Predicting Used Cars Price Using Machine Learning Model

Submitted by:

Aditya Paruleker

INTRODUCTION

- Business Problem Framing

  Since large number of used cars are being sold and purchased each year, determining whether the price listed on the website is genuine becomes an uphill task considering the amount of various factors that are involved in the market. Also in many developed countries, people often tend to purchase a used car as it is considered more affordable option. Data science can play a key role in this domain by not only helping the customers get the car of their choice based on their requirements, but also help the selling party find the best price for their used car. Companies involved in such businesses can also take advantage of this tool to adapt and grow in this market. The primary objective is to use the data science knowledge to build a model to predict the price of different types of used cars.

- Motivation for the Problem Undertaken

  Motivation behind this project is to help the people in the customers and people involved in the field of buying/selling the used cars to understand how exactly the prices of the cars vary with the variables. The information obtained from the model can help these customers get the right car as per their requirement, help seller find the best price and help companies involved make different strategies by concentrating on the business areas that will yield high returns. Further, the model will also be a good way for any parties involved understand the pricing dynamics of market.

Analytical Problem Framing

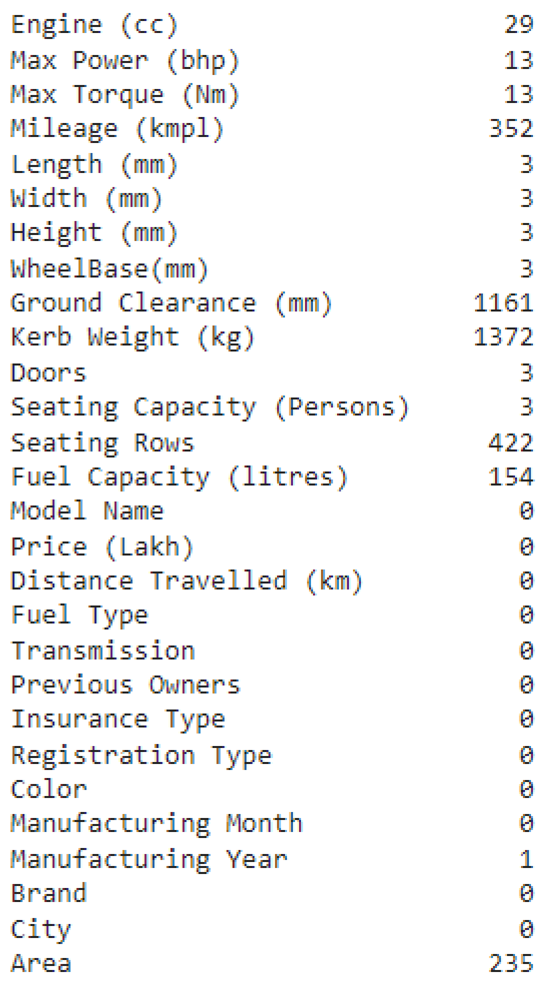
- Data Sources and their formats

  Details of more than 5000+ used cars from more than 10+ cities were collected from carswale.com website. The car type range from sedan to hatchback to SUVs.

- Preprocessing Done

  The imported dataset is analyzed to acquire basic information such as rows and column count, data type of each feature involved. Once basic insight is gained we proceed to the next phase,

  a) Null Value Analysis



```
Engine (cc)                    29
Max Power (bhp)                13
Max Torque (Nm)                13
Mileage (kmpl)                352
Length (mm)                     3
Width (mm)                      3
Height (mm)                     3
WheelBase(mm)                   3
Ground Clearance (mm)        1161
Kerb Weight (kg)             1372
Doors                           3
Seating Capacity (Persons)      3
Seating Rows                  422
Fuel Capacity (litres)        154
Model Name                      0
Price (Lakh)                    0
Distance Travelled (km)         0
Fuel Type                       0
Transmission                    0
Previous Owners                 0
Insurance Type                  0
Registration Type               0
Color                           0
Manufacturing Month             0
Manufacturing Year              1
Brand                           0
City                            0
Area                          235
```

During analysis it is observed that some of the features have empty string values along with mislabelled datatype. Once the features with empty string values are replaced with null values, the mislabelled datatype are relabelled again and null value analysis is performed to check the count of the missing values. Furthermore, the missing value features are tested for data loss and if the data loss is found to be more than 30% then the entire column is dropped, if not, the null values are replaced using mean for numeric dataype and mode for object datatype.

b) Outlier Analysis

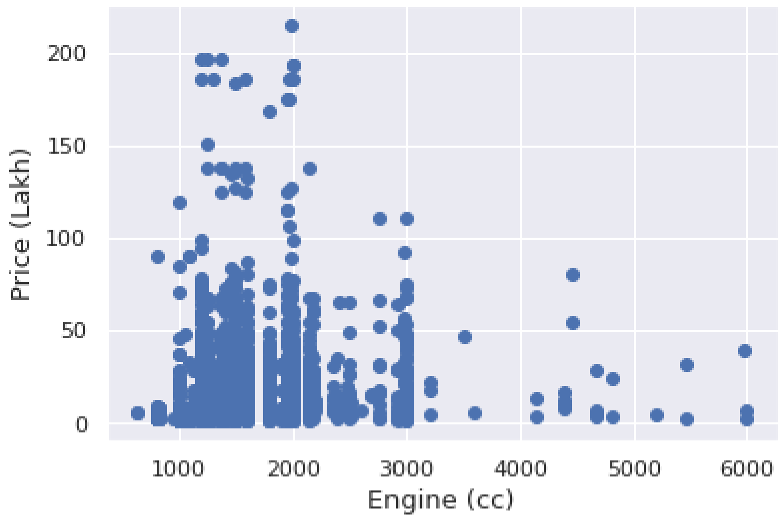


figure (a)

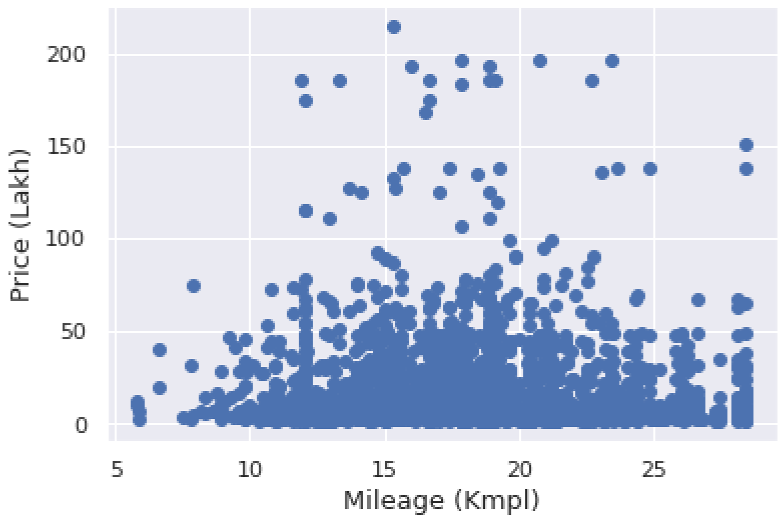


figure (b)

Extreme case of outlier was detected in some of the features. The outliers in the above figure (b) indicating that there are cars with good mileage (indicates non luxury cars) available at extremely high rate similarly in figure(a) there are used cars available with high engine cc (indicates luxury cars) with extremely cheap rate which can technically be not possible .

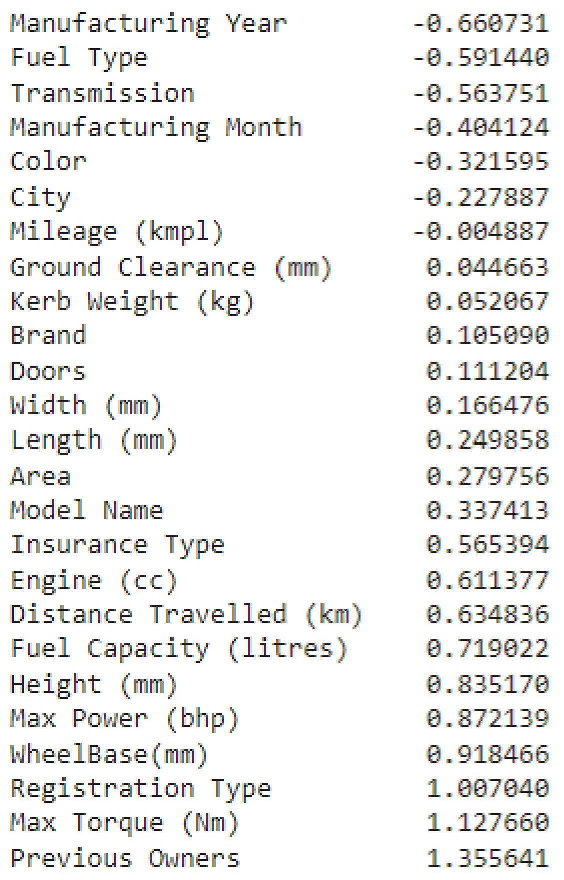
Usually removing the outliers from dataset is not always considered to be safe and simply removing all them might affect our models incase there were also outliers in the test dataset. But in this case there are some extreme outliers that are needed to be handled for better working of model.

c) **Separating Target and Input features**
Before proceeding with further preprocessing, the input features labelled x are separated from target features labelled y.

d) **Encoding**
Based on dataset analysis, all the categorical features with hierarchy were encoded using ordinal encoder while remaining categorical features were encoded using label encoder.

In this project all the categorical features were without any hierarchy and hence label encoder was used.

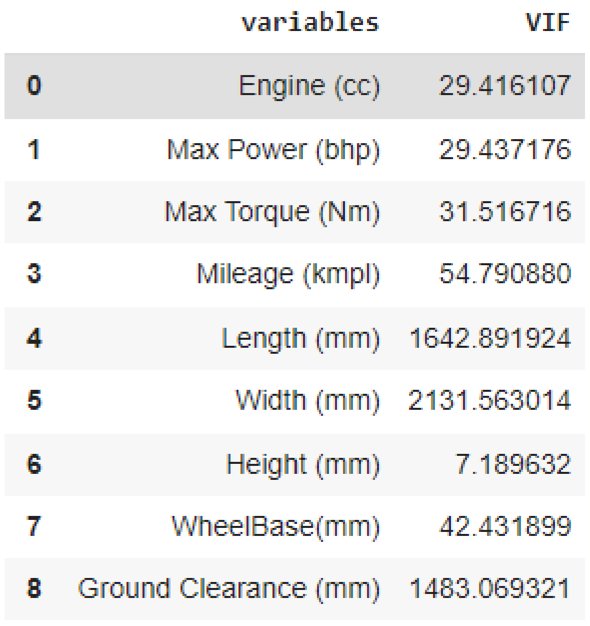
e) **Skew Analysis**
There were total of 7 numeric features that contained skewness. Quantile transformation was applied on these features to remove the skewness and hence all the skewness were reduced within range of +/-0.5.



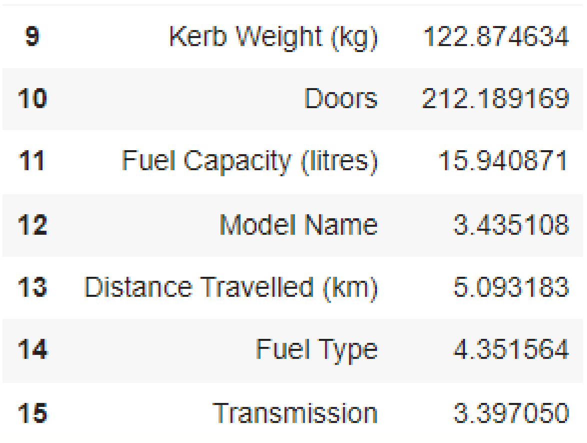
```
Manufacturing Year          -0.660731
Fuel Type                   -0.591440
Transmission                -0.563751
Manufacturing Month         -0.404124
Color                       -0.321595
City                        -0.227887
Mileage (kmpl)              -0.004887
Ground Clearance (mm)        0.044663
Kerb Weight (kg)             0.052067
Brand                        0.105090
Doors                        0.111204
Width (mm)                   0.166476
Length (mm)                  0.249858
Area                         0.279756
Model Name                   0.337413
Insurance Type               0.565394
Engine (cc)                  0.611377
Distance Travelled (km)      0.634836
Fuel Capacity (litres)       0.719022
Height (mm)                  0.835170
Max Power (bhp)              0.872139
WheelBase(mm)                0.918466
Registration Type            1.007040
Max Torque (Nm)              1.127660
Previous Owners              1.355641
```

f) Collinearity Analysis

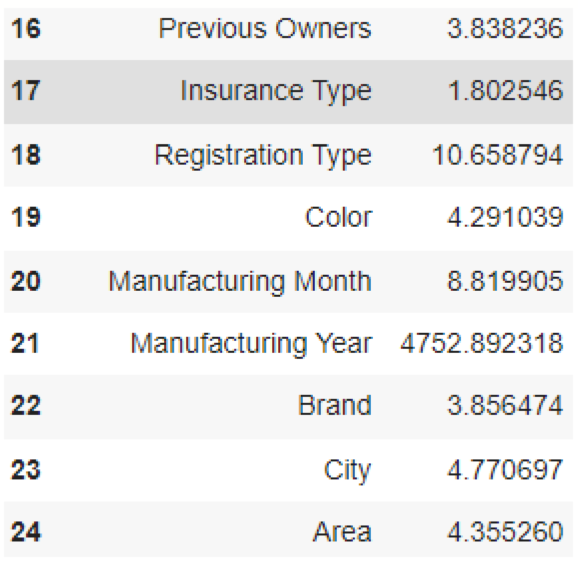
In order to avoid two or more input features from feeding same information into the model, it is necessary to detect correlation between independent variables using Variance Inflation factor or VIF method. Below is the screenshot of dataset and its VIF value.



	variables	VIF
0	Engine (cc)	29.416107
1	Max Power (bhp)	29.437176
2	Max Torque (Nm)	31.516716
3	Mileage (kmpl)	54.790880
4	Length (mm)	1642.891924
5	Width (mm)	2131.563014
6	Height (mm)	7.189632
7	WheelBase(mm)	42.431899
8	Ground Clearance (mm)	1483.069321



9	Kerb Weight (kg)	122.874634
10	Doors	212.189169
11	Fuel Capacity (litres)	15.940871
12	Model Name	3.435108
13	Distance Travelled (km)	5.093183
14	Fuel Type	4.351564
15	Transmission	3.397050



16	Previous Owners	3.838236
17	Insurance Type	1.802546
18	Registration Type	10.658794
19	Color	4.291039
20	Manufacturing Month	8.819905
21	Manufacturing Year	4752.892318
22	Brand	3.856474
23	City	4.770697
24	Area	4.355260

In above case, we decided to drop only those features that are found to have a very high VIF value and also contribute less towards target variable.

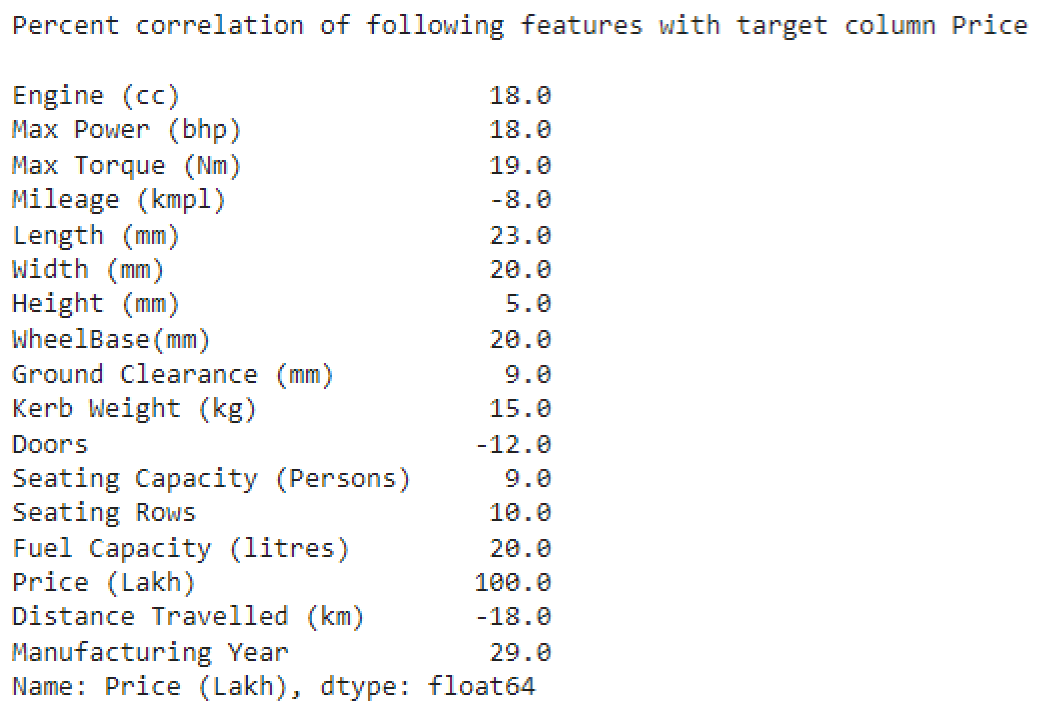
g) **Feature Scaling**

Feature scaling is a technique frequent used in machine learning to generalize the data points so that the distance between them is smaller.

In this project, since the data doesn't have Gaussian distribution, we use normalization technique to rescale the data.
The **Min-Max Normalization** uses distribution value between 0 and 1 to re-scales all the feature values.

- Data Inputs- Logic- Output Relationships

  Correlation is a measure of how strongly or weakly connected two features are in a dataset. Below table show the percentage of correlation each input feature has with the output column. Higher the percentage stronger is the bond, and vice versa. Positive value indicates that the relationship between the two variables move in the same direction whereas negative value indicates that the relationship between two variables move in opposite direction.



```
Percent correlation of following features with target column Price

Engine (cc)                     18.0
Max Power (bhp)                 18.0
Max Torque (Nm)                 19.0
Mileage (kmpl)                  -8.0
Length (mm)                     23.0
Width (mm)                      20.0
Height (mm)                      5.0
WheelBase(mm)                   20.0
Ground Clearance (mm)            9.0
Kerb Weight (kg)                15.0
Doors                          -12.0
Seating Capacity (Persons)       9.0
Seating Rows                    10.0
Fuel Capacity (litres)          20.0
Price (Lakh)                   100.0
Distance Travelled (km)        -18.0
Manufacturing Year              29.0
Name: Price (Lakh), dtype: float64
```

- Hardware and Software Requirements and Tools Used

  Following libraries were used in this project,

  Pandas
  Created by Wes McKinney in 2008, this python library is widely used for data manipulation, analyzing and cleaning of data. Apart from this, it also helps in finding correlation between columns and in deleting rows.

Numpy
Created by Travis Oliphant in 2005, this python library provides an array object called ndarray i.e. up to 50x faster than traditional python lists. It was has functions can be used in linear algebra, matrices etc

Seaborn
Seaborn is a high level interface based data visualization library that uses matplotlib library underneath the working.

Matplotlib
Unlike saeborn, matplotlib is a low level data visualization python library. Majority of its function lies in the submodule named pyplot

Scikit-Learn
It provides tools for classification, regression, clustering and dimensionality reduction through its interface in python.

Pickle
Used for serializing (pickling) and de-serializing (unpickling) python object structure so that the data can be easily transferred from system to system and then stored in a file.

Selenium
It is widely used open source tool that not only help in automating the web browser but also helps in web scrapping the data from the various sources.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

In this study several machine learning models were tested and their results were compared before selecting the model that gave best result among all.

Linear Regression
Using linear approach for modeling, this supervised machine learning model constructs relationship between scalar response and explanatory variables. There are many different types of regression model based on type of relationship between independent and dependent variables.

XGB Boost
This Extreme Gradient Boosting supervised machine learning model uses parallel tree boosting method for both regression and classification. It is widely preferred to be used when the size of the dataset is large enough.

Ensemble Methods
Ensemble technique uses several base estimators of machine learning model to predict the output. This results are then combined to improve the robustness of model over single estimator.

Randomforest Regressor
Using multiple decision trees as base, the model randomly performs the dataset sampling over the rows and features such that it form sample datasets for every model.

AdaBoost Regressor
The model first fits on the original dataset and then this Adative Boosting regressor further fits on extra copies of same dataset only with adjusted instances of weights based on the current prediction error.

GradientBoost Regressor
The model after calculating the residual i.e. the difference between actual value and predicted target value; trains the weak model for mapping the features to that residual.

Bagging Regressor
This ensemble method uses random subsets of the original dataset on which each base regressors are fitted; then final prediction is formed by either by voting or by averaging the aggregate of individual predictions.

Voting Regressor
Unlike bagging regressor, voting regressor uses whole dataset on which each base regressors are fitted and then final prediction is formed by averaging the individual predictions.

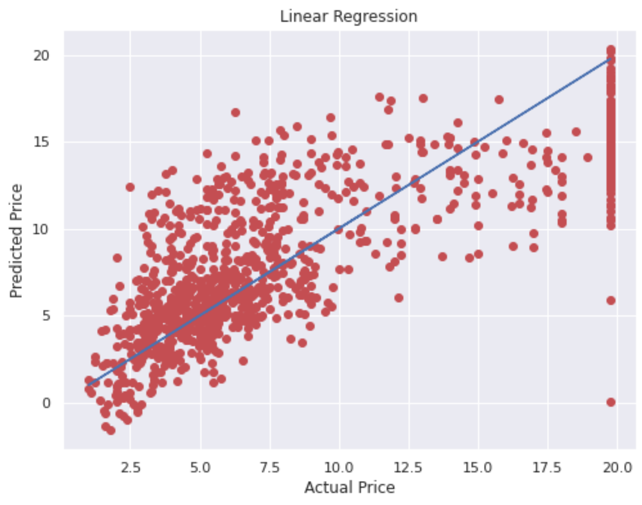
- Run and Evaluate models

Linear Regression

```
for i in range(100):

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,
random_state=i)
lr.fit(x,y)
pred_train=lr.predict(x_train)
pred_test=lr.predict(x_test)
print("At Random State : ",i)
print("Training r2: ",round(r2_score(y_train,pred_train)*100,2))
print("Testing r2: ",round(r2_score(y_test,pred_test)*100,2))
print("\n")

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,
random_state=94)
lr.fit(x,y)
pred_train=lr.predict(x_train)
pred_test=lr.predict(x_test)
print("Training r2: ",round(r2_score(y_train,pred_train)*100,2))
print("Testing r2: ",round(r2_score(y_test,pred_test)*100,2))
print("\n")
print("mean_absolute_error: ",mean_absolute_error(y_test,pred_tes
t))
print("mean_squared_error: ",mean_squared_error(y_test,pred_test)
)
print("Root mean_square: ",np.sqrt(mean_squared_error(y_test,pred
_test)))


ls=r2_score(y_test,pred_test)
for j in range(2,20):
cv_score=cross_val_score(lr,x,y,cv=j)
print(cv_score)
print("At cv: ",j)
print("Cross Validation Score: ",round(cv_score.mean()*100,2))
print("r2 Score: ",round(ls*100,2))
print("\n")

plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_test,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Linear Regression")
plt.show()
```

Output

```
Training r2:  65.35
Testing r2:  63.06


mean_absolute_error:  2.623941694261641
mean_squared_error:  12.053367355519674
Root mean_square:  3.471795984144183
```



Lasso Regression

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Lasso

parameters={'alpha':[.00001,.0001,.001,.01,.1,1,10],
       'random_state': list(range(0,10)),
       'selection':['cyclic', 'random'],
       'fit_intercept': ['True','False']}

ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)

from sklearn.linear_model import Lasso
ls=Lasso(alpha=10,random_state=6,fit_intercept=True,selection='ra
ndom')
ls.fit(x_train,y_train)
ls_score_train=ls.score(x_train,y_train)
pred_ls=ls.predict(x_test)
print("Training Accuracy: ",round(ls_score_train*100,2))
lss=r2_score(y_test,pred_ls)
print("Testing Accuracy: ",round(lss*100,2))
cv_score=cross_val_score(ls,x,y,cv=8)
cv_mean=cv_score.mean()
print("Cross Validation Score: ",round(cv_mean*100,2))
```

Output

```
Training Accuracy:  65.41
Testing Accuracy:  62.54
Cross Validation Score:  33.31
```

Model Selection User Defined Function

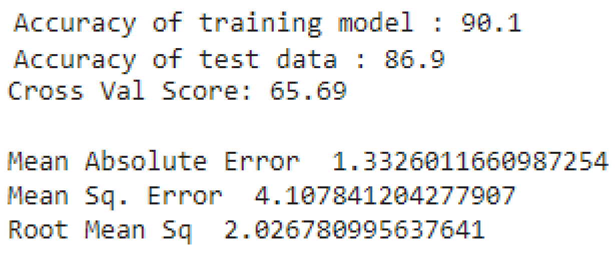
```
from sklearn.metrics import mean_absolute_error,mean_squared_erro
r

def model_selection(algorithm_instance,x_train,y_train,x_test,y_t
est):
algorithm_instance.fit(x_train,y_train)
model_pred_train=algorithm_instance.predict(x_train)
model_pred_test=algorithm_instance.predict(x_test)
print("Accuracy of training model :",round(r2_score(y_train,model
_pred_train)*100,2))
print("Accuracy of test data :",round(r2_score(y_test,model_pred_
test)*100,2))
```

```
 rfscore=cross_val_score(algorithm_instance,x,y,cv=8)
 rfc=rfscore.mean()
 print('Cross Val Score:',round(rfc*100,2))
 print("\nMean Absolute Error ",mean_absolute_error(y_test,model_
 pred_test))
 print("Mean Sq. Error ",mean_squared_error(y_test,model_pred_tes
 t))print("Root Mean Sq ",np.sqrt(mean_squared_error(y_test,model
 _pred_test)))
 print("\n")
```

XGB Boost

```
import xgboost as xg
xgb = xg.XGBRegressor()
model_selection(xgb,x_train,y_train,x_test,y_test)
```

Output



```
 Accuracy of training model : 90.1
 Accuracy of test data : 86.9
Cross Val Score: 65.69

Mean Absolute Error  1.3326011660987254
Mean Sq. Error  4.107841204277907
Root Mean Sq  2.026780995637641
```

Randomforest Regressor

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

parameter={'criterion':['mse','mae'],
     'max_features' : ["auto","sqrt","log2"],
     'n_estimators':range(0,100,25)}

   rf=RandomForestRegressor()
   clf=GridSearchCV(rf,parameter)
   clf.fit(x_train,y_train)
   print(clf.best_params_)

   rf=RandomForestRegressor(criterion='mse',max_features="auto", n
   _estimators=75)
   model_selection(rf,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 98.49
Accuracy of test data : 88.59
Cross Val Score: 67.75

Mean Absolute Error  1.0799747712418302
Mean Sq. Error  3.5775527142657957
Root Mean Sq  1.8914419669304674
```

AdaBoost Regressor

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor

parameter={'loss':['linear', 'square', 'exponential'],
     'random_state' : range(0,100,25),
      'learning_rate':[0,1.0],
      'n_estimators':range(0,100,25)}

   rf2=AdaBoostRegressor()
   clf=GridSearchCV(rf2,parameter)
   clf.fit(x_train,y_train)
   print(clf.best_params_)

   rf2=AdaBoostRegressor(learning_rate= 1.0, loss= 'exponential',
   n_estimators= 50, random_state= 50)
   model_selection(rf2,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 67.81
Accuracy of test data : 63.32
Cross Val Score: 17.01

Mean Absolute Error  2.749934536940478
Mean Sq. Error  11.499459354235276
Root Mean Sq  3.3910852767565838
```

GradientBoost Regressor

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor

parameter={'loss':['squared_error', 'absolute_error', 'huber','qu
antile'],
     'criterion':['friedman_mse', 'squared_error', 'mse'],
      'learning_rate':[0,1.0],
      'n_estimators':range(0,100,25)}

   rf3=GradientBoostingRegressor()
   clf=GridSearchCV(rf3,parameter)
   clf.fit(x_train,y_train)
   print(clf.best_params_)

   rf3=GradientBoostingRegressor(learning_rate= 1.0, loss= 'absolu
   te_error', n_estimators= 25, criterion= 'squared_error')
   model_selection(rf3,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 93.75
Accuracy of test data : 84.24
Cross Val Score: 64.88

Mean Absolute Error  1.3058981262709677
Mean Sq. Error  4.941989513250324
Root Mean Sq  2.2230585942008645
```

Bagging Regressor

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import BaggingRegressor

parameter={'oob_score':['True','False'],
     'n_jobs':range(0,10,2),
      'random_state':range(0,100,25),
      'n_estimators':range(0,100,25)}

   rf4=BaggingRegressor()
   clf=GridSearchCV(rf4,parameter)
   clf.fit(x_train,y_train)
   print(clf.best_params_)



rf4=BaggingRegressor(oob_score= True, n_jobs= 2, n_estimators= 75
, random_state= 75)
model_selection(rf4,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 98.48
Accuracy of test data : 88.93
Cross Val Score: 67.03

Mean Absolute Error  1.054331633986928
Mean Sq. Error  3.4717681165490193
Root Mean Sq  1.863268127927116
```

Voting Regressor

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import VotingRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor

estimators = [ ('knc', KNeighborsRegressor()), ('svr',SVR()) ]
parameter={
    'n_jobs':range(0,10,2),
     'verbose':[True,False]
    }
   rf5=VotingRegressor(estimators)
   clf=GridSearchCV(rf5,parameter)
   clf.fit(x_train,y_train)
   print(clf.best_params_)

   rf5=VotingRegressor(estimators, n_jobs= 2, verbose= True)
   model_selection(rf5,x_train,y_train,x_test,y_test)
```

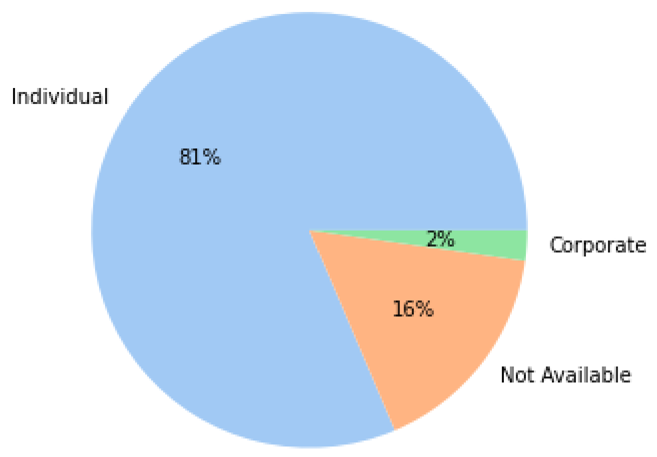
Output

```
Accuracy of training model : 77.38
Accuracy of test data : 70.12
Cross Val Score: 44.89

Mean Absolute Error  2.0811139743255507
Mean Sq. Error  9.750236300030377
Root Mean Sq  3.122536837257549
```

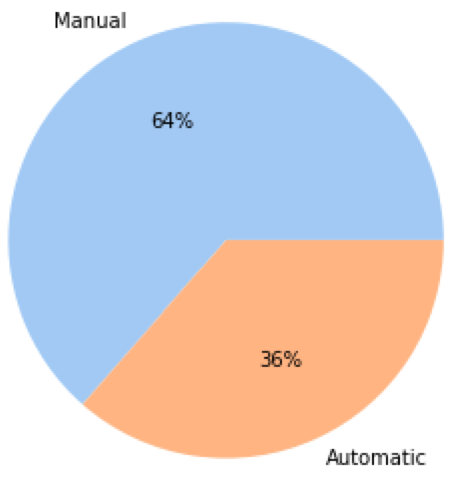
- Visualizations

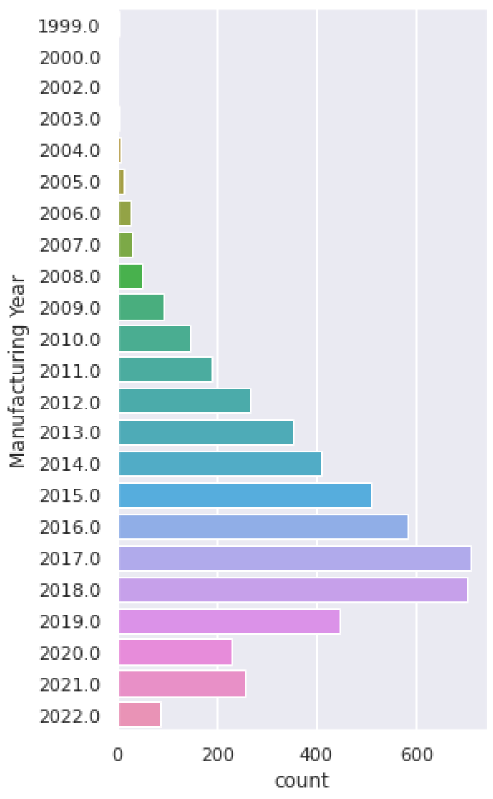
a) Univariate Analysis

```
Registration Type
Individual       4151
Not Available     834
Corporate         114
Name: Registration Type, dtype: int64
```

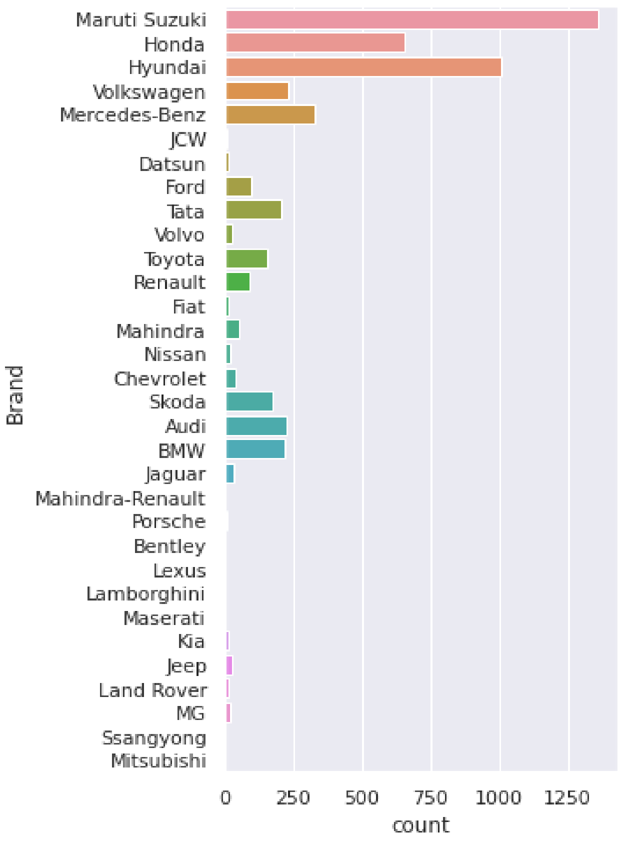


```
Transmission Type
Manual       3241
Automatic    1858
Name: Transmission, dtype: int64
```

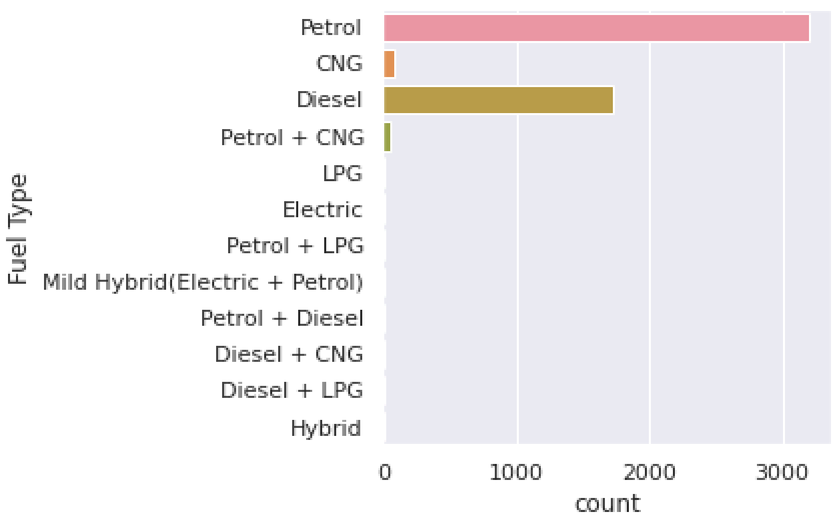


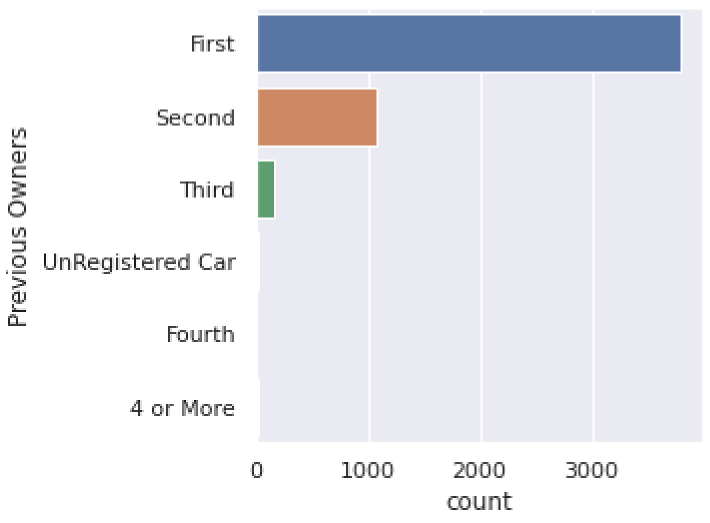


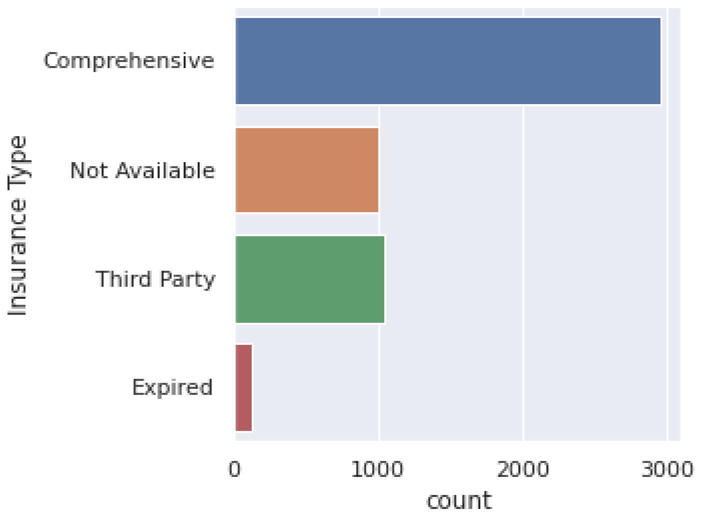
Manufacturing Year
1999.0
2000.0
2002.0
2003.0
2004.0
2005.0
2006.0
2007.0
2008.0
2009.0
2010.0
2011.0
2012.0
2013.0
2014.0
2015.0
2016.0
2017.0
2018.0
2019.0
2020.0
2021.0
2022.0
0
200
400
600
count

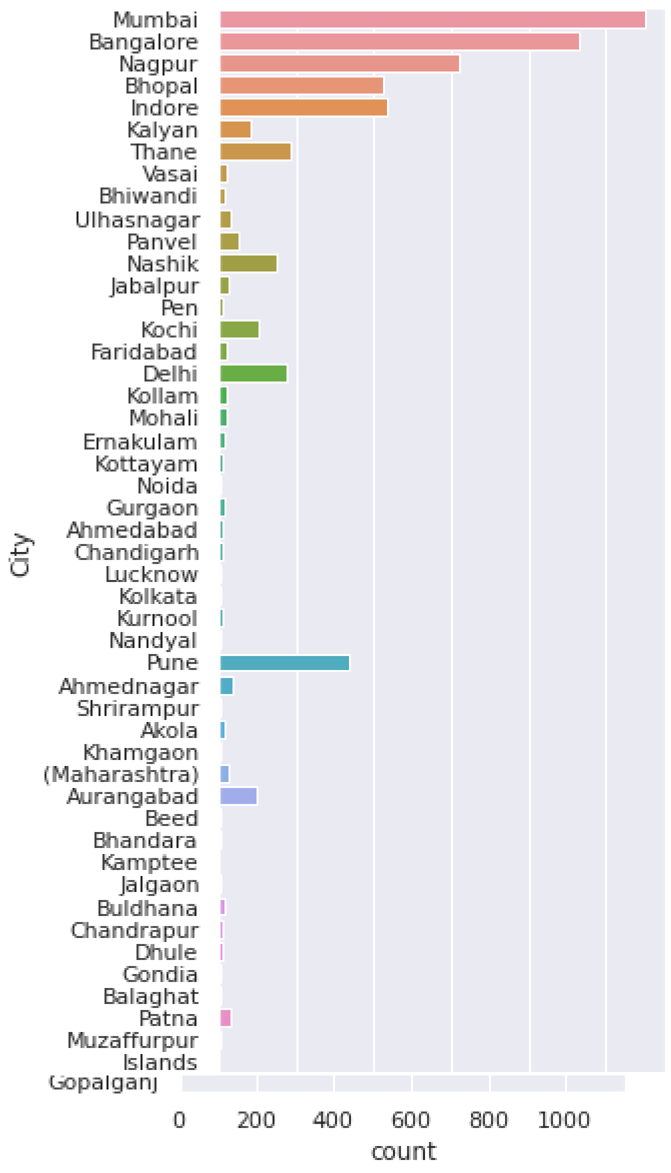


Brand
Maruti Suzuki
Honda
Hyundai
Volkswagen
Mercedes-Benz
JCW
Datsun
Ford
Tata
Volvo
Toyota
Renault
Fiat
Mahindra
Nissan
Chevrolet
Skoda
Audi
BMW
Jaguar
Mahindra-Renault
Porsche
Bentley
Lexus
Lamborghini
Maserati
Kia
Jeep
Land Rover
MG
Ssangyong
Mitsubishi
0
250
500
750
1000
1250
count


Fuel Type
Petrol
CNG
Diesel
Petrol + CNG
LPG
Electric
Petrol + LPG
Mild Hybrid(Electric + Petrol)
Petrol + Diesel
Diesel + CNG
Diesel + LPG
Hybrid
0
1000
2000
3000
count

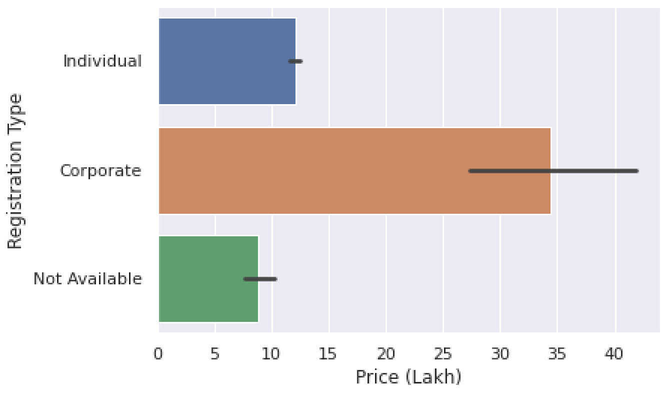

Previous Owners
First
Second
Third
UnRegistered Car
Fourth
4 or More
0
1000
2000
3000
count

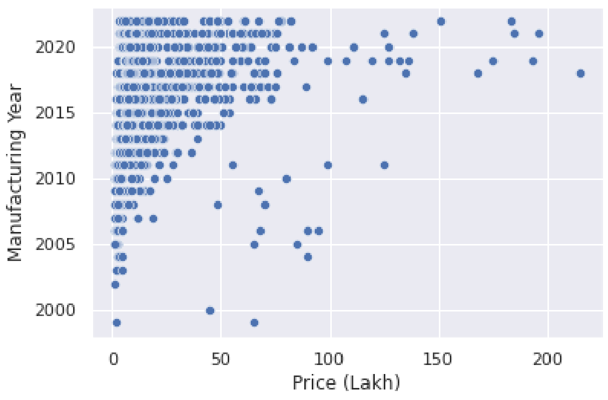

Insurance Type
Comprehensive
Not Available
Third Party
Expired
0
1000
2000
3000
count

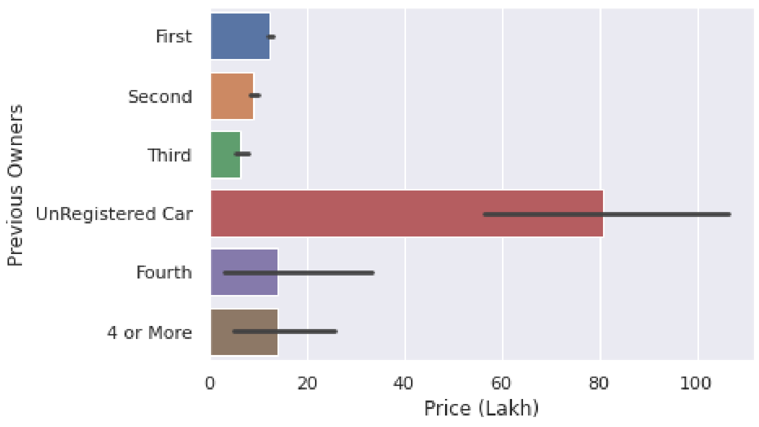


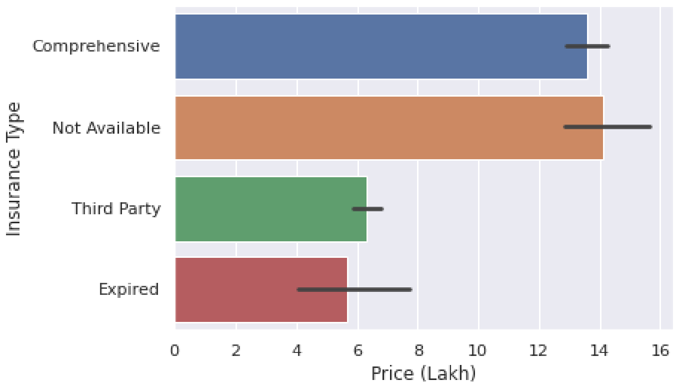
- The manufacturing year for car ranges from 1999 to 2022.
- The car with highest price in the dataset is of worth 215 lakh (2.5crore) whereas the car with lowest price is of worth 1 lakh.
- There are cars with engine cc range between 624 cc to max 5998 cc
- 3241 cars are of manual type and remaining 1858 cars are automatic transmission Type
- 4151 cars are individually registered, 114 cars are coporate registered whereas remaining 834 cars registration is not available.

b) Bivariate Analysis

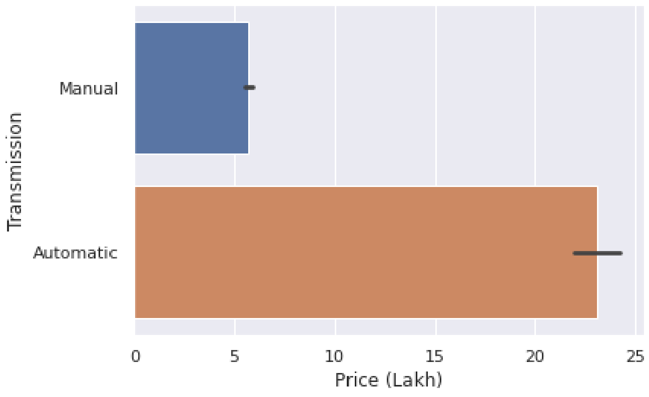




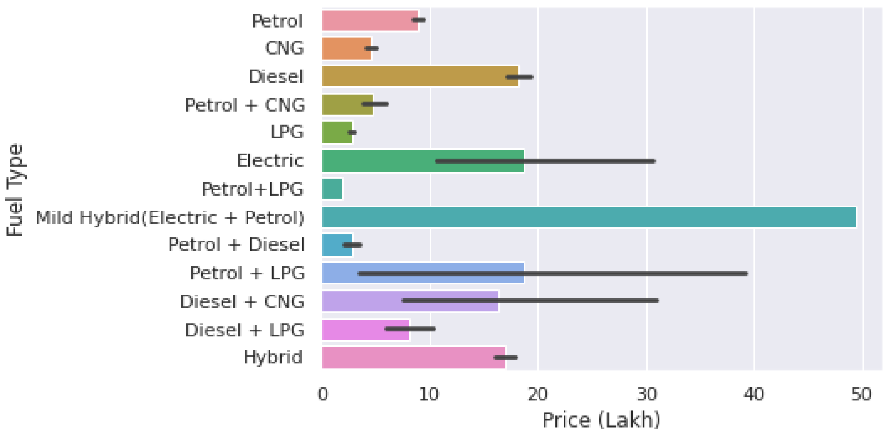




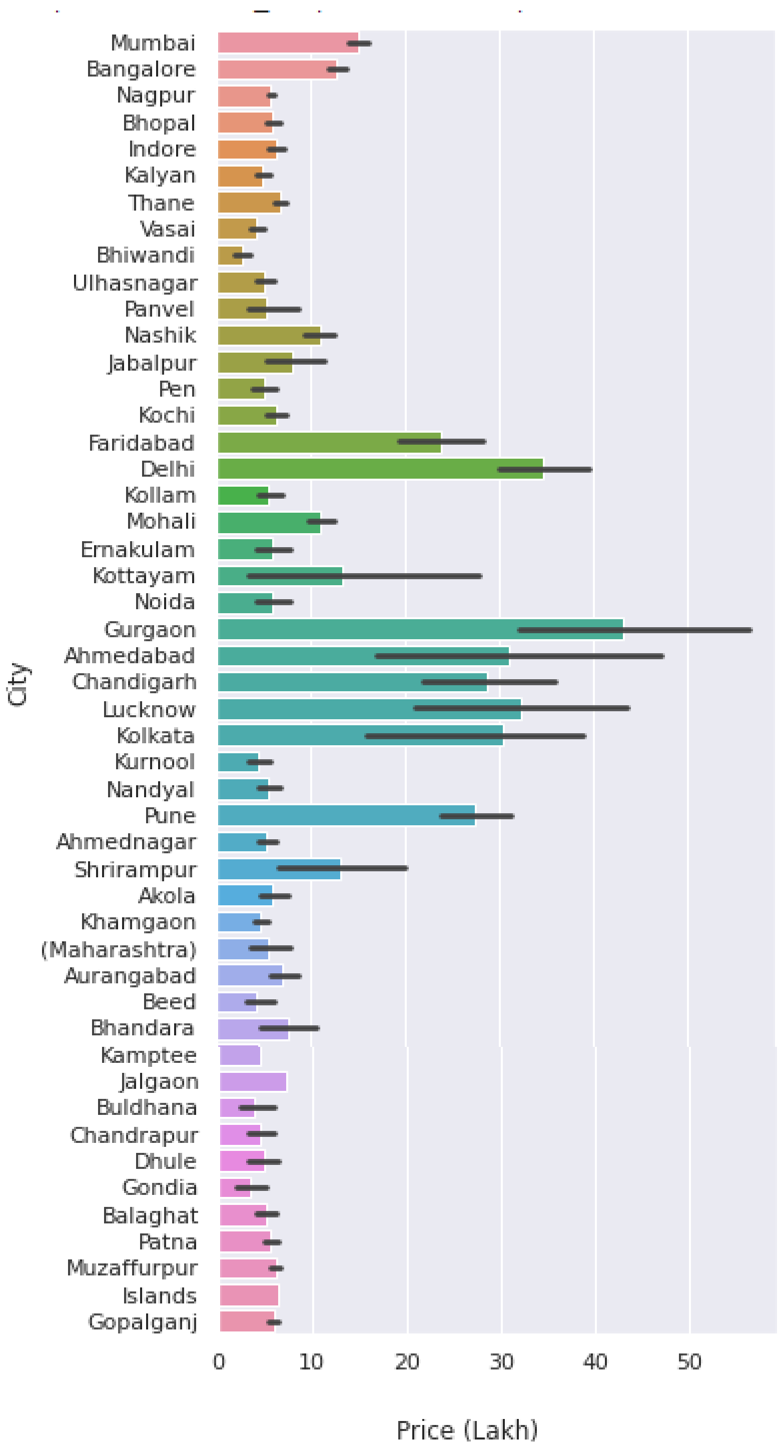
Comprehensive
Not Available
Third Party
Expired
Insurance Type
0
2
4
6
8
10
12
14
16
Price (Lakh)



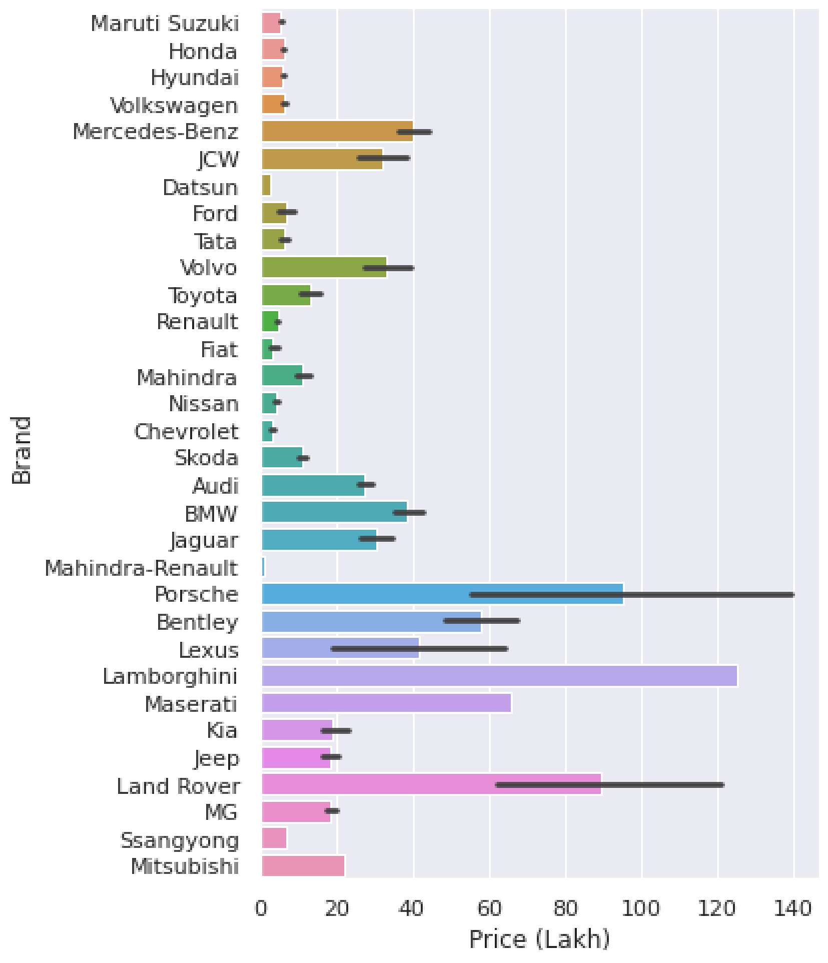
Manual
Automatic
Transmission
0
5
10
15
20
25
Price (Lakh)



Petrol
CNG
Diesel
Petrol + CNG
LPG
Electric
Petrol+LPG
Mild Hybrid(Electric + Petrol)
Petrol + Diesel
Petrol + LPG
Diesel + CNG
Diesel + LPG
Hybrid
Fuel Type
0
10
20
30
40
50
Price (Lakh)



Mumbai
Bangalore
Nagpur
Bhopal
Indore
Kalyan
Thane
Vasai
Bhiwandi
Ulhasnagar
Panvel
Nashik
Jabalpur
Pen
Kochi
Faridabad
Delhi
Kollam
Mohali
Ernakulam
Kottayam
Noida
Gurgaon
Ahmedabad
Chandigarh
Lucknow
Kolkata
Kurnool
Nandyal
Pune
Ahmednagar
Shrirampur
Akola
Khamgaon
(Maharashtra)
Aurangabad
Beed
Bhandara
Kamptee
Jalgaon
Buldhana
Chandrapur
Dhule
Gondia
Balaghat
Patna
Muzaffurpur
Islands
Gopalganj
City
0
10
20
30
40
50
Price (Lakh)



- There are total of 5099 used cars of 32 brands with 1355 different models.
- Lamborghini is most expensive brand while Mahindra-Renault is the least expensive brand.
- Automatic cars are more expensive than cars with manual transmission.
- Gurgaon has the highest selling price among all the cities.
- Mild Hybrid fuel type cars are the most expensive whereas petrol+LPG fuel type cars are least expensive.
- Cars with expired insurance are cheaper than comprehensive insurance type.
- Cars with unregistered previous owner are more expensive than cars with previous owner.

CONCLUSION

- Key Findings and Conclusions of the Study

  - In this study we found that bagging regression algorithm performs slightly better than random regression and rest of the algorithm tested.
  - Length, width, wheelbase, engine, kerb weight, manufacturing year, fuel capacity, max power, max torque are the top features of the used cars that highly impact the sale price among all the features in the dataset.
  - Gurgaon followed by Delhi are the cities with highest selling price.
  - 64% of the used cars are manual and remaining 36% are automatic transimission type.
  - 84% of used cars are registered individually when compared to corporate registered which are as low as 2%.

- Limitations of this work and Scope for Future Work

  Although we were successfully able to find the top features that contributed to the sale price, our best saved model has mean absolute error of 1.05, thus indicating that the margin of difference between actual price and the predicted price is around 1.05 lakh Indian Rupees. The model has potential to improve in future if we use bigger dataset with more features to train the model.