
# CAR PRICE PREDICTION PROJECT

Submitted by:

Asheem Siwach

# ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Mr. Shubham Yadav for his constant guidance and support.

Reference sources are :-

- Google
- Stackoverflow.com
- Analytics vidhya
- Notes and repository from DataTrained

# INTRODUCTION

- ## Business Problem Framing

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

- ## Review of Literature

  The goal of this project is to build a model which can be used to predict the car price using various features available. This project is more about data exploration, finding the better insights from data and using the skills and techniques to build an efficient model which can predict the prices after the recession faced during covid-19 crisis. Since we scrape a good amount of data that are related to the price of car, we can do better data exploration and derive some interesting features using the available columns.

# Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

In this project we have created a model to predict the "SalePrice" of cars which are for resale in Used Cars Market. So "SalePrice" is our target feature and it is continuous in nature. Therefore we are dealing with Regression type of problem.

This project is based on 2 main phases –
1). Data Collection
2). Model Building

Data Collection Phase –
You have to scrape at least 5000 used cars data. You can scrape more data as well, it's up to you.
More the data better the model
In this section You need to scrape the data of used cars from websites (Olx, cardekho, Cars24 etc.)
You need web scraping for this. You have to fetch data for different locations. The number of
columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometers, fuel, number of owners, location and at last target variable Price of the car. This data is to give you a hint about important variables in used car model. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.
Try to include all types of cars in your data for example- SUV, Sedans, Coupe, minivan, Hatchback.

Model Building :

After collecting the data, you 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.

Follow the complete life cycle of data science. Include all the steps like

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

- ## Data Sources and their formats

  Data sources for our model are online Car Re-sale websites like Car24. We scraped our data using the web scrapping tool Selenium. We scrapped almost 5000 rows of data and then saved all data into different excel files so that it can be used for model building.

  After saving into excel files we load it into our Jupyter notebook and now let's have a look on data formats –

After loading the data we checked data formats -

```
In [13]: # checking datatypes and other information
         dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4790 entries, 0 to 4789
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Model         4790 non-null   object
 1   History       4789 non-null   object
 2   Owner         4789 non-null   object
 3   Kms_driven    4789 non-null   float64
 4   Fuel_type     4789 non-null   object
 5   Transmission  4549 non-null   object
 6   Registration  4789 non-null   object
 7   Insurance     3626 non-null   object
 8   Price         4790 non-null   object
dtypes: float64(1), object(8)
memory usage: 336.9+ KB
```

- There is 2 type of data present - 1.) Float 2.) Object in nature
- Insurance and Transmission features contains null values & 1 row is present in dataset with null values in all features.

Only Kms_driven is a continuous variable and rest of all are categorical variables.

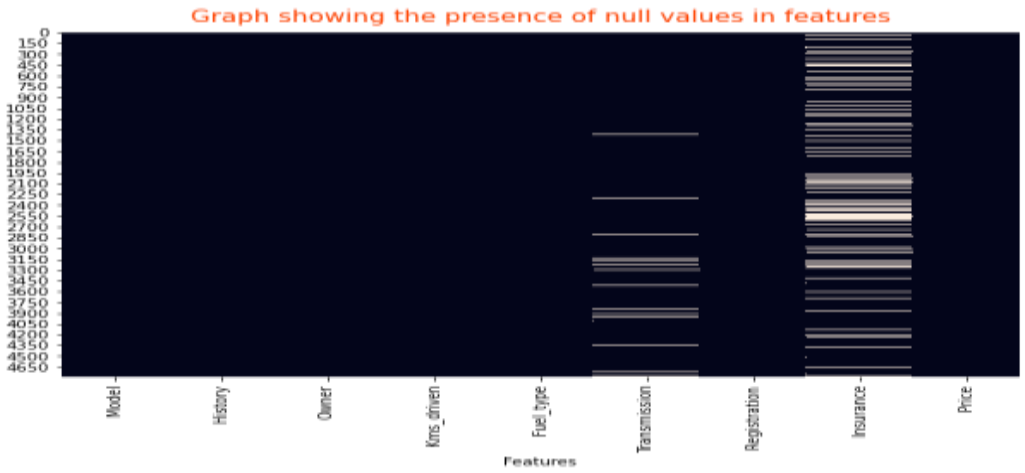
- ## Data Preprocessing Done

### 1. Checking Null Values:

Now after loading the data and checking the formats we moved to data preprocessing step. In this step we have to change data into proper formats and then do some data cleaning by treating null values.

We have used heatmap visualization technique to see the presence of null values in our dataset along with some method to find number of null values.

```
In [14]: # Checking null values using Heatmap
         for i in dataset:
             if dataset[i].isnull().sum()>0:
                 print("{} -> Having null values = {}".format(i,dataset[i].isnull().sum()))

         # Heatmap
         plt.figure(figsize=(10,6))
         sns.heatmap(dataset.isnull(),cbar=False)
         plt.title("Graph showing the presence of null values in features",color='orangered',pad=10,fontsize=15,loc='center')
         plt.xlabel("Features")
         plt.show()

History -> Having null values = 1
Owner -> Having null values = 1
Kms_driven -> Having null values = 1
Fuel_type -> Having null values = 1
Transmission -> Having null values = 241
Registration -> Having null values = 1
Insurance -> Having null values = 1164
```




Using the graph we can see that null values present in Transmission and Insurance variables.

## 2. Data Cleaning :

For data cleaning ,first we drop the duplicated values present in dataset.

```
In [ ]:
# After combining data let's check duplicated data
dataset.duplicated().sum()

Out[ ]:
419
```

Model Variable -After removing the duplicated values we splited the Model variable into many useful variable which gives exact information about one character.

```
In [19]:
# Now using split function finding out the followings

dataset['Make_yr']=dataset['Model'].str.split().str[0]      # make year of vehicle
dataset['Brand_name']=dataset['Model'].str.split().str[1]   # brand to which vehicle belongs
dataset['model']=dataset['Model'].str.split().str[2]        # model of vehicle
dataset['Variant']=dataset['Model'].str.split().str[3]      # variant of vehicle
```

```
In [20]:
# Let's convert make_yr into interger
dataset['Make_yr']=dataset['Make_yr'].astype(int)

# We can find one more feature showing the age of vehicle
dataset['Years_old']=2021-dataset['Make_yr']
```

Transmission Variable -We find out the most frequent value in Transmission variable and fill null values with that.

Manual transmission is most preferred

```
In [38]:
# There are only 2 types of transmissions either automatic or manual.
# So filling null values with most repeated type of transmission
dataset['Transmission'].fillna("MANUAL",inplace=True)
```

Registration Variable – In this variable some useless data was present ,so using the split method we extract only first 2 digit data which represent to which State car belongs.

```
In [32]:
# Based on the RTO we will extract the first 2 aplhabets  as these have some effect on the price
dataset['RTO']=dataset['Registration'].str.split("-").str[0]

dataset['RTO'].unique()
```

```
Out[32]: array(['DL', 'HR', 'CH', 'UP', 'PB', 'AS', 'GJ', 'WB', 'AP', 'MH', 'UK',
       'HP', 'KL', 'MA', 'MJ', 'KA'], dtype=object)
```

Insurance variable -Now we move towards Insurance variable and find out that there are some values with same meaning but different representation. Firstly we treated these values and again based on the most frequent value fill the null values.

```
In [34]:
# Check unique values in insurance_type feature
dataset['Insurance'].unique()
```

```
Out[34]: array(['Zero Depreciation', 'Third_party', 'Comp', 'Zero_Dep', nan,
       'Third Party', '3rd Party', 'Comprehensive'], dtype=object)
```

```
In [35]:
# Create dictionary to replace all values with these 3 values
val_to_replace={"Zero Depreciation":"Zero Depreciation","Zero_Dep":"Zero Depreciation" ,
                "Comprehensive":"Comprehensive","Comp":"Comprehensive",
                "Third Party":"Third Party","Third_party":"Third Party","3rd Party":"Third Party"}

# Mapping values to be replaced in list
dataset['Insurance']=dataset['Insurance'].map(val_to_replace)
```

```
In [37]:
# Let's see null values in Insurance feature
dataset['Insurance'].isnull().sum()
```

```
Out[37]: 1163
```

```
In [38]:
# Fill null values using the most repeated character
dataset['Insurance'].fillna("Third Party",inplace=True)
```

SalePrice Variable – In this variable due to some sign like "," ,"₹" it was not giving us continuous values. So we removed these signs and convert into continuous variable.

```
In [39]:
# As we have Re-sale price mentioned in our dataset and it is object in nature.
# Convert it into number(int,float)
price=[]
for i in dataset['Price']:
    price.append(i.replace("₹","").replace(",",""))

dataset['SalePrice']=price
dataset['SalePrice']=dataset['SalePrice'].astype(float)
```

So we have completed the process of data cleaning. Now will drop some unnecessary columns.

```
In [55]:
# Copying the dataset
model_dataset=dataset.copy()

# removing unnecessary columns
model_dataset.drop(columns=['Make_yr','Model',"Price","Registration","History"],inplace=True)
```

## 3. Checking for Outliers :

```
In [60]:
# Checking outliers
model_dataset.plot.box(figsize=(12,6))
plt.title("BoxPlot for Outliers",color='indianred',fontsize=16,pad=11)
plt.show()
```


Kms_driven and SalePrice variables contains outliers. Let's treat them using Zscore method. But before treating them convert categorical data into numerical data.

We are using Ordnal Encoding method to covert categorical data into numerical data –

```
In [61]:
from sklearn.preprocessing import OrdinalEncoder

# Using ordinal encoder to encoder categorical data into numerical
ol=OrdinalEncoder()
for i in cols:
    model_dataset[[i]]=ol.fit_transform(model_dataset[[i]])
```

```
In [62]:
# checking outliers using zscore method
from scipy.stats import zscore
z=np.abs(zscore(model_dataset))
z
```

```
Out[62]: array([[0.5682677 , 1.42411539, 0.73644487, ..., 2.16054217, 1.22487635,
                 0.0674221 ],
                [0.5682677 , 1.41100608, 0.73644487, ..., 1.82642836, 0.35973705,
                 0.47848498],
                [0.5682677 , 1.41083162, 0.73644487, ..., 0.82408693, 1.22487635,
                 0.50283136],
                ...,
                [0.5682677 , 0.07330777, 1.35787489, ..., 1.49231455, 1.94730109,
                 0.86357595],
                [0.5682677 , 0.13908305, 1.35787489, ..., 1.15820074, 1.94730109,
                 0.37689848],
                [0.5682677 , 5.16132098, 1.35787489, ..., 1.18059592, 1.94730109,
                 0.09383772]])
```

```
In [65]:
# Checking percentage loss of data
print("Percentage loss of data in treating outliers {}%".format(round((model_dataset.shape[0]-card
```

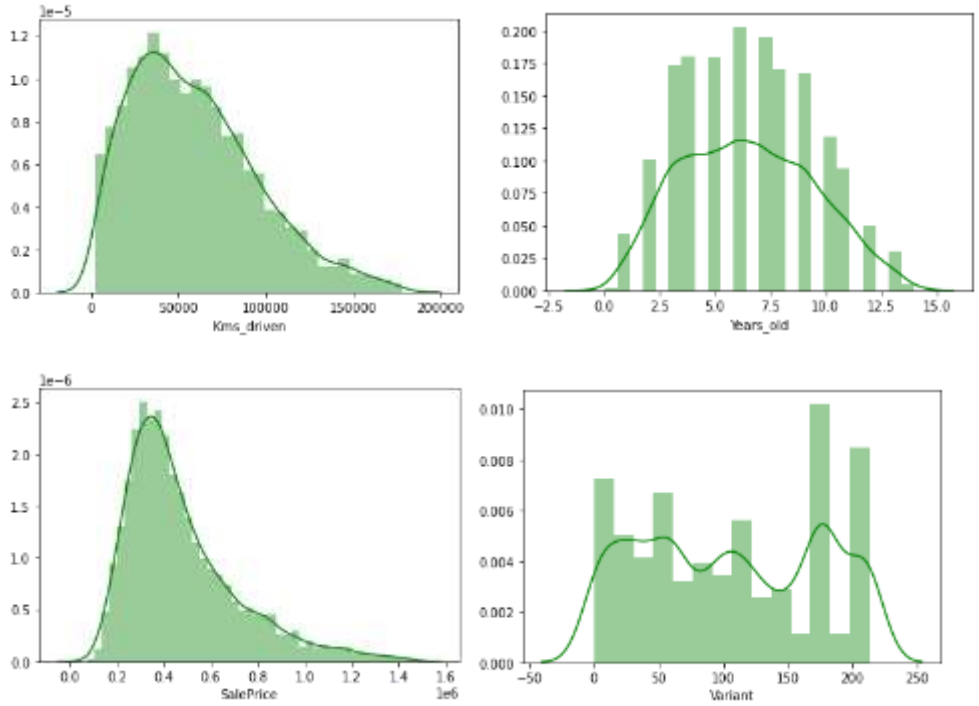
```
Percentage loss of data in treating outliers 3.38%
```

Removed 3.3% of outliers from dataset.

## 4. skewness in data –

```
In [66]:
# Checkinig skewness
cardata_new.skew()
```

```
Out[66]: Owner           1.189748
         Kms_driven      0.715876
         Fuel_type      -0.685095
         Transmission   -2.831751
         Insurance       0.040706
         Brand_name      0.146087
         model           0.002558
         Variant         0.059977
         Years_old       0.181510
         RTO             0.212382
         SalePrice       1.298426
         dtype: float64
```




Treated skewness using the PowerTransformer() technique-

```
In [68]:
# Removing skewness
from sklearn.preprocessing import PowerTransformer

pw = PowerTransformer()

cardata_new['Transmission']=pw.fit_transform(cardata_new[['Transmission']])
cardata_new['Owner']=pw.fit_transform(cardata_new[['Owner']])
cardata_new['Brand_name']=pw.fit_transform(cardata_new[['Brand_name']])
```

## 5. Scaling Data :

Now in the last step of data preprocessing we scale our dataset into equal range because sometimes our model can be biased towards higher values present in dataset to eliminate this risk we are using MinMaxScaler to scale our data.

```
In [71]:
# now rescaling our data so that model will not be biased
from sklearn.preprocessing import MinMaxScaler
mxn_scale=MinMaxScaler()

for i in cardata_new.columns:
    cardata_new[[i]]=mxn_scale.fit_transform(cardata_new[[i]])
```

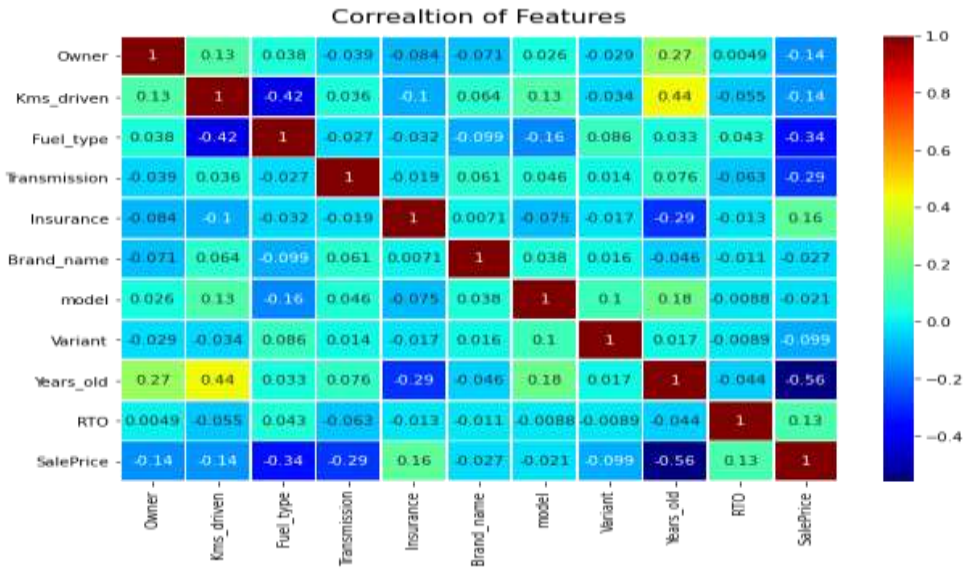
```
In [72]:
cardata_new.head(5)
```

Out[72]:

| | Owner | Kms_driven | Fuel_type | Transmission | Insurance | Brand_name | model | Variant | Years_old | RTO | SalePrice |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.000445 | 1.0 | 0.0 | 1.0 | 0.373834 | 0.541353 | 0.671362 | 0.000000 | 0.153846 | 0.298101 |
| 1 | 0.0 | 0.000432 | 1.0 | 1.0 | 0.5 | 0.666604 | 0.112782 | 0.262911 | 0.071429 | 0.384615 | 0.423053 |
| 2 | 0.0 | 0.000472 | 1.0 | 0.0 | 0.0 | 0.327958 | 0.458647 | 0.000390 | 0.285714 | 0.153846 | 0.429560 |

- ## Data Inputs- Logic- Output Relationships

```
In [69]:
# Plotting correlational plot
plt.figure(figsize=(10,7))
sns.heatmap(cardata_new.corr(), annot=True, fmt='.2g', linewidths=0.8,
    linecolor='white', cmap='jet')
plt.title("Correaltion of Features", fontsize=16, pad=11)
plt.show()
```




### Observations:

SalePrice having 50% negative correalation wiht Years_old feature which indicated that more the old less will be the SalePrice of car and vice-versa.

Variant is least correlated with SalePrice which indicated less fluctuations in SalePrice based on Variant of car.

Years_old and Kms_driven are sharing positive correlation of 40% which is quite realistic as more the years more will be the distance travelled.

- ## Hardware and Software Requirements and Tools Used

There is no such requirement for hardware ,but I have used intel i5 8th generation processor.

Software: Jupyter Notebook (Anaconda 3)

Language : Python 3.9

Libraries used in project:

a. Pandas
b. Numpy
c. Matplotlib
d. Seaborn
e. Sklearn
f. scipy
g. selenium
h. time

# Model/s Development and Evaluation

- Splitting Dataset

First of all we are splitting data into dependent and independent variable.

```
In [73]: 1 X=cardata_new.iloc[:,:-1]
         2 y=cardata_new['SalePrice']
         3 print("Splitted dependent & Independent data successfully.")

Splitted dependent & Independent data successfully.
```

Now will split data into training and testing data using train_test_split method –

```
[75]: 1 # now lets provide the best random state to the model
      2 x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=final_random_state)
      3
      4
      5 # Success
      6 print("Training and testing split was successful.")

Training and testing split was successful.
```

- Identification of possible problem-solving approaches (methods)

  As we already mentioned above we have continuous target variable so based on the type of variable we are using Regression approach for our model and will use some algorithms like

  a) Linear Regression
  b) Ridge Regression
  c) Support Vector Regressor
  d) DecisionTree Regressor
  e) KNeighbors Regressor
  f) RandomForest Regressor
  g) AdaBoost Regressor
  h) GradientBoosting Regressor

- Testing of Identified Approaches (Algorithms)

  After initiating the instances for our model we will create a function which will give us all the metrics score we are using in our model building by passing the model into it.



- Run and Evaluate selected models

**Evaluations of Algorithms**

```
In [86]:
# Table view of result of each metrix from above algorithms
evaluations = pd.DataFrame({"Model":Model,"R2 Score":R2Score,"MAE":MAE,"MSE":MSE,
                            "RMSE":RMSE,"CV Score":CVScore,"Stnd_dev":Std})
evaluations
```

Out[86]:

| | Model | R2 Score | MAE | MSE | RMSE | CV Score | Stnd_dev |
|---|---|---|---|---|---|---|---|
| 0 | LinearRegression() | 54.051 | 0.087556 | 0.013420 | 0.115846 | 46.353 | 0.057337 |
| 1 | Ridge() | 54.049 | 0.087512 | 0.013421 | 0.115849 | 46.367 | 0.057066 |
| 2 | DecisionTreeRegressor() | 80.807 | 0.048439 | 0.005606 | 0.074871 | 72.461 | 0.061658 |
| 3 | SVR() | 68.391 | 0.069861 | 0.009232 | 0.096083 | 61.930 | 0.040841 |
| 4 | RandomForestRegressor() | 90.080 | 0.035893 | 0.002903 | 0.053882 | 84.538 | 0.025321 |
| 5 | KNeighborsRegressor() | 66.042 | 0.066268 | 0.009918 | 0.099590 | 55.130 | 0.046838 |
| 6 | AdaBoostRegressor() | 50.929 | 0.098763 | 0.014332 | 0.119717 | 41.175 | 0.131246 |
| 7 | GradientBoostingRegressor() | 82.658 | 0.049714 | 0.005065 | 0.071169 | 78.761 | 0.015046 |

After sending the model into our created function, it created list for us and using these list we created a dataframe of all algorithms and metrics which we used for training.

Among these algorithms ,RandomForest Regressor and GradientBoosting Regressor are giving better result then others. So we will tune them using RandomizedSearchCV HyperParameter tunnig technique and find best result out of them.

- Hyperparameter tunning –

1). RandomForest Regressor:

```
In [48]:
rf_rdcv = RandomizedSearchCV(rf_reg, param_distributions=parameters,
                             scoring='r2', n_jobs=-1, verbose=2, cv=5)

rf_rdcv.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed:  4.2min finished
```

```
Out[48]:
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'bootstrap': [True],
                                        'criterion': ['mse', 'mae'],
                                        'max_depth': [80, 90, 100, None],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [8, 10, 12],
                                        'n_estimators': [100, 200, 300, 1000]},
                   scoring='r2', verbose=2)
```

```
In [49]:
print(rf_rdcv.best_params_)
print(rf_rdcv.best_score_)
```

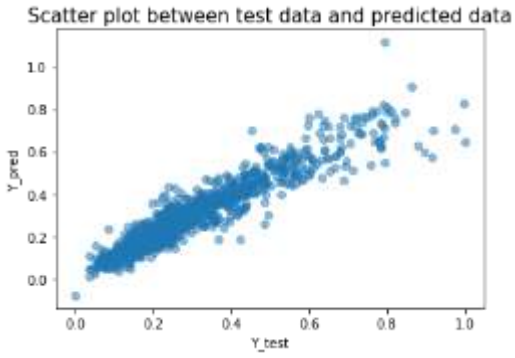
```
{'n_estimators': 1000, 'min_samples_split': 8, 'min_samples_leaf': 4, 'max_features': 'auto', 'max_depth': None, 'criterion':
'mse', 'bootstrap': True}
0.8174849155513318
```

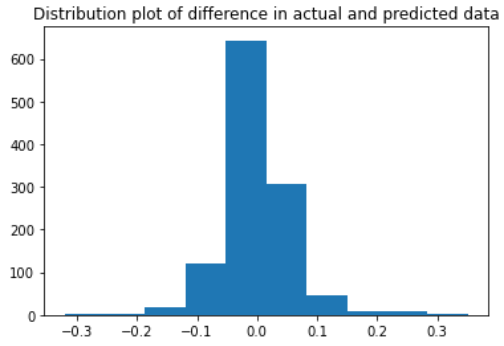
```
In [50]:
# Let's now try best parameter
rfr = RandomForestRegressor(n_estimators= 100, min_samples_split=10, min_samples_leaf= 1,
                            max_features= 'auto', max_depth= 90, criterion= 'mse',
                            bootstrap= True, random_state=final_random_state)

rfr.fit(x_train,y_train)
print("Training score = %.2f"%rfr.score(x_train,y_train))
rfr_pred = rfr.predict(x_test)
print("Coefficient of determination = %.2f "%r2_score(y_test,rfr_pred))
print("\nMean Absolute error = %.3f"%mean_absolute_error(y_test,rfr_pred))
print("Means Squared error = %.3f"%mean_squared_error(y_test,rfr_pred))
cv=cross_val_score(rfr, X,y, cv=5 ,scoring='r2')
print("\nCross vald. score = %.2f"%cv.mean())
print("Standard deviation = %.2f"%cv.std())
```

```
Training score = 0.94
Coefficient of determination = 0.89

Mean Absolute error = 0.038
Means Squared error = 0.003

Cross vald. score = 0.82
Standard deviation = 0.03
```

## 2). GradientBoosting Regressor:

```
In [94]: gbr_cv = RandomizedSearchCV(grd_reg, param_distributions=gradient_parameters,
                                     scoring='r2', n_jobs=-1, verbose=2, cv=5)

         gbr_cv.fit(x_train,y_train)

Fitting 5 folds for each of 10 candidates, totalling 50 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  35 out of  50 | elapsed:  2.0min remaining:   50.2s
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 13.5min finished

Out[94]: RandomizedSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=-1,
                            param_distributions={'criterion': ['mse', 'mae'],
                                                 'learning_rate': [1, 0.5, 0.2, 0.1,
                                                                   0.05, 0.02, 0.01,
                                                                   0.001],
                                                 'max_depth': range(1, 16, 2),
                                                 'max_features': ['auto', 'sqrt'],
                                                 'min_samples_split': range(200, 1001, 200),
                                                 'n_estimators': [10, 50, 70, 100, 200,
                                                                  1000]},
                            scoring='r2', verbose=2)

In [95]: print(gbr_cv.best_params_)
         print(gbr_cv.best_score_)

{'n_estimators': 1000, 'min_samples_split': 600, 'max_features': 'auto', 'max_depth': 7, 'learning_rate': 0.5, 'criterion': 'ms
e'}
0.8752076104604496

In [96]: # Let's now try best parameter
         gbr = GradientBoostingRegressor(n_estimators= 100, min_samples_split=1000, learning_rate=0.5,
                                         max_features= 'sqrt', max_depth= 11, criterion= 'mse')
         gbr.fit(x_train,y_train)
         print("Training score = %.2f"%gbr.score(x_train,y_train))
         gbr_pred = gbr.predict(x_test)
         print("Coef. of determination = %.2f "%r2_score(y_test,gbr_pred))
         print("\nMean Absolute error = %.3f"%mean_absolute_error(y_test,gbr_pred))
         print("Means Squared error = %.3f"%mean_squared_error(y_test,gbr_pred))
         cvscross_val_score(gbr, X,y, cv=5 ,scoring='r2')
         print("\nCross vald. score = %.2f"%cv.mean())
         print("Standard deviation = %.2f"%cv.std())

Training score = 0.93
Coef. of determination = 0.89

Mean Absolute error = 0.039
Means Squared error = 0.003

Cross vald. score = 0.84
Standard deviation = 0.03

In [97]: # Let's plot the graph of predicted and actual data
         plt.scatter(y_test,gbr_pred, alpha=0.5)
         plt.xlabel("Y_test")
         plt.ylabel("Y_pred")
         plt.title("Scatter plot between test data and predicted data",fontsize=13)
         plt.show()
```




After testing various algorithms and hypertunnig the best two algorithms we find that both RandomForestRegressor and GradientBoostingRegressor are giving better results but we are finalising the GradientBoostingRegressor algorithm for our model as there is minimum difference in coefficient of determination and cross validation score and also less variance in result in the higher SalePrice.

The selected model gives coefficient of determination value =0.89 with minimum mean absolute error of 0.039 and cross validation score 0.84 with a least standard deviation of 0.03.

- **Finalising and Saving the Model**


- **Key Metrics for success in solving problem under consideration**

In this model we have used -

R2score ,

mean absolute error,

mean squared error,

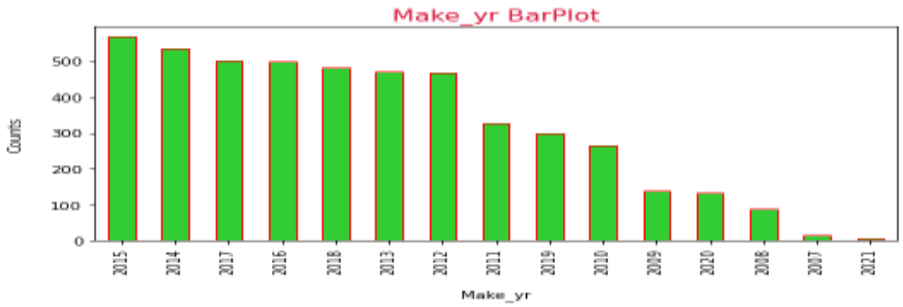
rootmean squared error ,

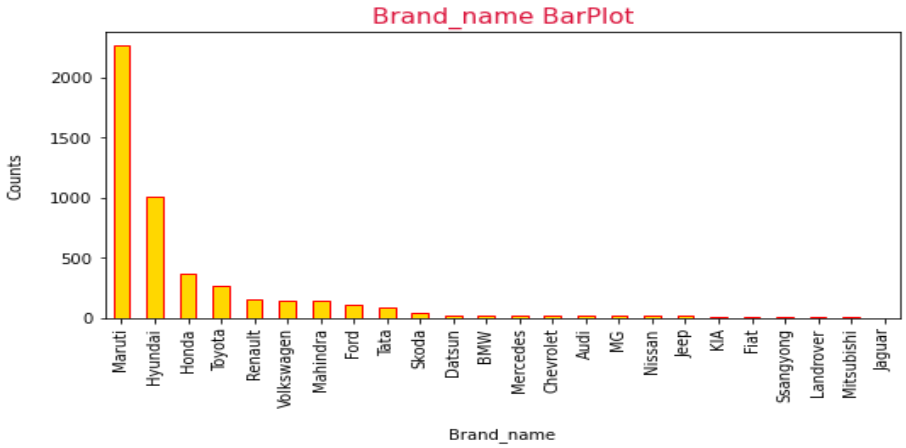
cross validation score ,

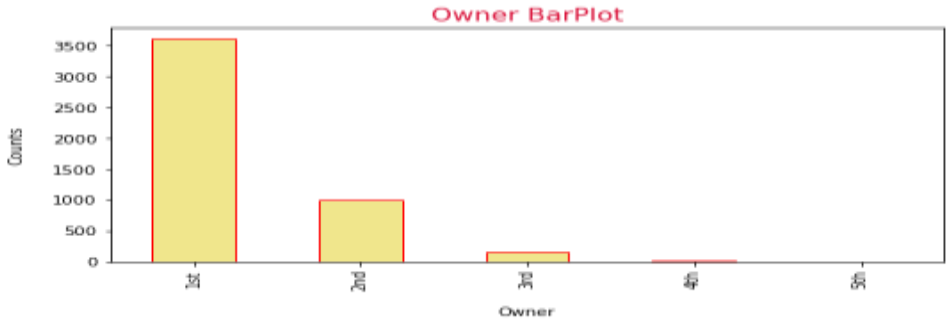
Standard deviation to find the best result out of any algorithm.

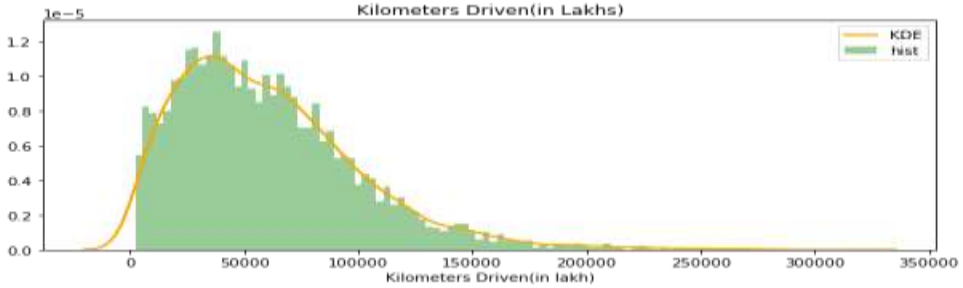
- **Visualizations**

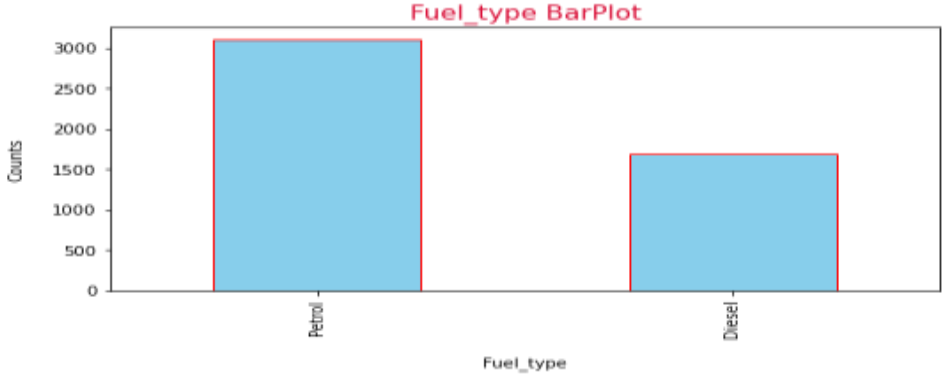
Checking data distribution in each variable –

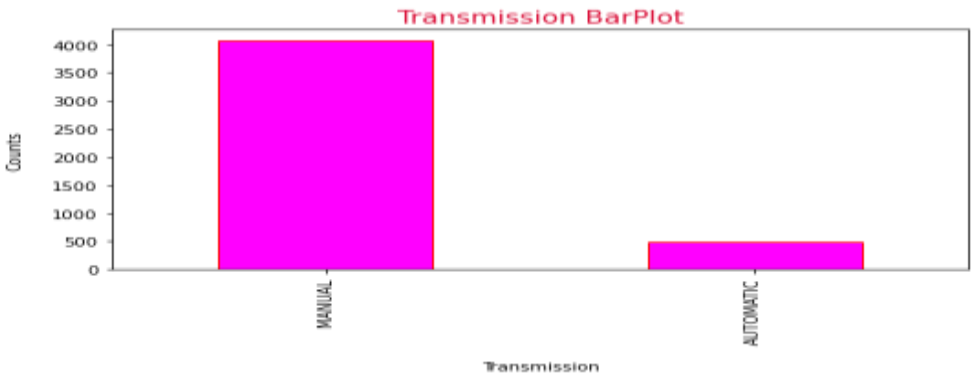

Make_yr BarPlot
Counts
0
100
200
300
400
500
2015
2014
2017
2016
2018
2013
2012
2011
2019
2010
2009
2020
2008
2007
2021
Make_yr

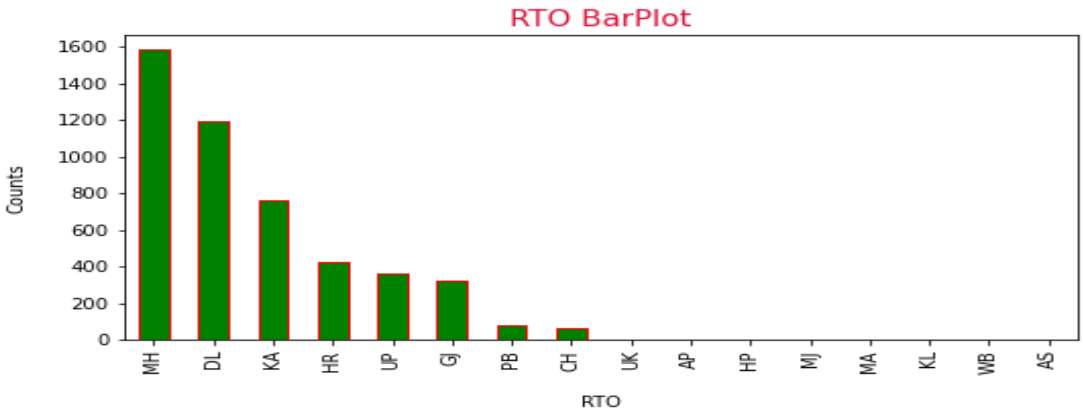

Brand_name BarPlot
Counts
0
500
1000
1500
2000
Maruti
Hyundai
Honda
Toyota
Renault
Volkswagen
Mahindra
Ford
Tata
Skoda
Datsun
BMW
Mercedes
Chevrolet
Audi
MG
Nissan
Jeep
KIA
Fiat
Ssangyong
Landrover
Mitsubishi
Jaguar
Brand_name

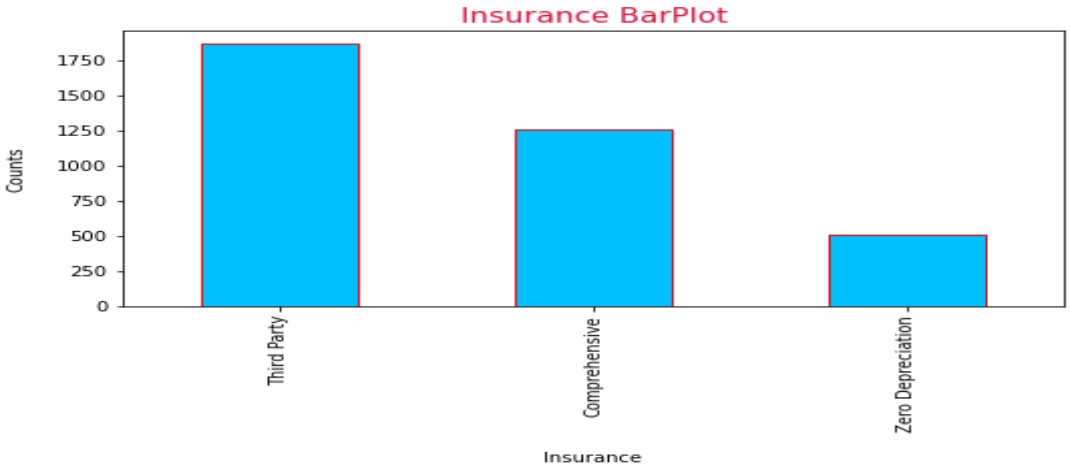

Owner BarPlot
Counts
0
500
1000
1500
2000
2500
3000
3500
1st
2nd
3rd
4th
5th
Owner


Kilometers Driven(in Lakhs)
1e−5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
KDE
hist
0
50000
100000
150000
200000
250000
300000
350000
Kilometers Driven(in lakh)

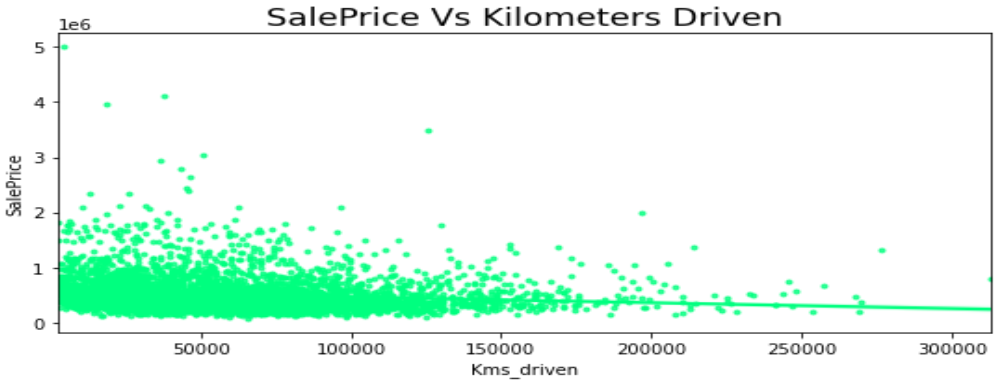

Fuel_type BarPlot
Counts
0
500
1000
1500
2000
2500
3000
Petrol
Diesel
Fuel_type

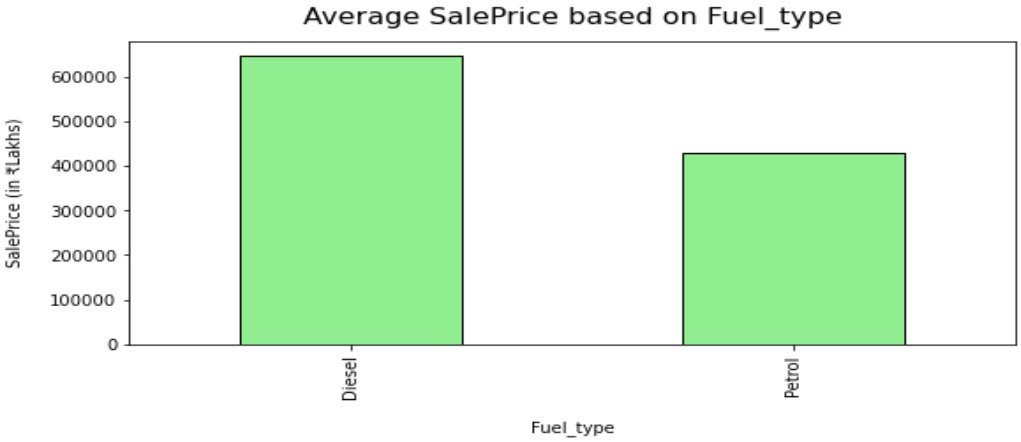


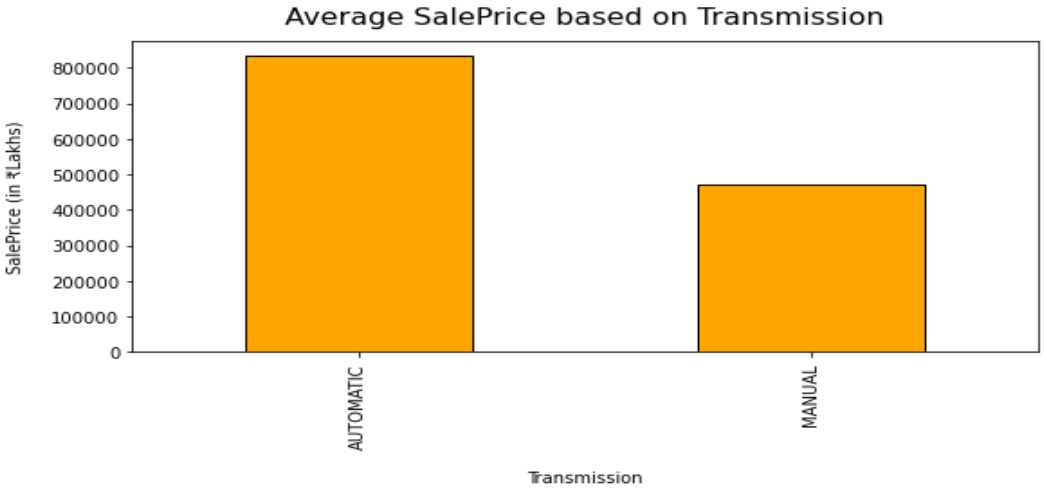



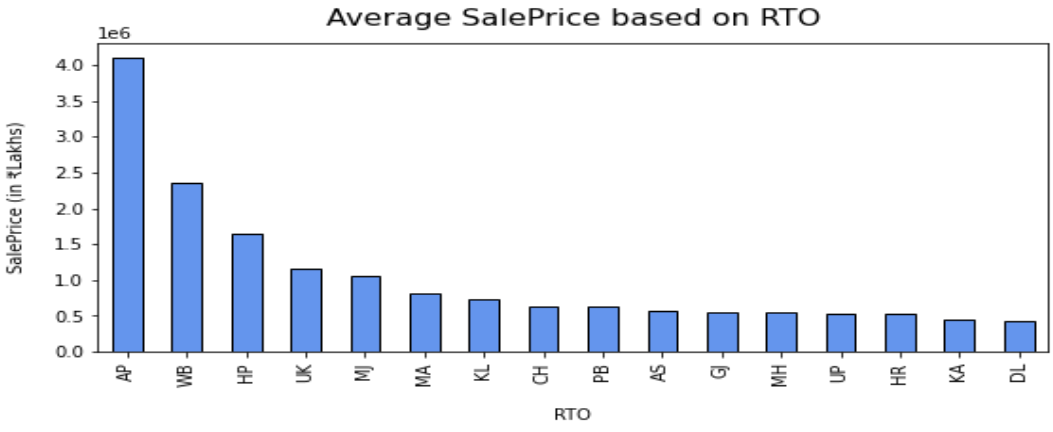



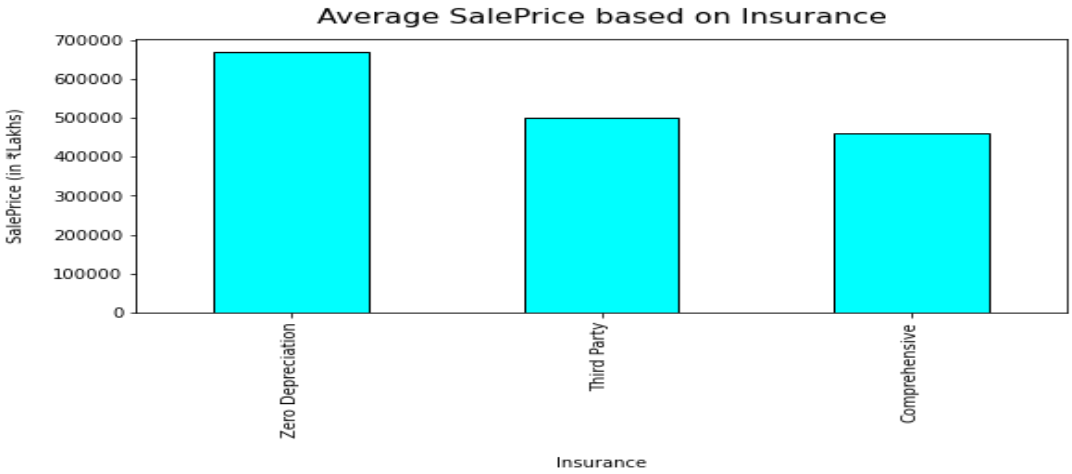

Now checking Average SalePrice based on each variable –

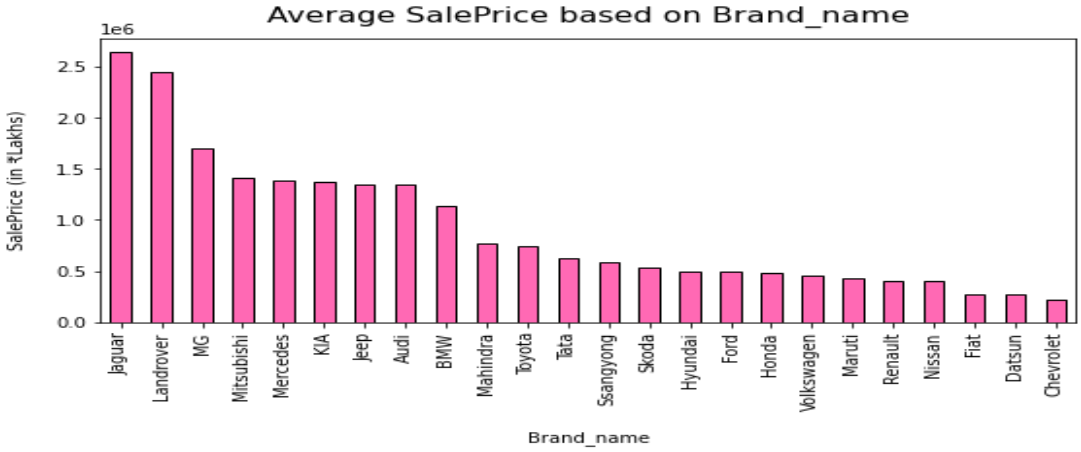

SalePrice Vs Kilometers Driven
1e6
SalePrice
Kms_driven
0
1
2
3
4
5
50000
100000
150000
200000
250000
300000

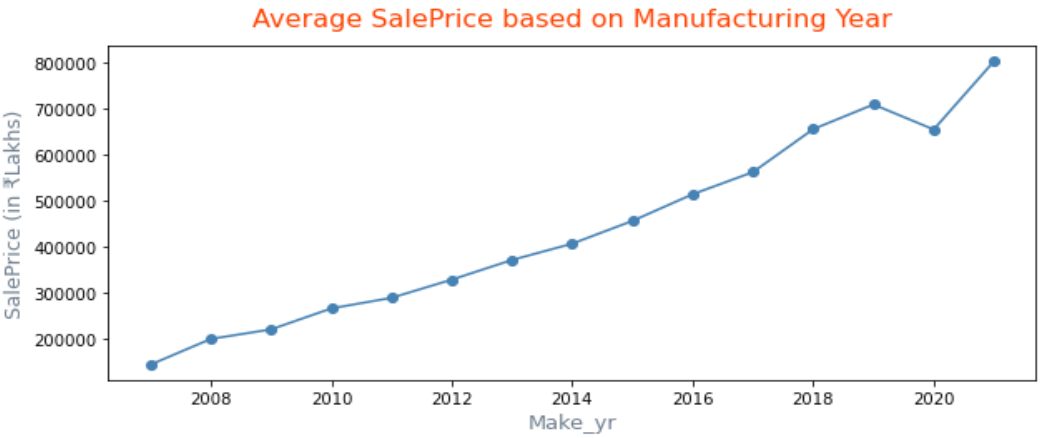

Average SalePrice based on Fuel_type
SalePrice (in *Lakhs)
Fuel_type
Diesel
Petrol
0
100000
200000
300000
400000
500000
600000

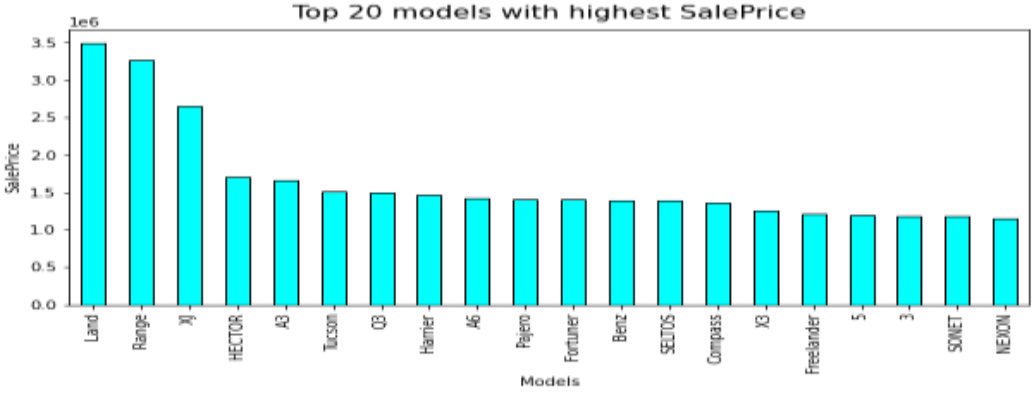

Average SalePrice based on Transmission
SalePrice (in *Lakhs)
Transmission
AUTOMATIC
MANUAL
0
100000
200000
300000
400000
500000
600000
700000
800000

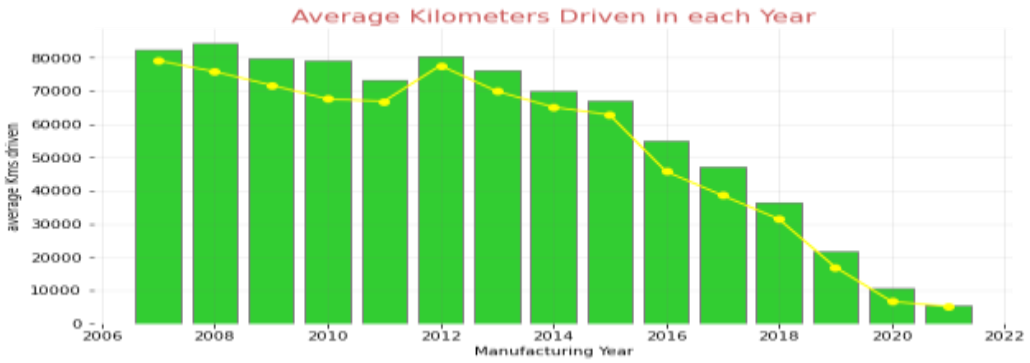

Average SalePrice based on RTO
1e6
SalePrice (in *Lakhs)
RTO
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
AP
WB
HP
UK
MJ
MA
KL
CH
PB
AS
GJ
MH
UP
HR
KA
DL

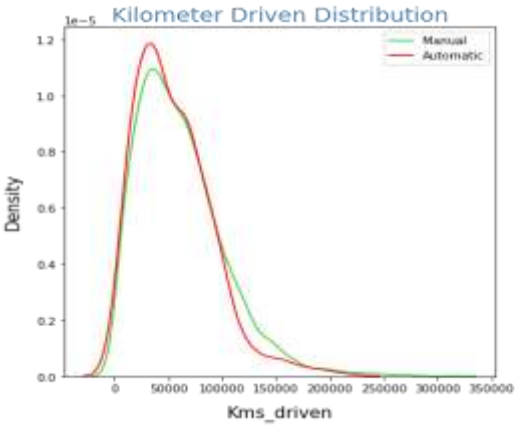

Average SalePrice based on Insurance
SalePrice (in ₹Lakhs)
700000
600000
500000
400000
300000
200000
100000
0
Zero Depreciation
Third Party
Comprehensive
Insurance

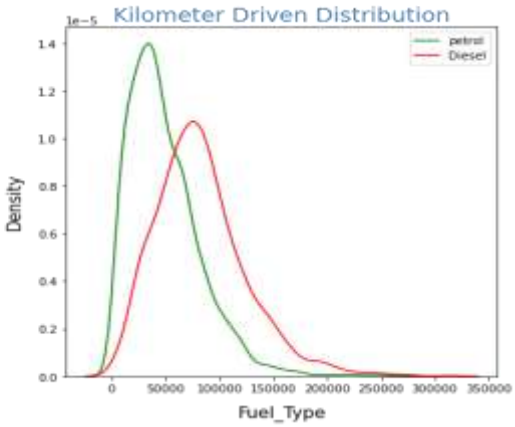

Average SalePrice based on Brand_name
1e6
SalePrice (in ₹Lakhs)
2.5
2.0
1.5
1.0
0.5
0.0
Jaguar
Landrover
MG
Mitsubishi
Mercedes
KIA
Jeep
Audi
BMW
Mahindra
Toyota
Tata
Ssangyong
Skoda
Hyundai
Ford
Honda
Volkswagen
Maruti
Renault
Nissan
Fiat
Datsun
Chevrolet
Brand_name

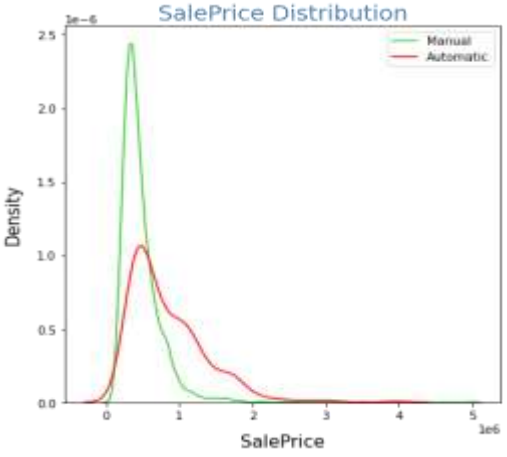

Average SalePrice based on Manufacturing Year
SalePrice (in ₹Lakhs)
800000
700000
600000
500000
400000
300000
200000
2008
2010
2012
2014
2016
2018
2020
Make_yr

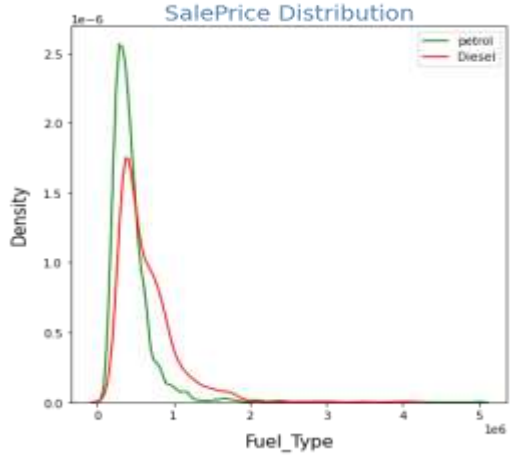

Top 20 models with highest SalePrice
1e6
SalePrice
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Land
Range
XJ
HECTOR
A3
Tucson
Q3
Harrier
A6
Pajero
Fortuner
Benz
SELTOS
Compass
X3
Freelander
5
3
SONET
NEXON
Models

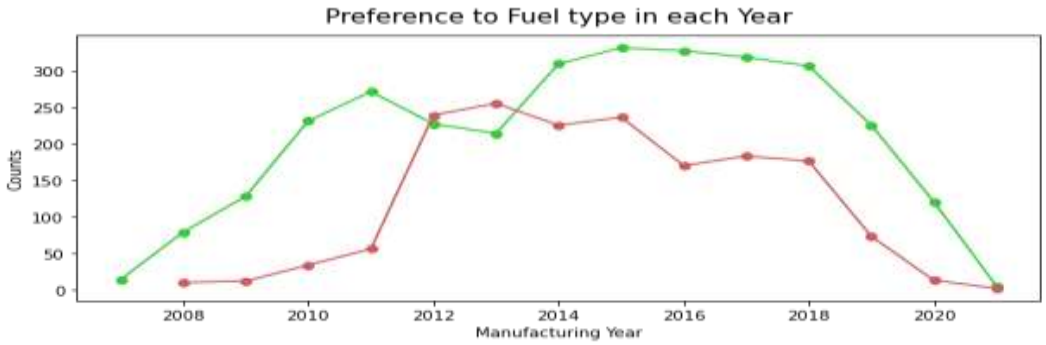

Average Kilometers Driven in each Year
average Kms driven
Manufacturing Year


Kilometer Driven Distribution
Manual
Automatic
Density
Kms_driven


Kilometer Driven Distribution
petrol
Diesel
Density
Fuel_Type


SalePrice Distribution
Manual
Automatic
Density
SalePrice


SalePrice Distribution
petrol
Diesel
Density
Fuel_Type


Preference to Fuel type in each Year
Counts
Manufacturing Year

- ## Interpretation of the Results

1. Cars manufactured in between 2012-2018 are more in number for sale. While a few cars are for resale before 2010.
2. Maruti and Hundai are the most prefered brands for resale in indian market. These brands are sharing more than 50% of resale market share.
3. More likely the cars which are 3-9 years old are for resale.
4. History feature represents there is only "Non-Accidental" values. This feature not providing any valuable information, we can drop it.
5. In car resale market most of the cars for sale are first owners.
6. Minimum kilometers covered by a car is 2`259km`s while maximum kilometers covered is `312882km`. There is huge difference in minimum and maximum kilometers.
7. The distribution plot of Kilometers driven tells that the data gathered is not uniform. Some cars have travelled less while some very high. The difference between the mean and median shows the `presence of outliers` and the `rightskewness` is present in data as `mean is greater than median`.
8. Manual transmission is most prefered.
9. Resale of car in used car market depends on the number of owners present for the car as from the above graph it clearly shows the average saleprice of 1st and 5th owners have a lot of difference.
10. Based on the scatter plot we find that there is slightly negative relation of SalePrice upon Kilometers driven.
11. Another feature which impact the SalePrice is Fuel type. Cars in which Diesel is used are having high average saleprice then Petrol as Deisel cars are more costly than Petrol cars.
12. Based on the transmission ,we find out that cars with automatic transmission are having high SalePrice compared to Manual transmission. Again Automatic transmission are costly than Manual type of transmission.

13. Based on the RTO locations above graph showing that `AndraPradesh, West Bengal, HimachalPradesh` are among the most valued for high SalePrice while Delhi is in least valued RTO locations.
14. In Comprehensive ,any damage to vehicle, owner or third person all liabilities are recoverd through insurance company. There is no need to bear all the charges own but in third-party insurance as clear with the name only covers for third party liabilities while in Zero Depreciation we have some premium add on.
15. So basically according to the need of vehicle owners Zero Depreciation is more valuable then others and having High salePrice then other types of insurance.
16. Based on the Brands we find out Average SalePrice of some premium brand are very high than other brands present in Indian market like `Jaguar, Landrover` and `Fiat, Datsun, Chevrlet` are the brands with least SalePrice.
17. In covid crisis, whole economy suffered a lot & due to this demand supply for some goods decreases. This decrease in demands sets the saleprice at low indexes that's why we are creating a model for our client and this behaviour in SalePrice can be seen in above graph as the constant increase in SalePrice each year suddenly falls in 2020 but in 2021 is again stars to rise.
18. After modelling we find out that GradientBoostingRegressor algorithm is giving best result than other. So we finalised this algorithm .
19. The selected model gives coefficient of determination value =0.89 with minimum mean absolute error of 0.039 and cross validation score 0.84 with a least standard deviation of 0.03.

# CONCLUSION

- <u>Key Findings and Conclusions of the Study</u>

After working on this project ,we got a lot of insights of how to collect data , preprocess it ,remove unwanted data and how to tackle null values to create a good model.

In this project we find various factors responsible for the sale price of a car. Let's take each variable to describe our observations from this project –

i. Brand Name –

After analysing the brand name variable we find that in Indian car market brands like "Maruti , Hyundai, Honda , Toyota" are covering most of the market while moving towards the Saleprice value we find that the premium brands like "Jaguar, LandRover , Mitsubishi, Mercedes , jeep , Audi , BMW" are among the most valued brands in Re-sale Car market.

ii. Make_yr Variable –

Analysing the Make_yr variable we find that the cars who have been manufactured in 2012-2018 are more in number for sale while athe average graph of SalePrice denotes that there is constant rise in SalePrice each year upto 2020, in 2020 due to covid crisis it comes down but after that again start to rise.

Model Variable –

"Rover , A3 , A6 ,Q3 ,Tuscon, Pajero , Benz' are among the top model whose average sale price is high.

iii. Owner Variable –

Owner variable shows the number of owners while selling the car. So most of the data contains least owners which is 1 and a few data was present of more than 2 owners. While checking the effect of Owners on SalePrice we find that more the owners less will be the sale price.

iv. Kms_driven Variable –

In this variable we have a right skewness as some of the cars had travelled a lot while some very less. As some people travels more through public transport while some use their own vehicle to travel this cause variance in data. There is very high difference in minimum and maximum kilometers driven in datatset.

v. Fuel_type Variable –

Nowdays there are more than 2 types of fuel used in cars they may be petrol , diesel, electric or hybrid but the data we collected contains only petrol and diesel and based on this we do analysis and found out that people owning petrol cars are more in number but diesel cars are more costilier than petrol cars.

vi. Transmission Variable –

In transmission we have 3 options – Manual, Automatic, Hybrid.But the data we collected contains only Manual and Automatic type of transmission. As Automatic transmission is newly introduced in Indian Market a few years back so there is not much data based on this type of transmission. While finding the saleprice we noticed that Automatic transmission has higher Average SalePrice as compared to Manual.

vii. RTO Variable –

RTO variable define the State in which Road tax Office is present. And we find that Delhi , UttarPardesh are among the States with minimum Avg SalePrice while AndraPradesh and WestBengal are having maximum Avg. salePrice.

viii. Insurance Variable –

Insurance variable contains a lot of dirty data. First we cleaned it using the replace method and then it also have a lot of null values which were treated using the most frequent value.

After this we tried to find that most of the cars having Third Party insurance and the cars with Zero Depriciation are among the most

valued during the sale as in Zero depreication , all liabilities are covered by insurance company.

- Limitations of this work and Scope for Future Work

  In this project we have to deal with a lot of problems :

  a) Websites are not well maintained as we get almost 10% of duplicated data.
  b) The data present on contains a lot of missing features which create hinderance in analysing the data properly.
  c) We have to face a lot of outliers in SaelPrice and transmission features.

Thankyou