmpl', 'Torque_nm', 'Length_mm', 'Width_mm', 'Height_mm', 'Wheel
_Base_mm', 'Price_Rs.', 'Car_Age'], object: ['Fuel_Type', 'Ownership', 'Transmission', 'Insurance_Validity', 'Seats', 'RTO', 'C
olor', 'Max_Torque', 'Engine_Type', 'No_of_cylinder', 'Turbo_charger', 'Front_Brake_Type', 'Steering_Type', 'Tyre_Type', 'Brand
_Name', 'Model_Name']}
```

2.3.8 Encoding Categorical Data:

As are Categorical columns, let's encoded it.

```
# Using Label encoder for transforming Categorical data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Cat:
    data[i] = le.fit_transform(data[i])
data.head()
```

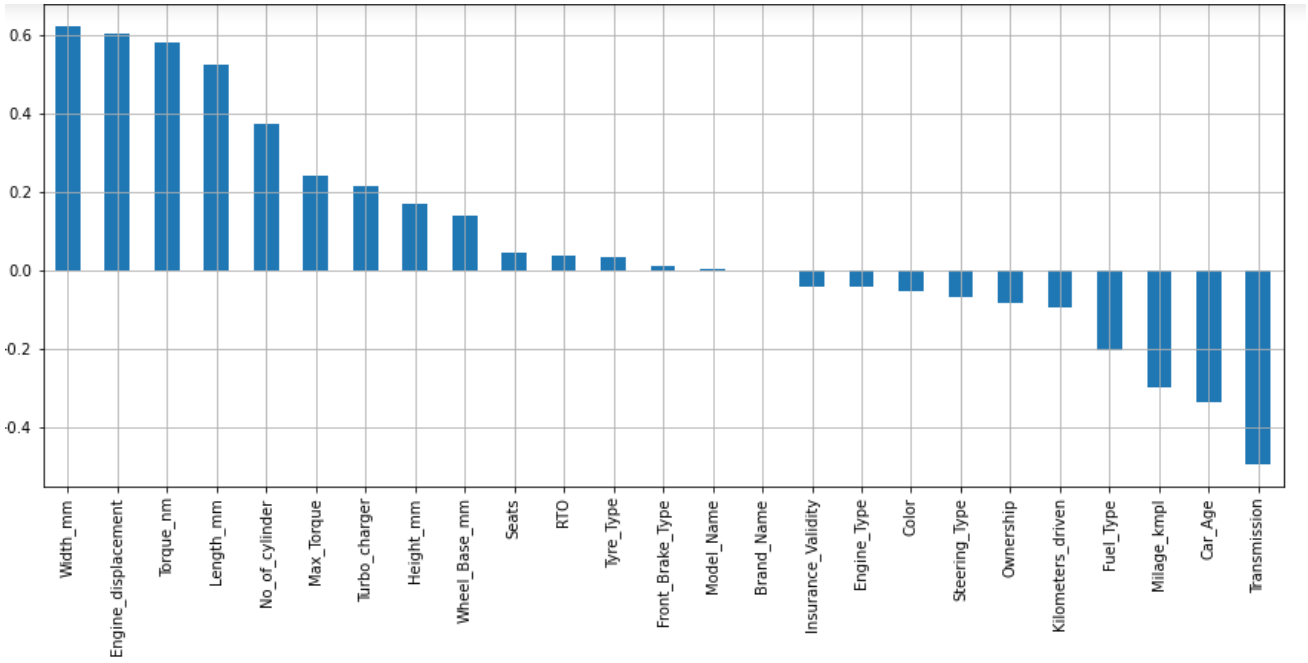
	Kilometers_driven	Fuel_Type	Ownership	Transmission	Insurance_Validity	Seats	RTO	Engine_displacement	Milage_kmpl
0	86226.0	1	1	1	2	3	57	1956.0	17.10
1	13248.0	4	1	0	2	3	57	1330.0	16.42
2	60343.0	4	1	0	2	3	57	2494.0	12.98
3	90281.0	1	1	1	2	5	57	2179.0	15.10
4	39821.0	4	1	0	2	3	57	1199.0	17.10

Now no categorical data is present.

2.3.9 Correlation:

Observations of the correlation:

1. Brand_Name and Model_Name is very less correlated with target variable.
2. Maximum correlation observe in Width and Engine_displacement followed by torque & length.
3. Most of features are moderately & poorly correlated with each other.



2.3.10 Outliers Detection and Removal:

Let’s check Outliers. From the previous Boxplot , it is seen that there are some outiers in numarical featues columns.
But it is a realistic dataset. So let's keep it.

2.3.11 Checking Skewness:

Let's say for this case, the skewness **range is -1 to +1.**

Any data have skewness above this level, are skewed. So, skewness is present in some features including target. Let's remove the skewness except Price as it is the target variable.

Let's use PowerTransformer to transform skewness in features.

```
skew_data = ['Kilometers_driven', 'Engine_displacement', 'Milage_kmpl', 'Torque_nm', 'Height_mm','Wheel_Base_mm']
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method = 'yeo-johnson')
```

```
data[skew_data] = scaler.fit_transform(data[skew_data].values)
data.head()
```

2.3.12 Checking Multicollinearity:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif= pd.DataFrame()
vif["VIF"]= [variance_inflation_factor(x_scale,i)for i in range(x.shape[1])]
vif["Features"] = x.columns
vif
```

For this data no Multicollinearity is present between different features.

2.4 Data Inputs- Logic- Output Relationships

We can see in the correlation that every features are correlated with each other and also they are highly correlated with target variable label.

2.5 State the set of assumptions (if any) related to the problem under consideration

No such assumptions are taken for this case.

2.6 Hardware and Software Requirements and Tools Used

Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz
RAM: 4.00 GB
System Type: 64-bit operating system, x64-based processor
Window: Windows 10 Pro
Anaconda – Jupyter Notebook
Libraries Used –

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

Except this, different libraries are used for machine learning model building from sklearn.

3. Model/s Development and Evaluation

3.1 Identification of possible problem-solving approaches (methods):

In this problem regression-based machine learning algorithm like linear regression can be used. For that first data encoding and data scaling using standard scalar is done. For building an appropriate ML model before implementing classification algorithms, data is split in training & test data using train_test_split. Then different statistical parameter like R2 score, MSE, RMSE etc. are determined for every algorithm. Hyper parameter tuning is performed to get the accuracy score much higher and accurate than earlier.

Then cross-validation is done to check best CV Score. Then the best curve is chosen from 7 different algorithm. Then final model is determined.

3.2 Testing of Identified Approaches (Algorithms)

Total 7 algorithms used for the training and testing are:

1. Linear Regression
2. DecisionTree Regressor
3. KNeighbors Regressor
4. GradientBoosting Regressor
5. RandomForest Regressor
6. Support Vector Regression
7. AdaBoost Regressor

3.3 Key Metrics for success in solving problem under consideration:

From metrics module of sklearn library import mean_absolute_error, mean_squared_error, r2_score. From model_selection also, we use cross_val_score. Those are the matrices use to validate the model's quality. Let's discuss every metrics shortly.

- R2 Score: R-squared (R2) is a statistical measure that shows how much of a dependent variable's variance is explained by one or more independent variables in a regression model.
- Mean Squared Error: You can determine how closely a regression line resembles a set of points using the mean squared error (MSE). This is accomplished by squaring the distances between the points and the regression line (also known as the "errors").
- Root Mean Squared Error: **Root Mean Square Error** is the measure of how well a regression line fits the data points. RMSE can also be construed as Standard Deviation in the residuals.
- Mean Absolute Error: The average discrepancy between the calculated and real values is calculated using mean absolute error. As it calculates inaccuracy in observations made on the same scale, it is also known as scale-dependent accuracy. It serves as a machine learning evaluation metric for regression models.

3.4 Run and Evaluate selected models

First find the best random state of train_test_split to get best accuracy. Here the random state is 445. Then after splitting the data into 4 different part and check the shape of the data.

```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

```
Training feature shape: (4099, 25)
Training target shape: (4099,)
Test feature shape: (1367, 25)
Test target shape: (1367,)
```

A Linear Regression:

```
x_train,x_test,y_train,y_test = train_test_split(x_scale,y,test_size = 0.25, random_state=445)

lin_reg= LinearRegression()

lin_reg.fit(x_train, y_train)

y_pred = lin_reg.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.6747691673480358
Mean absolute error: 478747.1763771032
Mean square error: 611403923181.5319
Root mean square error: 781923.2207714078
```

Then apply Hyper Parameter Tuning.

```
from sklearn.model_selection import GridSearchCV

grid = dict(fit_intercept=['True', 'False'], n_jobs=[1,-1])

grid_lin = GridSearchCV(estimator=lin_reg, param_grid= grid, cv=9)

grid_lin.fit(x_train, y_train)
grid_lin.best_params_
```

```
{'fit_intercept': 'True', 'n_jobs': 1}
```

```
grid_lin_best = LinearRegression(fit_intercept= 'True', n_jobs= 1)

grid_lin_best.fit(x_train, y_train)
y_pred = grid_lin_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

No such improvement seen after GridSearchCV.

B Decision Tree Regressor:

```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)

y_pred = dt.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.8595921262147387
Mean absolute error: 182715.4301389905
Mean square error: 350486438631.18286
Root mean square error: 592018.9512432714
```

Then apply Hyper Parameter Tuning.

```
param = {'criterion' : ["squared_error", "absolute_error"], 'min_samples_split' : range(1,5),
    'splitter' : ["best", "random"], 'max_features':["auto", "sqrt", "log2"],
    'min_samples_leaf' : range(1,5)}

grid_search = GridSearchCV(estimator = dt,cv=5,param_grid = param)
grid_search.fit(x_train, y_train)

print("Best Parameters:" , grid_search.best_params_)
```

```
Best Parameters: {'criterion': 'absolute_error', 'max_features': 'auto', 'min_samples_leaf': 3, 'min_samples_split': 3, 'splitt
er': 'random'}
```

```
grid_dt_best = grid_search.best_estimator_
grid_dt_best.fit(x_train, y_train)

y_pred = grid_dt_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

After using Gridseaech CV, R2 is not improved.

C KNeighbors Regressor:

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn import neighbors
rmse_val = []
for i in range(1,20):
    i = i+1
    knn = neighbors.KNeighborsRegressor(n_neighbors = i)

    knn.fit(x_train,y_train)
    y_pred=knn.predict(x_test)
    error =np. sqrt(mean_squared_error(y_test,y_pred))
    rmse_val.append(error)
    print('RMSE value for k= ' , i , 'is:', error)
```

For k=2 we get the best RMSE value for KNeighborsRegressor()

```
knn =KNeighborsRegressor(n_neighbors= 2)
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.7504507667323603
Mean absolute error: 254099.51389904902
Mean square error: 622925336544.0422
Root mean square error: 789256.1919579993
```

Then apply Hyper Parameter Tuning.

```
param = {'algorithm' : ['auto', 'ball_tree', 'kd_tree'],
             'leaf_size' : [30,40,25],
             'n_neighbors' : [2],'weights': ['uniform', 'distance'], 'p':[1,2,3]}

gridsearchknn = GridSearchCV(estimator = knn, param_grid=param, cv=5)

gridsearchknn.fit(x_train, y_train)

print("Best Parameters:" , gridsearchknn.best_params_)
```

```
Best Parameters: {'algorithm': 'auto', 'leaf_size': 30, 'n_neighbors': 2, 'p': 1, 'weights': 'distance'}
```

```
grid_knn_best = gridsearchknn.best_estimator_

grid_knn_best.fit(x_train, y_train)

y_pred = grid_knn_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.7997598643183639
Mean absolute error: 202716.2524865619
Mean square error: 499839860358.66095
Root mean square error: 706993.5362919953
```

Nothing is improved after GridSearchCV.

D Random Forest Regressor:

```
from sklearn.ensemble import RandomForestRegressor

rf= RandomForestRegressor()
rf.fit(x_train, y_train)

y_pred = rf.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.870623012523412
Mean absolute error: 149848.62629114848
Mean square error: 322951116337.32587
Root mean square error: 568287.881568247
```

Then apply Hyper Parameter Tuning.

```
params = {'n_estimators' : [100,110,80], 'criterion' : ["squared_error", "absolute_error"]}

rf_grd = GridSearchCV(rf, param_grid = params)
rf_grd.fit(x_train, y_train)
print('best params : ', rf_grd.best_params_)
```

```
best params :  {'criterion': 'absolute_error', 'n_estimators': 110}
```

```
grid_rf_best = rf_grd.best_estimator_

grid_rf_best.fit(x_train, y_train)

y_pred = grid_rf_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.8583488471373917
Mean absolute error: 157171.02845647404
Mean square error: 353589914556.6887
Root mean square error: 594634.2695781069
```

R2 score, RMSE is not improved after GridSearchCV.

E Support Vector Regression:

R2 score is very poor after using svm with three different kernel.

```
from sklearn.svm import SVR
svr_rbf = SVR(kernel='rbf')
svr_rbf.fit(x_train, y_train)

y_pred = svr_rbf.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
```

```
R2 Score: -0.0643397829885024
Mean absolute error: 647641.818277414
Mean square error: 2656807271390.2466
```

```
from sklearn.svm import SVR
svr_poly = SVR(kernel='poly')
svr_poly.fit(x_train, y_train)

y_pred = svr_poly.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
```

```
R2 Score: -0.0640842713728571
Mean absolute error: 647591.5552525278
Mean square error: 2656169462741.898
```

```
from sklearn.svm import SVR
svr_lin = SVR(kernel='linear')
svr_lin.fit(x_train, y_train)

y_pred = svr_lin.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
```

```
R2 Score: -0.053846181602811294
Mean absolute error: 639056.3895765635
Mean square error: 2630613120884.762
```

F Gradient Boosting Regressor :

```
from sklearn.ensemble import  GradientBoostingRegressor

gbdt= GradientBoostingRegressor()
gbdt.fit(x_train, y_train)

y_pred = gbdt.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9106339819187068
Mean absolute error: 168473.91282115385
Mean square error: 223075647879.0166
Root mean square error: 472308.8479787528
```

Then apply Hyper Parameter Tuning.

```
params = {'loss': ['squared_error', 'absolute_error'], 'n_estimators':[100,120,80],
          'criterion':['squared_error', 'mse'],'max_features': ['auto', 'sqrt']}

gbdt_grd = GridSearchCV(gbdt, param_grid = params, cv=5)
gbdt_grd.fit(x_train, y_train)
print('best params : ', gbdt_grd.best_params_)
```

```
best params :  {'criterion': 'squared_error', 'loss': 'squared_error', 'max_features': 'auto', 'n_estimators': 120}
```

```
grid_gbdt_best = gbdt_grd.best_estimator_

grid_gbdt_best.fit(x_train, y_train)

y_pred = grid_gbdt_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9189617287207444
Mean absolute error: 161670.50052482125
Mean square error: 202287908275.9488
Root mean square error: 449764.2807915595
```

R2 score, RMSE are slightly improved after GridSearchCV.

G Ada Boost Regressor:

```
from sklearn.ensemble import AdaBoostRegressor

ada= AdaBoostRegressor()
ada.fit(x_train, y_train)

y_pred = ada.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.6626150536634019
Mean absolute error: 678031.304334998
Mean square error: 842181033737.0099
Root mean square error: 917704.2190907754
```

Then apply Hyper Parameter Tuning.

```
params = { 'loss' : ['linear', 'square'], 'learning_rate': [0.1,0.001,1,0.01] ,
        'n_estimators':[50,60,40] }

ada_grd = GridSearchCV(ada, param_grid = params, cv=5)
ada_grd.fit(x_train, y_train)
print('best params : ', ada_grd.best_params_)
```

```
best params :  {'learning_rate': 0.1, 'loss': 'linear', 'n_estimators': 50}
```

```
grid_ada_best = ada_grd.best_estimator_

grid_ada_best.fit(x_train, y_train)

y_pred = grid_ada_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.7835441916593812
Mean absolute error: 344376.45776954875
Mean square error: 540317457569.1136
Root mean square error: 735062.8936146304
```

As per 7 different regression model we can see that the model with maximum R2 score and min RMSE value is GradientBoostingRegressor ().

3.5 Cross validation:

Let's check the cross validation score taking cross fold =5, before final prediction. Here also GradientBoostingRegressor() is the best model with max mean cv score and min standard deviation.

```
from sklearn.model_selection import cross_val_score

all_models = [lin_reg , dt , grid_knn_best , rf , grid_gbdt_best, grid_ada_best]

for i in all_models:
    cvscore = cross_val_score(i, x_scale,y, cv = 5)
    print('Cross Validation Score of :',i)
    print("\n Cross Validation Score : " ,cvscore)
    print("\nMean CV Score :",cvscore.mean())
    print("\nStd deviation :",cvscore.std())
    print("\n-----------")
    print("-----------")
```

3.6 Hyper Parameter Tuning:

We apply Hyper parameter tuning in every model. It can see that the best model after applying Hyper parameter is GradientBoostingRegressor.

Here R2 score is slightly improved after using hyper parameter tuning. First, R2 score was 0.9106339819187068, but after applying hyper parameter tuning it is 0.9189617287207444.

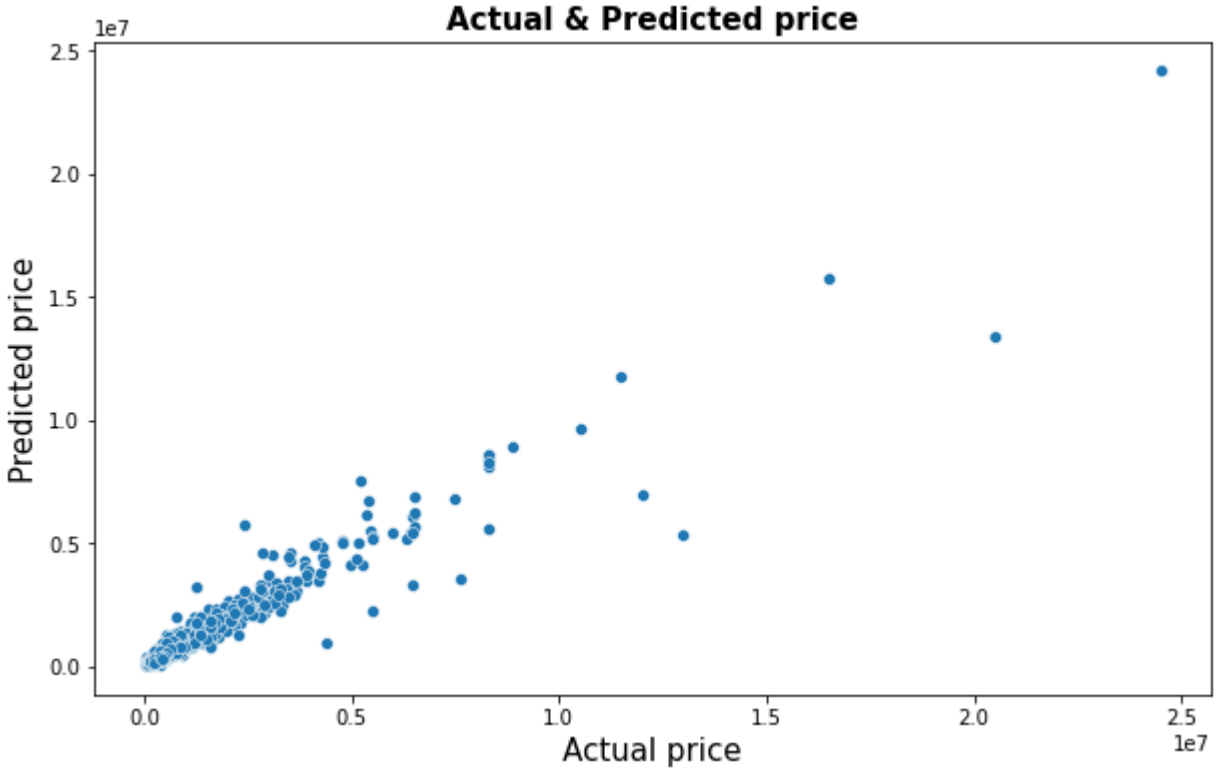
3.7 Final Model:

For final model the target variable as as follows after using GradientBoostingRegressor.

```
y_pred = grid_gbdt_best.predict(x_test)
y_pred
```

```
array([1040482.93852642, 1513892.06606559, 1718294.65923096, ...,
        377041.31282705, 4465253.92820718,  293447.76873334])
```

Let’s visualize the actual and predicted value of target data.

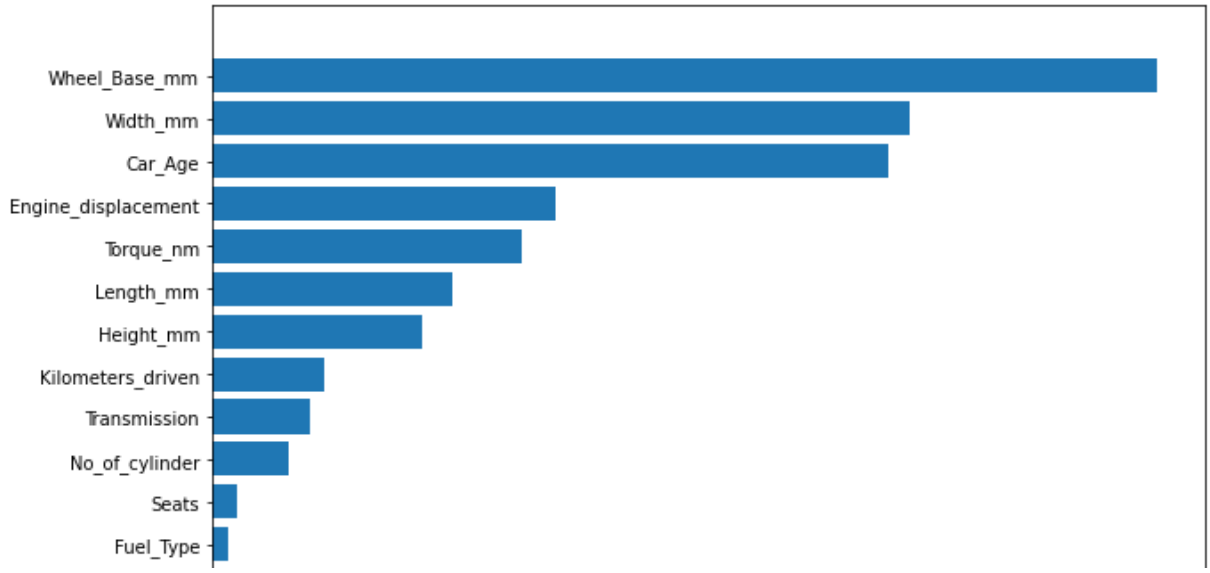



```
print('Final R2 Score:', r2_score(y_test, y_pred))
print('\nFinal Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('\nFinal Mean square error:', mean_squared_error(y_test, y_pred))
print('\nFinal Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
Final R2 Score: 0.9189617287207444

Final Mean absolute error: 161670.50052482125

Final Mean square error: 202287908275.9488

Final Root mean square error: 449764.2807915595
```

3.8 Check the important feature:

```
fimp = list(zip(feature,grid_gbdt_best.feature_importances_))
fimp.sort(key = lambda x : x[1])
plt.figure(figsize=(10,12))
plt.barh([x[0] for x in fimp],[x[1] for x in fimp])

plt.show()
```



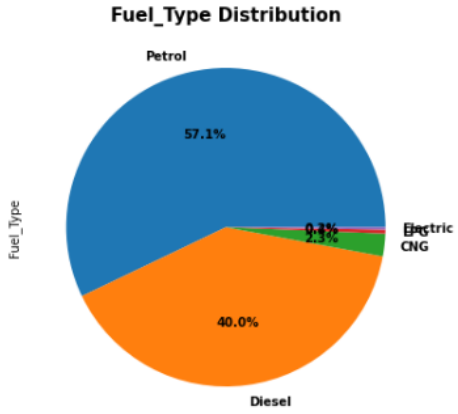
1. Wheel base, width are the most important feature for predicting price.
2. Interestingly Brand name, model name, steering type is the less importance feature.

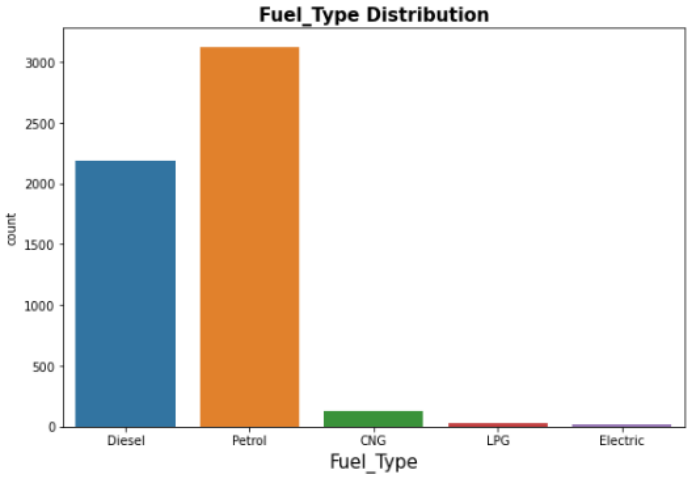
3.9 Load the model:

Let's save the model using pickle for future use. Then see the actual and predicted value of 6 random sample.

3.10 Visualizations:

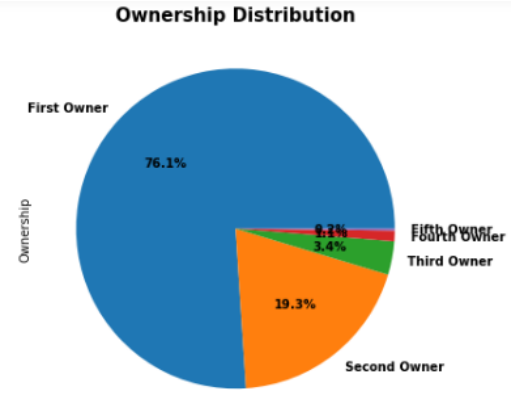
Let's start the observation exploration of feature analysis.

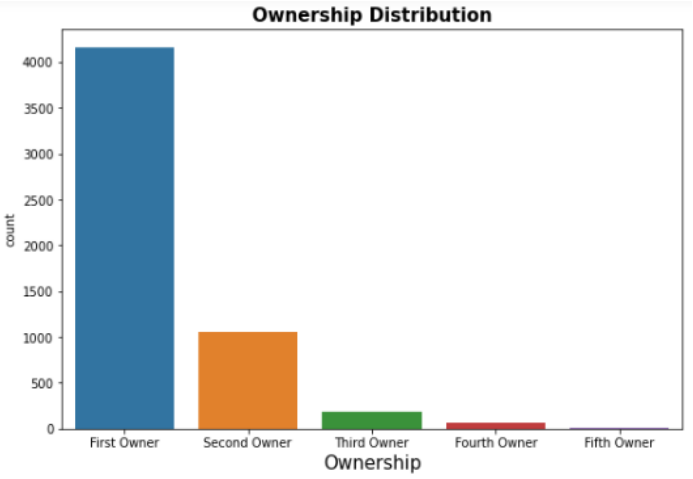


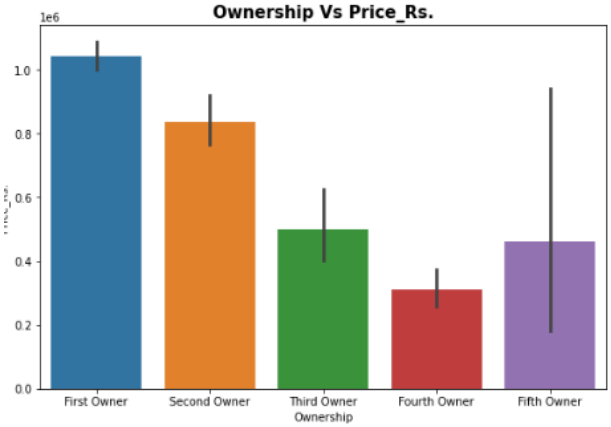


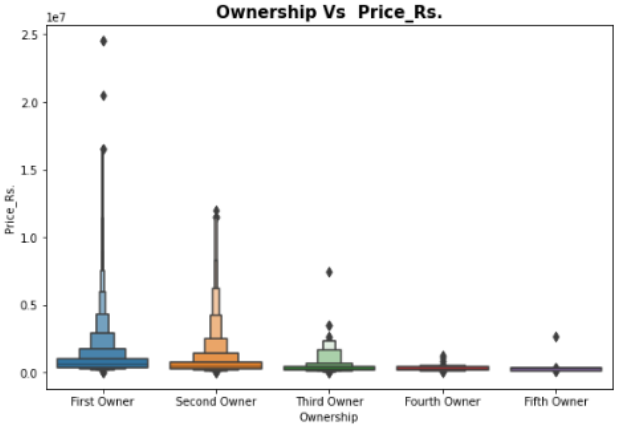
Observations:

1. Maximum Fuel type is Petrol which is 57 %.
2. Minimum Fuel type is Electric.
3. Maximum price is for Petrol and LPG car is the least price car.



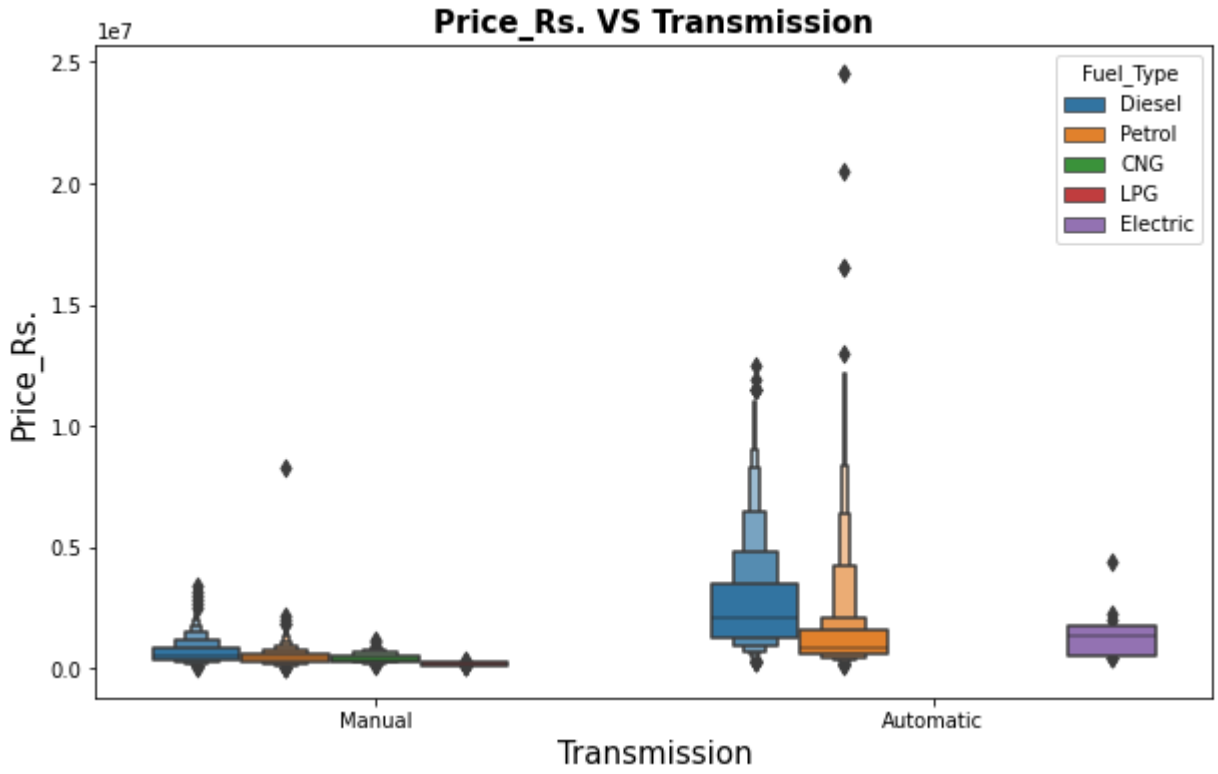






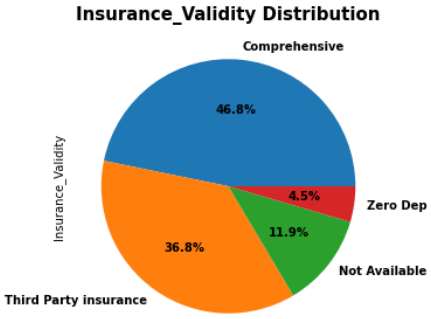
Observations:

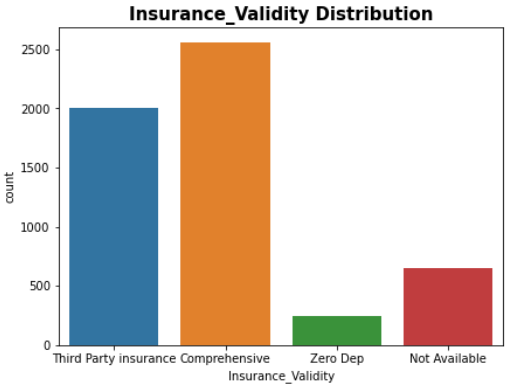
1. Maximum ownership is First owner around 76%.
2. Minimum is fifth owner.
3. Car from first owner is high price.



Observations:

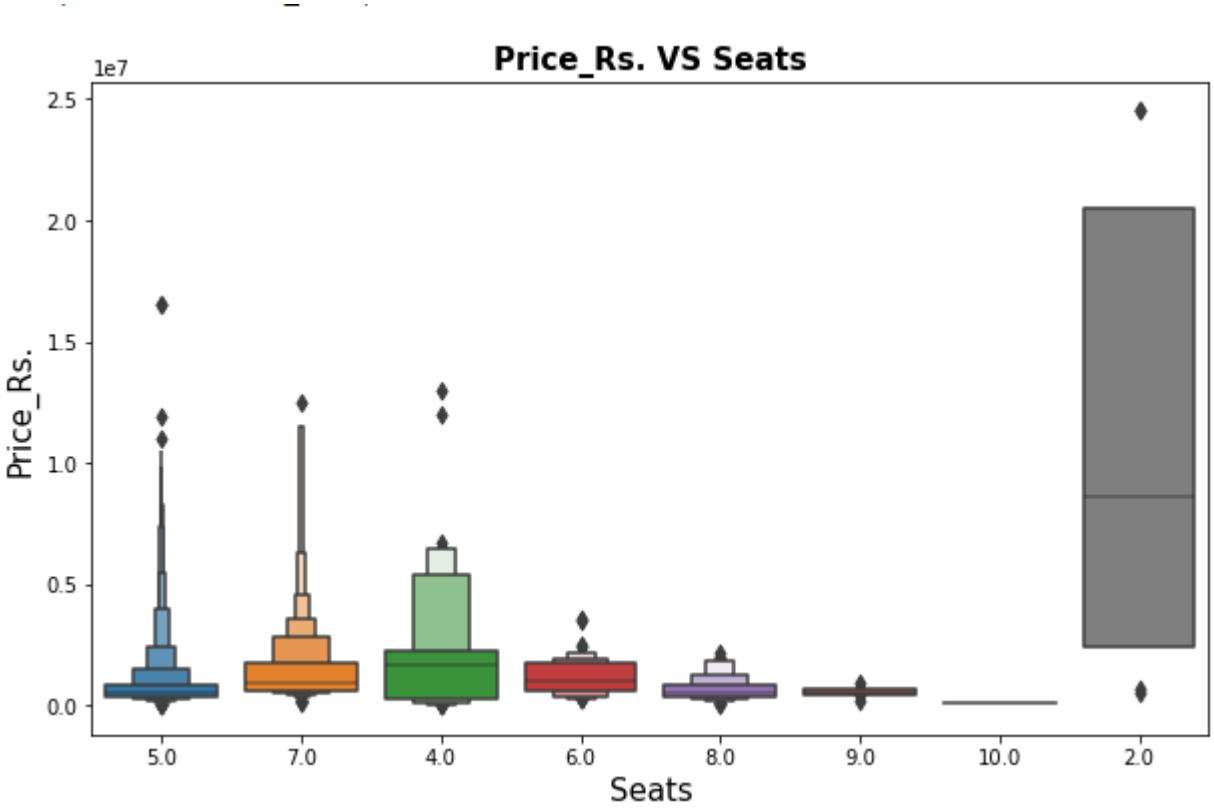
1. There are two Transmission type, Manual and Automatic.
2. Maximim transmission is Manual.
3. The pricing is high if the Transmission is Automatic.





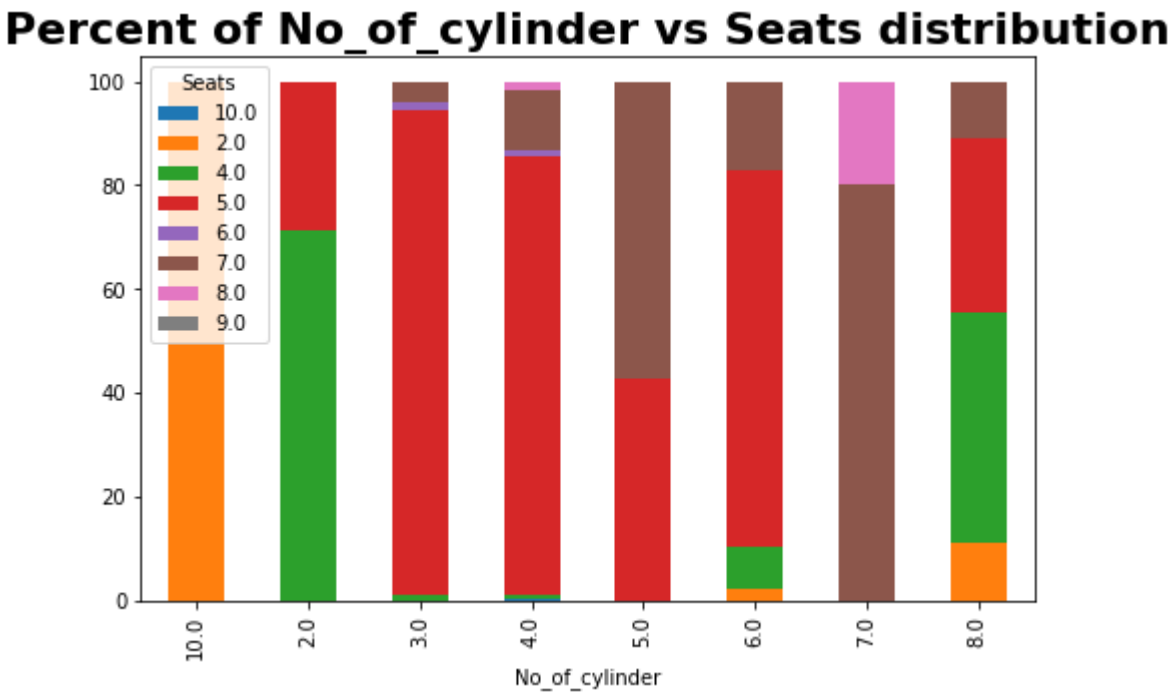
Observations:

1. Maximim Insurance_Validity is Comprehensive which is 2055.
2. No such relationship between Insurance_Validity and corrosponding price.



Observations:

1. Around 85% car is 5 seater.
2. Very few cars are 9 or 10 seater.
3. Among all 7 cars are 2 seater and they are very high in price.
4. Price is very less for 9 and 10 seated car



Observations:

1. No of cylinder of maximum car is 4 followed by 3,
2. Car price is high for 8 and 10 cylinder.

Now after some more observations the following observations are noted.

1. Maximim Turbo_charger type is No.
2. Only 2 are Twin type and their price is high.
3. Maximum Front_Brake_Type is Disc and Ventilated Disc.
4. Minimum type is Dual Circuit with ABS, ABS with BAS, Vacuum assisted hydraulic dual circuit w, Single Piston Sliding Caliper, Vented Disc.
5. The car price is high if Front_Brake_Type is Six piston claipers.
6. Maximum car are Power Steering_Type and it's price is also high compare to other steering types.
7. Hydraulic Steering_Type is most rare.
8. The car price is low if the Steering_Type is Hydraulic or Manual.
9. Maximum tyre is Tubeless_Radial and it's price is low.
10. Tubeless_ Runflat is the most rare type and it is the cosier tyre compare to other types.
11. For maximum case, car age is 5 followed by 4.
12. Car price is high for 1 and 2 year old car.
13. The car price is very low if the age of the car is above 12 years.
14. Maximum milage is around 20 (range is 10 to 30)
15. Most of the cases the price is high if the torque is high value.

3.11 Interpretation of the Results

After all the pre-processing steps, the ataset is ready to train machine learning models. All unnecessary features are deleted as they might give overfitting problem as well as it also could increase the time complexity. Now apply this dataset on different ML Regression Model (as discussed on part 3.4 - 'Run and Evaluate selected models') and check the best model for this particular dataset.

4. CONCLUSION

4.1 Key Findings and Conclusions of the Study

- As car model get old eventually its price reduces with time.
- In terms of Avg. Price as number of cylinders increases the average price increases.
- Electric cars are a very tiny market and also relatively expensive when compared to gasoline-powered vehicles.
- More than 50 % of car users prefer Power steering compares to others.
- Most cars with manual steering are at least ten years old.

4.2 Learning Outcomes of the Study in respect of Data Science

- Null removal is an important part of any problem.
- Scaling and standardization of data is mandatory.
- Null value imputation is an important steps.
- R2 score can be improved after applying hyper parameter tuning.
- Data needs to be much precise and detailed for much better score.

4.3 Limitations of this work and Scope for Future Work

- R2 score can increase with hyper parameter tuning with several different parameter. As it takes a lot of time, I am not able to use lot of parameters here for tuning.
- We can scrape more information from many internet marketplaces like olx and car24. Clearly, more information leads to more accurate forecasting.