FLIP ROBO

11/11/2021

Used car price prediction

Report

Submitted by:

Sharukh Ansari

Acknowledgement

I'm extremely grateful to SME Khushboo Garg who give me an opportunity to work on this project and provide all the data required and information about the dataset which is crucial for completing this project and guided us in every possible manner. I'd also like to show my gratitude to live Data trained support who help me with this project and guided me in taking care of the skewness.

Introduction

Business problem

The price of a brand-new car in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But, due to the increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and its value in the present-day scenario. **In fact, seller** also has no idea about the car's existing value or the price he should be selling the car at. To overcome this problem, we have developed a model which will be highly effective. Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value. Because of which it will be possible to predict the actual price a car rather than the price range of a car.

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in India. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models. We will compare the performance of various machine learning algorithms like RandomForestRegressor, GradientBoostingRegressor, XGBRegressor, BaggingRegressor and AdaBoostRegressor. Depending on various parameters we will determine the price of the car. Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been developed which acquires input from any user and displays the Price of a car according to user's inputs.

Review of Literature: -

Predicting the price of Used Car Using Machine Learning Techniques. In this we are investigating the application of supervised machine learning techniques to predict the price of used cars in India. The predictions are based on historical data collected from used car selling Websites (Cardekho.com). Different techniques like Random Forest Regressor, Gradient Boosting Regressor, XGBRegressor, AdaBoostRegressor and Bagging Regressor have been used to make the predictions. To build a model for predicting the price of used cars we have used Supervised machine learning. Here we have used all the ensemble techniques to predict the price of Used cars.

Motivation:-

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

Objective: -

In this project we are predicting the price of used car. In this project we are going to build multiple models and check for the r2_score and cross validation score and select the best model based on difference between them and tune that model to increase the r2_score with different parameters

Mathematical/statistical Modelling Analysis: -

In this Project the data has Outliers and skewness and before going ahead we need to take care to avoid overfitting and underfitting of model. For avoid underfitting or overfitting we need to make data normally distributed for that I have used interquartile and z score method.

Data Sources and their format:

For this project I have done web scrapping from cardekho to get the current market price of the used cars so that our model can understand and have the idea what price trend currently in the market and predict the accurate price as much as possible. After scrapping I have saved all the data by creating Data frame in csv format.

Data pre-processing: -

In this section we are going to explore the dataset which we get from scrapping and clean it for model building. But first we need to check what columns we get after scrapping and their missing values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7193 entries, 0 to 7192
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         7193 non-null   int64
 1   Engine             7184 non-null   object
 2   Brand Name         7185 non-null   object
 3   Location           7193 non-null   object
 4   Driven             7185 non-null   object
 5   Insurance          7183 non-null   object
 6   Owner              7185 non-null   object
 7   Transmission       7185 non-null   object
 8   Milege             7185 non-null   object
 9   Fuel Type          7176 non-null   object
 10  Max Power(In bhp)  7164 non-null   object
 11  Price              7176 non-null   object
dtypes: int64(1), object(11)
memory usage: 674.5+ KB
```

From above we can see clearly that there is **7193 rows and 12 columns** and few columns has missing values

There are only 1 numerical column which is Unnamed: 0 and remaining columns are of object datatype. Where as one of them is **Price.** Which is also our dependent or target column.

Data Preparation: -

We are going to drop all the rows which have missing values. And split few columns to make new columns and clean some unwanted values or character from the columns to make data clean to process further

```
# Finding Out the garbage value
for i in df['Engine']:
    if "CC" not in i:
        print(i)
```

```
17 x 7.5J
103.25bhp
5
6
6
6
5
17 x 7.5J
17 x 7.5J
8
17 x 7.5J
17 x 7.5J
17 x 7.5J
```

Above you can see that Engine Columns contains some garbage Values and we need to remove that row.

```
# NOw removing that garbage values
for i in ['17 x 7.5J','5','6','8','103.25bhp']:
    index = np.where(df['Engine']==i)
    df.drop(df.index[index],inplace=True)
```

```
df['Engine'] = df['Engine'].str.split().str[0]
df['Engine']=df['Engine'].astype('int')

df["Make Year"] = df['Brand Name'].str.split().str[0]
df['Make Year']=df['Make Year'].astype('int')

df['Model'] = df['Brand Name'].str.split(" ").str[2]

df['Variant'] = df['Brand Name'].str.split(" ").str[4:]

df['Manufacturer Name'] = df['Brand Name'].str.split().str[1]

df['Driven'] = df['Driven'].str.split().str[0]
df['Milege'] = df['Milege'].str.split().str[0]
```

```
price = list(df['Price'])
for i in range(0,len(price)):
    if "Cr*" in str(price[i]):
        price[i]= price[i].replace('Cr*',"").replace(" ","")
        price[i]= (float(price[i])*10000000)
```

```
for i in range(0,len(price)):
    if "Lakh*" in str(price[i]):
        price[i]=price[i].replace('Lakh*',"").replace(" ","")
        price[i] = pd.to_numeric(price[i],errors='coerce')
        price[i] = price[i]*100000
```

After analysing Engine columns, we have found out that it has some garbage values which we drop.

Feature engineering: -

Make Year: - we have created this new feature from Brand name by extracting the manufacturing year present just Infront of the Brand Name.

Model: - We have created new feature name Model which contain the model's name of all the cars present inside our dataset by using split.

Manufacturer Name: - We have extracted Manufacturer name from Brand Name.

Variant: - we have separated the Variant name present in Brand name by using split function.

Driven: - we need to remove "**kms**", and "**,**" present between total Driven to make this columns completely numerical.

Mileage: - We must remove km/pl and "**,**" to make it completely numerical.

Price: - In this columns we have price present in decimal, and remove the unwanted punctuation and symbol to make that column numerical

```
price = list(df['Price'])
for i in range(0,len(price)):
    if "Cr*" in str(price[i]):
        price[i]= price[i].replace('Cr*',"").replace(" ","")
        price[i]= (float(price[i])*10000000)
```

```
for i in range(0,len(price)):
    if "Lakh*" in str(price[i]):
        price[i]=price[i].replace('Lakh*',"").replace(" ","")
        price[i] = pd.to_numeric(price[i],errors='coerce')
        price[i] = price[i]*100000
```

```
for i in range(0,len(price)):
    price[i]= str(price[i]).replace("₹ ","").replace(",","").replace("*","")

df['Price'] = price
df['Price'] = pd.to_numeric(df['Price'],errors='coerce')
```

```
variant = list(df['Variant'])
for i in range(0,len(variant)):
    if variant[i]==[]:
        variant[i] = np.nan
    else:
        variant[i] = ",".join(variant[i])


df['Variant']=variant
```

Engine :- In this columns we know that the engine power is always measure in bhp but we need to remove the bhp and all those rows which have garbage values(not in bhp).

```
variant = list(df['Variant'])
for i in range(0,len(variant)):
    if variant[i]==[]:
        variant[i] = np.nan
    else:
        variant[i] = ",".join(variant[i])


df['Variant']=variant
```

```
# Dropping all the garbage values
unmatch = []
for i in df['Max Power(In bhp)']:
    if "bhp" not in i:
        if i not in unmatch:
            unmatch.append(i)

for i in unmatch:
    index = np.where(df['Max Power(In bhp)']==i)
    df.drop(df.index[index],inplace=True)
```

```
df['Max Power(In bhp)'] = df['Max Power(In bhp)'].str.replace("bhp","").replace(" ","")
```

```
df['Driven'] = df['Driven'].str.replace(",","")
df['Variant'] = df['Variant'].str.replace(","," ")
```

```
df.drop(columns='Brand Name',axis=1,inplace=True)
```

Variant: - We need to remove Empty list with “nan” and join the remaining which have values in variant column.

Now dataset is completely clean and ready for further analysis and now I need to check again for null values and datatype of the columns.

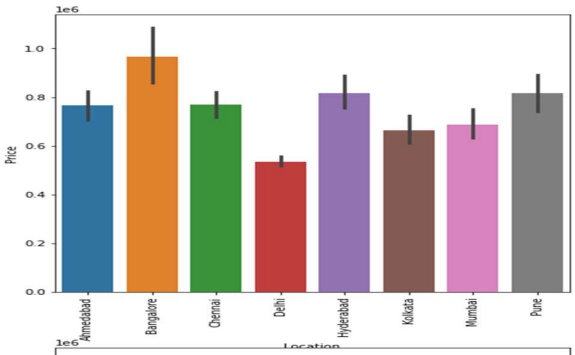
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7045 entries, 0 to 7192
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         7045 non-null   int64
 1   Engine             7045 non-null   int32
 2   Location           7045 non-null   object
 3   Driven             7045 non-null   int32
 4   Insurance          7045 non-null   object
 5   Owner              7045 non-null   object
 6   Transmission       7045 non-null   object
 7   Milege             7045 non-null   float32
 8   Fuel Type          7045 non-null   object
 9   Max Power(In bhp)  7045 non-null   float32
 10  Price              7045 non-null   float64
 11  Make Year          7045 non-null   int32
 12  Model              7045 non-null   object
 13  Manufacturer Name  7045 non-null   object
dtypes: float32(2), float64(1), int32(3), int64(1), object(7)
memory usage: 688.0+ KB
```

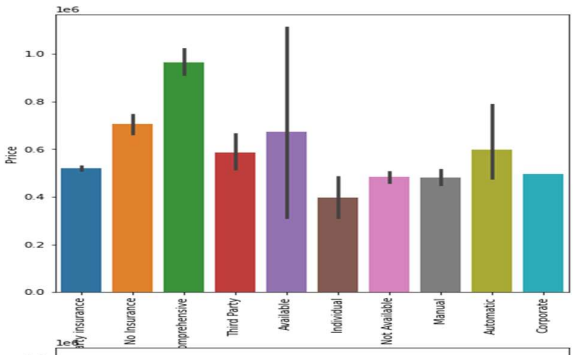
From above we can see that there are 7 numerical column and 7 object data type column. Now the dataset has 7045 rows and 14 columns.

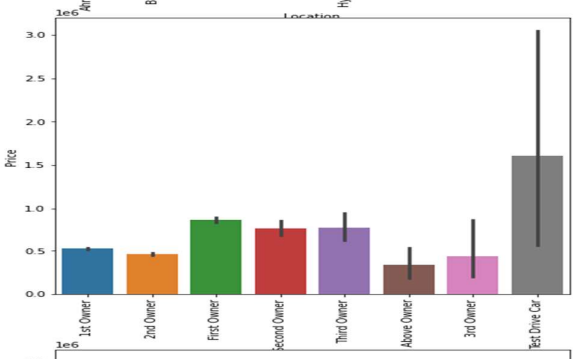
	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	7045.0	3593.354578	2076.815814	0.0	1797.000000	3591.000000	5392.000000	7192.0
Engine	7045.0	1408.245848	474.324649	0.0	1197.000000	1248.000000	1498.000000	5461.0
Driven	7045.0	59011.627253	43705.529241	1.0	33405.000000	55000.000000	76843.000000	1019000.0
Milege	7045.0	19.748520	4.100941	0.0	17.209999	19.870001	22.320000	36.0
Max Power(In bhp)	7045.0	96.839470	40.135265	32.5	74.000000	85.000000	105.940002	641.0
Price	7045.0	718589.422569	921547.492618	30000.0	356000.000000	515000.000000	720000.000000	30800000.0
Make Year	7045.0	2014.971043	3.147295	1986.0	2013.000000	2015.000000	2017.000000	2021.0

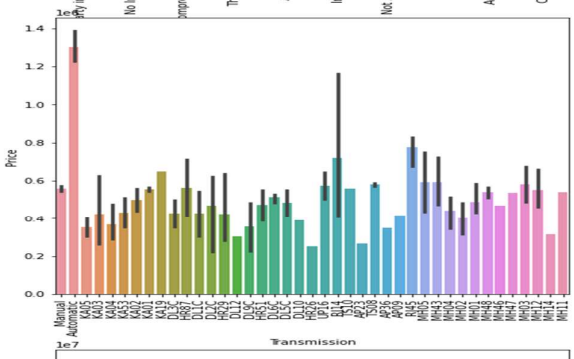
In above image we can see the basic statistics of the all the numerical column.

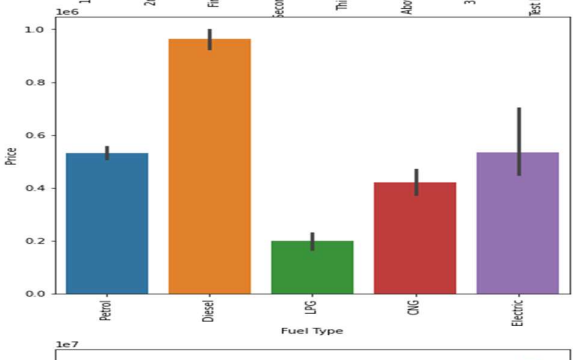
Let’s see the distribution of the data.

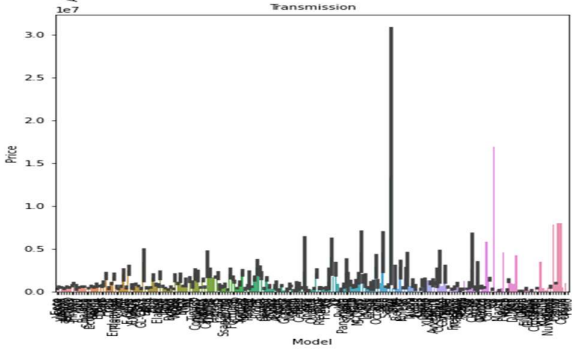


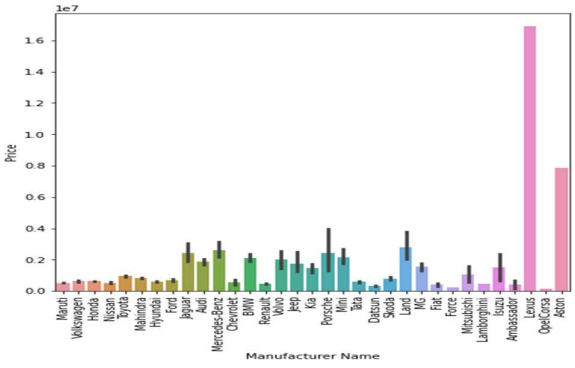




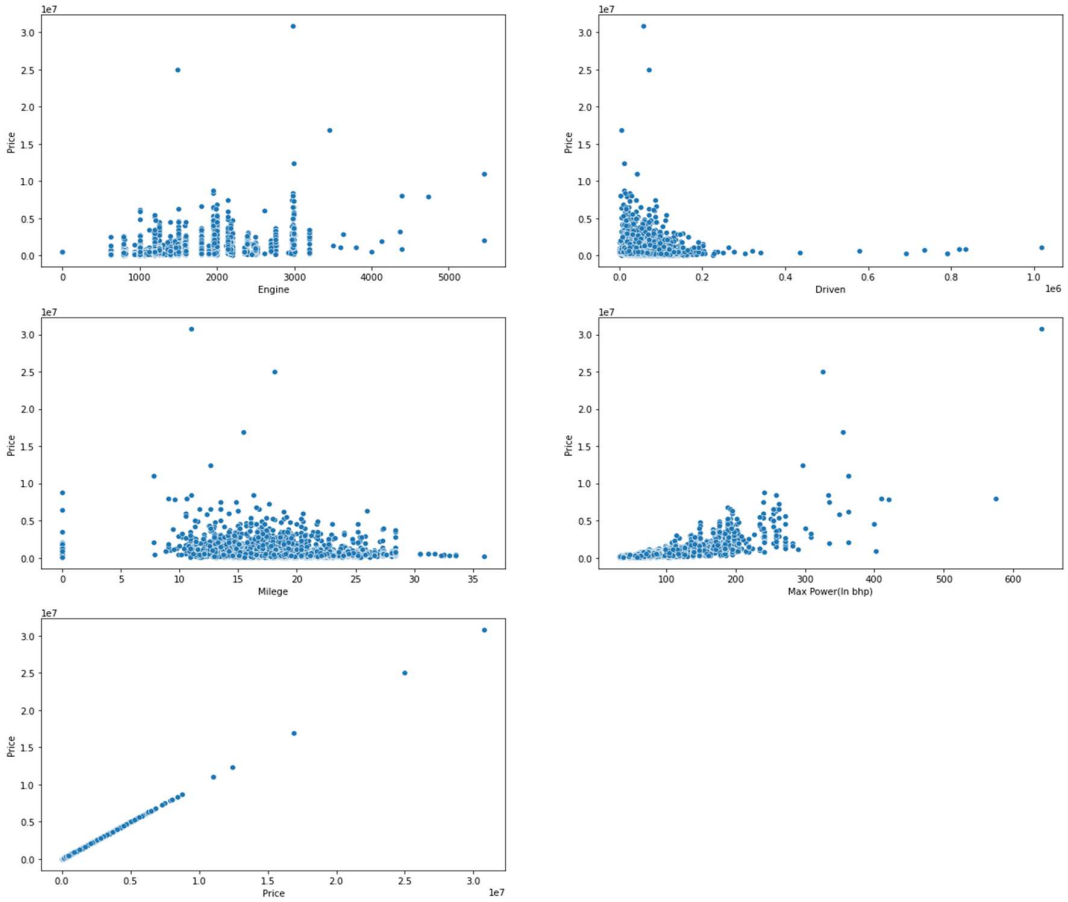








- Used car price is high in Bangalore. whereas Delhi car has lowest Price as compared to other location.
- The car which has comprehensive insurance has a slightly higher price than other car which have third party or other type of insurance.
- The Diesel Model of a car has high price comparatively any other car which use different fuel type.
- The Lexus and auton brand car have slightly high price as compare to other brand and Maruti and Force comes under Budget cars.



It seems that all the columns have Outliers and none of the columns is showing any linear relation with the target Variable.

Encoding the categorical columns: -

```
df['Insurance'] = df['Insurance'].replace("No Insurance",0).replace("Not Available",0)
df['Insurance'] = df['Insurance'].replace("Comprehensive",1).replace("Comprehensive",1)
df['Insurance'] = df['Insurance'].replace("Individual",2)
df['Insurance'] = df['Insurance'].replace("Corporate",4)
```

```
df['Owner'] = df['Owner'].replace("1st Owner",1).replace("First Owner",1)
df['Owner'] = df['Owner'].replace("2nd Owner",2).replace("Second Owner",2)
df['Owner'] = df['Owner'].replace("Third Owner",3).replace("3rd Owner",3)
df['Owner'] = df['Owner'].replace("Fourth & Above Owner",4)
df['Owner'] = df['Owner'].replace("Test Drive Car",0)
```

```
df['Transmission']=np.where(df["Transmission"]=="Manual",0,1)
```

```
df['Fuel Type'].unique()
```

```
array(['Petrol', 'Diesel', 'LPG', 'CNG', 'Electric'], dtype=object)
```

```
df['Fuel Type'] = df['Fuel Type'].replace("Diesel",1)
df['Fuel Type'] = df['Fuel Type'].replace("LPG",2)
df['Fuel Type'] = df['Fuel Type'].replace("CNG",3)
df['Fuel Type'] = df['Fuel Type'].replace("Electric",4)
df['Fuel Type'] = df['Fuel Type'].replace("Petrol",0)
```

I have Encode Insurance, Owner and Fuel Type columns manually.

Using Ordinal Encoder to encode Manufacturer and Model column: -

```
from sklearn.preprocessing import OrdinalEncoder
# Separating categorical and numerical columns
cat_df = df.select_dtypes(include='object')
```

```
num_df = df.select_dtypes(include=np.number)
```

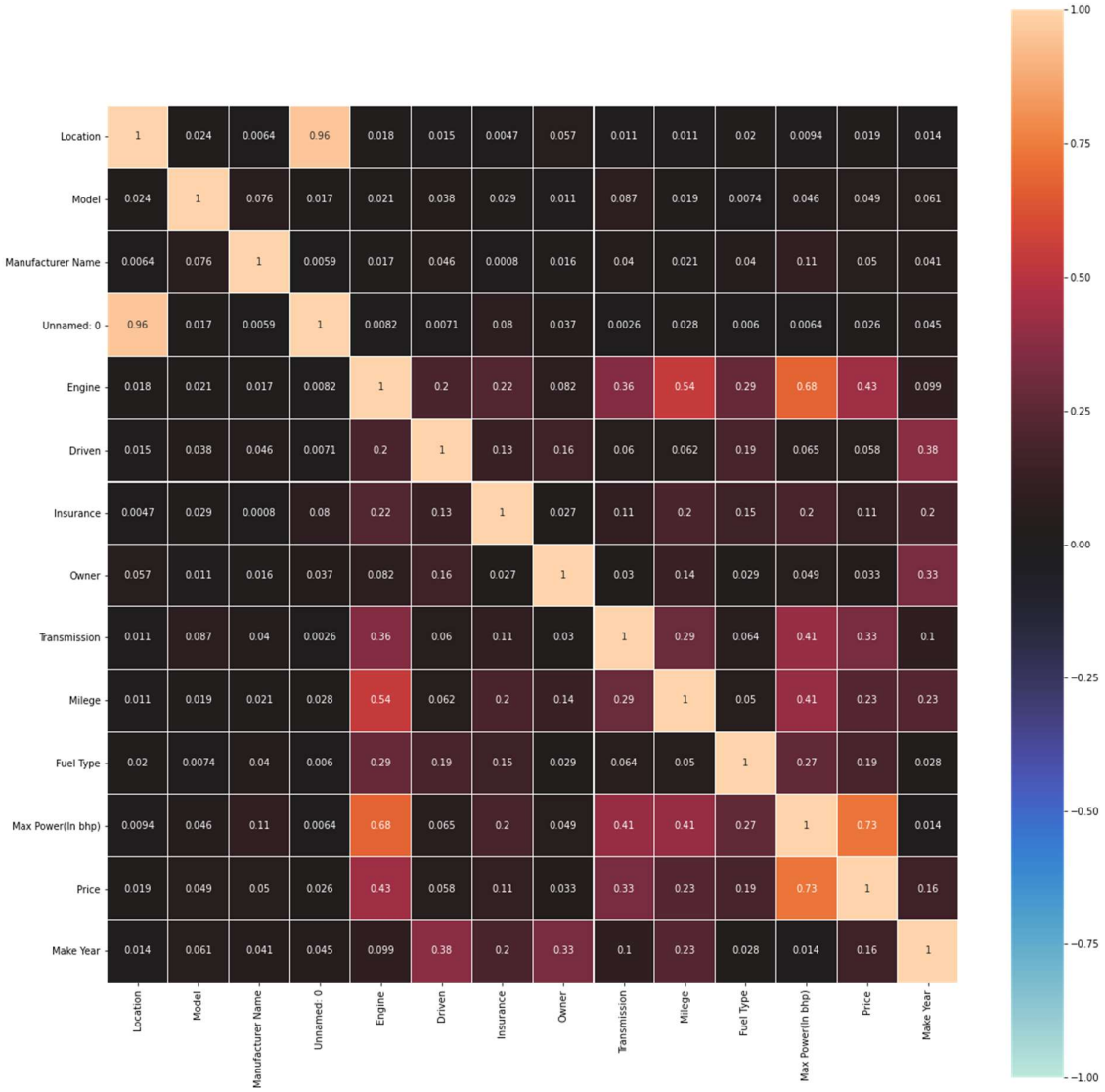
```
encoding = OrdinalEncoder()
for i in cat_df:
    cat_df[[i]] = encoding.fit_transform(cat_df[[i]])
```

```
cat_df = cat_df.astype('float')
```

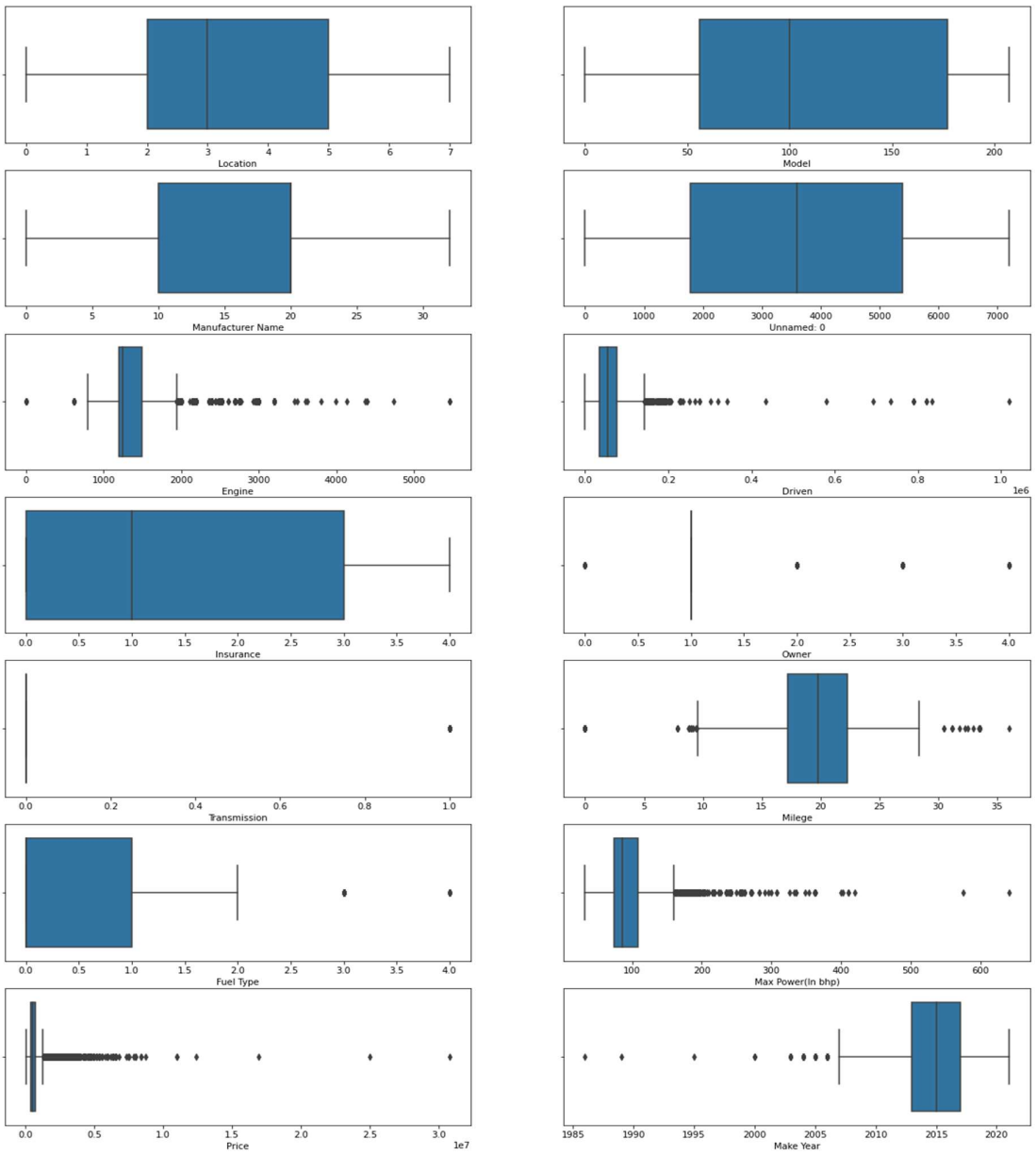
```
final_df = pd.concat([cat_df,num_df],axis=1)
```

Here I have encode remaining column using Ordinal encoder and then concat it with the numerical to form the final dataset.

Checking the correlation using the correlation heatmap: -



From above heat map we can see that only Max Power (In bhp) has strong correlation with the target variable and rest of the column doesn't have that strong relation with the target variable.



Handling Outliers and skewness: -

Dirven, Engine and Mileage are the Continuous columns and all of them is having Outliers. Whereas Price also has outlier, but it is out target columns.

Mileage column: -

I am using Interquartile method to remove Outliers from the mileage columns because from above boxplot we can see that mileage columns also contain "0" and we all ware that mileage of any car

cannot be zero. Therefore, i am using Interquartile method to drop upper and lower outliers of mileage using Interquartile method.

```
lower = q1['Milege']-(1.5*iqr['Milege'])
index = np.where(final_df['Milege']<lower)
final_df.drop(final_df.index[index],inplace=True)
```

```
upper = q3['Milege']+(1.5*iqr['Milege'])
index = np.where(final_df['Milege']>upper)
final_df.drop(final_df.index[index],inplace=True)
```

Z score method: -

```
from scipy.stats import zscore
```

```
zscr = zscore(final_df[['Engine','Driven','Max Power(In bhp)']])
abs_zscr = np.abs(zscr)
filtered_entries = ((abs_zscr < 2.7 )&(abs_zscr > -2.7 )).all(axis=1)
new_df = final_df[filtered_entries]
```

By applying z score at 2.7 I tried to remove the Outliers as much as possible so that data become normally distributed, and we know that 99.9% data lies between -3 to +3. But here i have tried different threshold value to remove Outlier and reduce the skewness from the data as much as possible

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
```

```
new_df[["Driven","Engine","Max Power(In bhp)"]] = pt.fit_transform(new_df[["Driven","Engine","Max Power(In bhp)"]])
```

```
new_df.skew()
```

```
Location             0.197536
Model               -0.043693
Manufacturer Name    0.168663
Unnamed: 0           0.002894
Engine              -0.013366
Driven              -0.057288
Insurance            0.186719
Owner                2.203492
Transmission         1.560312
Milege               0.068993
Fuel Type            0.643225
Max Power(In bhp)   -0.022128
Price                4.100998
Make Year           -0.834065
dtype: float64
```

I have used PowerTransformation method to remove skewness present in continuous dependent columns.

Machine algorithm only understand the numerical data that's why we convert all the columns in to numerical by encoding all the categorical column and we can see that only Max Power (In bhp) has some sort of linear relation with the target variable.

Requirement: -

Hardware requirements

Operating system- Windows 7,8,10
Processor- dual core 2.4 GHz (i5 or i7 series Intel processor or equivalent AMD)
RAM-4GB

Software Requirements

Jupyter Notebook
Chrome
Python 3.6 or higher

There are two primary phases in the system:
1. Training phase: The system is trained by using the data in the data set and fits a model (line/curve) based on the algorithm chosen accordingly.

2. Testing phase: the system is provided with the testing data, and it is tested for its working. The accuracy is checked. And therefore, the data that is used to train the model or test it, must be appropriate. The system is designed to detect and predict price of used car and hence appropriate algorithms must be used to do the two different tasks. Before the algorithms are selected for further use, different algorithms were compared for its accuracy. The well-suited one for the task was chosen.

Here we have two major problems first one cleaning the data and second one is to remove skewness and take care of outliers.
For cleaning data we have used **split** method and for taking care of outliers there is two method one of them is **z score and other one is interquartile method**. Both the method is explained above.

We have use multiple ensemble regression technique to predict the price of used cars. But before passing down the data for training and testing we have to split the data into target and dependent variable.

```
x = new_df.drop(columns=["Price","Unnamed: 0"],axis=1)
y = new_df["Price"]
```

Model Building: -

1.Bagging Regressor is an ensemble technique used for predict the complicate relationship with the target variable. And it giving the accuracy of 84% and cross validation score of 80%

```
bag_reg = BaggingRegressor()
bag_reg.fit(x_train,y_train)
```

```
BaggingRegressor()
```

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

```
diff=[]
r2score=[]
cross=[]
cv=cross_val_score(bag_reg,x_scaled,y,cv=kf).mean()
print(" R2 score :",r2_score(y_test,y_pred),"\n","="*60,"\n","cross Validation score :",
      cv)
```

```
R2 score : 0.8419870425168285
============================================================
cross Validation score : 0.8037695503803018
```

2. Random Forest Regression: -Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

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

```
RandomForestRegressor()
```

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

```
cv=cross_val_score(rf,x_scaled,y,cv=kf).mean()
print(" R2 score :",r2_score(y_test,y_pred),"\n","="*60,"\n","cross Validation score :",
      cv)
```

```
 R2 score : 0.8492563825605572
 ============================================================
 cross Validation score : 0.8189959697341509
```

3. AdaBoost is one of the first boosting algorithms to be adapted in solving practices. Adaboost helps you combine multiple “weak classifiers” into a single “strong classifier”. AdaBoost algorithms can be used for both classification and regression problem.

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

```
AdaBoostRegressor()
```

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

```
cv=cross_val_score(ada,x_scaled,y,cv=kf).mean()
print(" R2 score :",r2_score(y_test,y_pred),"\n","="*60,"\n","cross Validation score :",
      cv)
```

```
 R2 score : 0.720747653699487
 ============================================================
 cross Validation score : 0.5714046504074421
```

It is giving too much difference between the R2 score and cross validation score.

4. XGBoost is an algorithm that has recently been dominating applied machine learning. Xgboost is used with the complex relation dataset. It gives the good accuracy with the dataset has complex relation.

```
xgb = XGBRegressor()
xgb.fit(x_train,y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

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

```
cv=cross_val_score(xgb,x_scaled,y,cv=kf).mean()
print(" R2 score :",r2_score(y_test,y_pred),"\n","="*60,"\n","cross Validation score :",
      cv)
```

```
 R2 score : 0.8723308703401611
 ============================================================
 cross Validation score : 0.8233379932022368
```

It is giving us the good accuracy but the difference between the cross validation and R2 score is around 5 digit.

5. Gradient Boosting: -Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

```
gbt= GradientBoostingRegressor()
gbt.fit(x_train,y_train)
```

```
GradientBoostingRegressor()
```

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

```
cv=cross_val_score(gbt,x_scaled,y,cv=kf).mean()
print(" R2 score :",r2_score(y_test,y_pred),"\n","="*60,"\n","cross Validation score :",
      cv)
```

```
 R2 score : 0.8346642469291911
 ============================================================
 cross Validation score : 0.8047095459019674
```

Key metrics: - Here I have used r2_score and cross validation score to choose the best model. Here the model which is giving me the least difference between the cross validation and r2 score is considered as the best model. Here that model is Gradient Boosting. Below you can see all the accuracy and their cross-validation score

	Model	R2 score	Cross validation score	Difference
0	Bagging Regressor	0.841987	0.803770	0.038217
1	RandomForest Regressor	0.849256	0.818996	0.030260
2	AdaBoostRegressor	0.720748	0.571405	0.149343
3	XGB Regressor	0.872331	0.823338	0.048993
4	GradientBoosting Regressor	0.834664	0.804710	0.029955

Here from the above you can see that the difference of Gradient boosting between r2 score and cross validation is least. So I am going to tune that.

Hyperparameter Tunning using GridSearchCv: -

```
grid_search = GridSearchCV(gbt,param_grid=params,n_jobs=-1)
```

```
grid_search.fit(x_train,y_train)
```

```
GridSearchCV(estimator=GradientBoostingRegressor(), n_jobs=-1,
             param_grid={'learning_rate': [0.1, 0.001, 0.0001, 0.2, 0.5],
                         'max_depth': [3, 5, 6],
                         'n_estimators': [100, 10, 50, 200, 500],
                         'random_state': [266, 177, 216]})
```

```
gbt = GradientBoostingRegressor(learning_rate=0.1,n_estimators=500,max_depth=3,random_state=216)
gbt.fit(x_train,y_train)
```

```
GradientBoostingRegressor(n_estimators=500, random_state=216)
```

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

```
r2_score(y_test,y_pred)
```

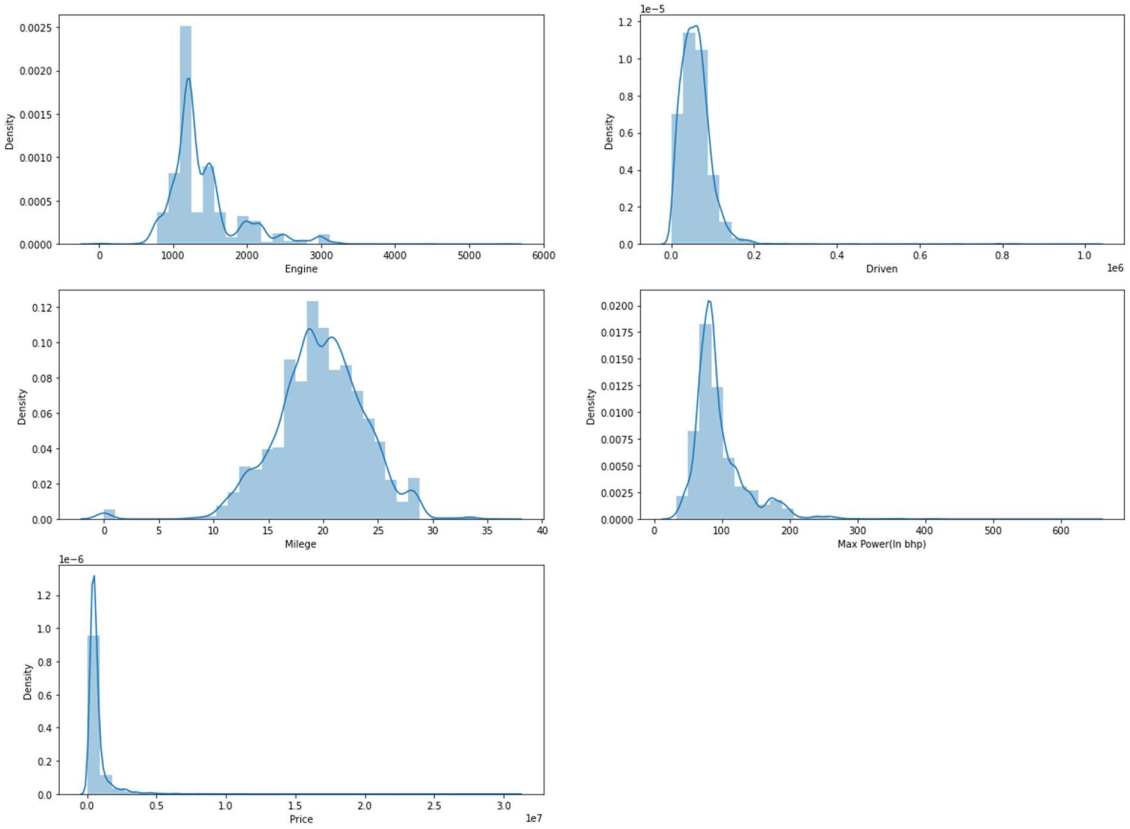
```
0.8633456702127562
```

```
cross_val_score(gbt,x_scaled,y,cv=kf).mean()
```

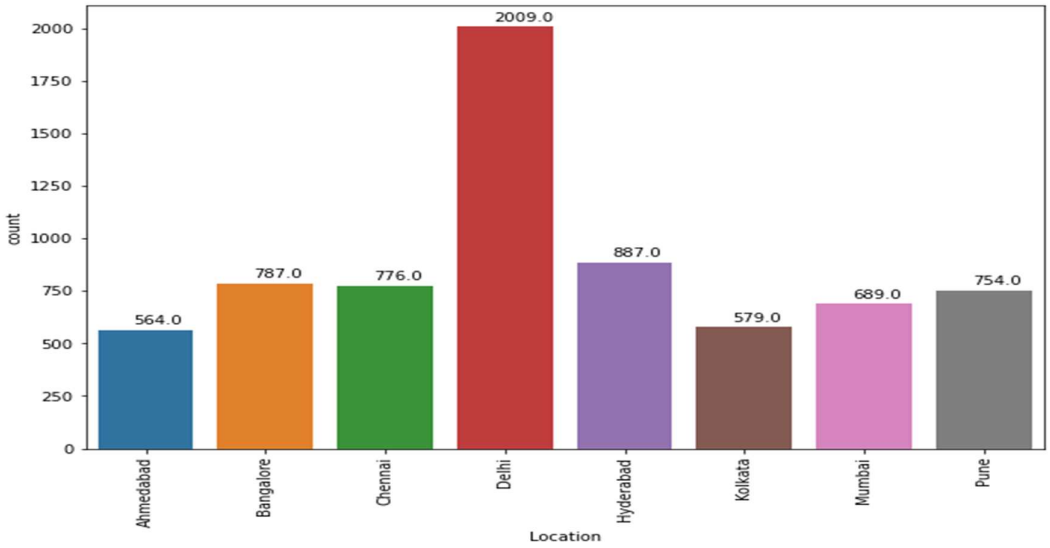
```
0.8420568829043521
```

After Hyperparameter tunning the accuracy of the model is increased by 2%.

Visualisation: -

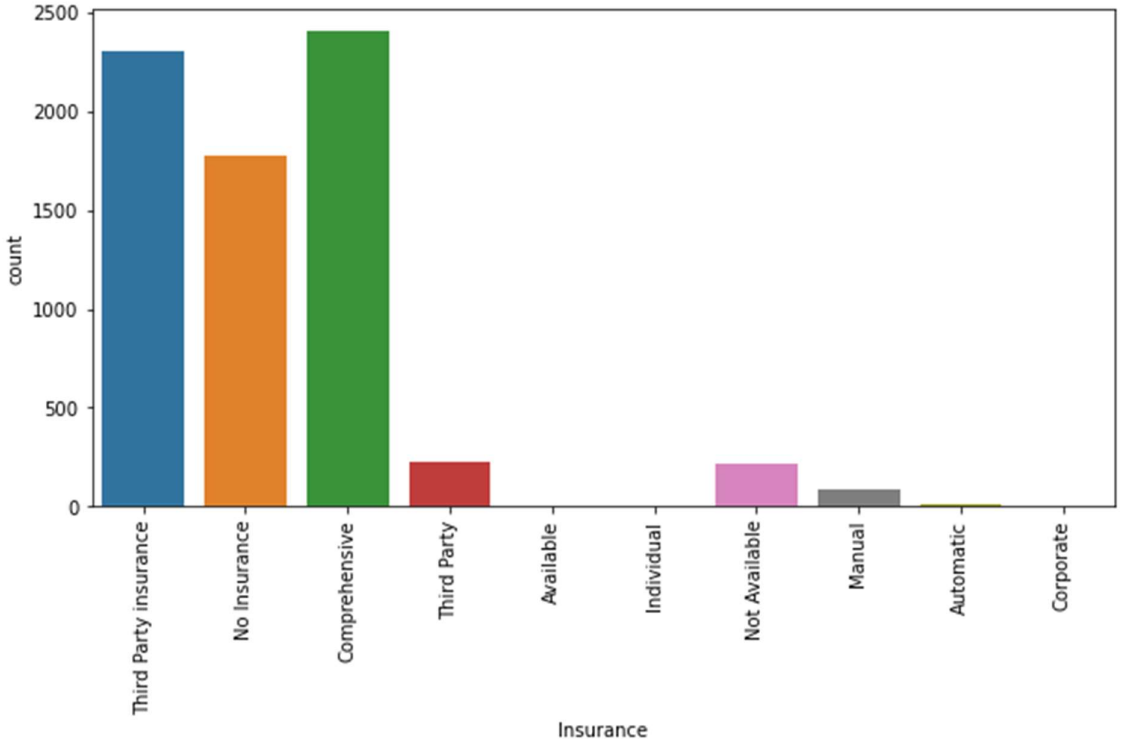


We are checking the distribution of the all the numerical columns using distplot. From distplot we can check whether data is normally distributed or not. It also shows whether data has skewness or not.

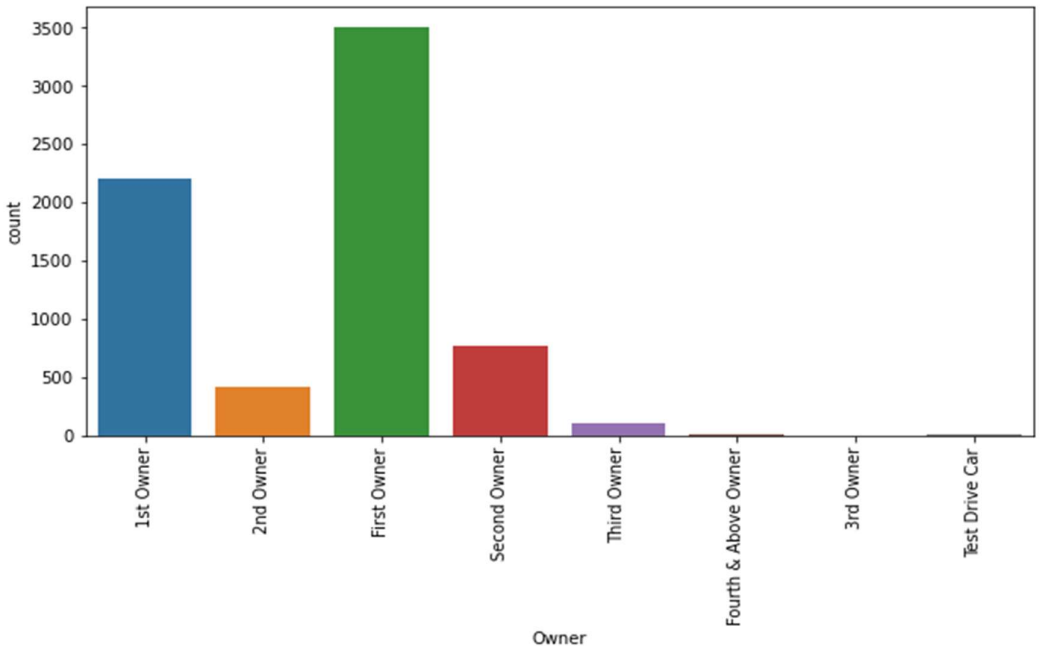


It shows the no. of rows present in the dataset location wise.

Insurance:-



There are major four type of insurance Present Third party, Comprehensive, individual and Corporate. and here some garbage values present like Manual and Automatic.



From above we can check the no. or unique present inside the data.

Interpretation of Result:

From above all the graphs we can check distribution of the data or relation with the target variable of a particular column by visualisation. It is the best method rather than by doing any statistical test to check whether any column is related to target variable.

Conclusion: -

The increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. The proposed system will help to determine the accurate price of used car price prediction. This report compares 5 different algorithms for machine learning : Bagging Regressor, Gradient Regressor, RandomForest Regressor, XGBRegressor and AdaBoostRegressor.

Future scope: -

In this project we have less data and very less no. of features. In future this machine learning model may bind with various website which can provide real time data for price prediction. Also we may add large historical data and large number of features of car price which can help to improve accuracy of the machine learning model. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset.

References:

[1] Enis gegic, Becir Isakovic, Dino Keco, Zerina Masetic, Jasmin Kevric, "Car Price Prediction Using Machine Learning"; (TEM Journal 2019)

[2] Ning sun, Hongxi Bai, Yuxia Geng, Huizhu Shi, "Price Evaluation Model In Second Hand Car System Based On BP Neural Network Theory"; (Hohai University Changzhou, China) [4] Nitis Monburinon, Prajak Chertchom, Thongchai Kaewkiriya, Suwat Rungpheung, Sabir Buya, Pitchayakit Boonpou, "Prediction of Prices for Used Car by using Regression Models" (ICBIR 2018)