FLIP ROBO

CAR PRIZE PREDICTION MODEL

Submitted by:

Bhushan Kumar Sharma,

Intern Data Scientist

ACKNOWLEDGMENT

*****

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with this dataset and it has helped me to improve my model building skills.

A huge thanks to my academic team “Datatrained” who is the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank to many other persons who has helped me directly or indirectly to complete the project.

Following are the external references which I used:

www.geeksforgeeks.org

www.stackoverflow.com

www.w3school.com

www.google.com

Datatrained Lectures

. 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. This project contains two phase.

➢ Conceptual Background of the Domain Problem

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. Our results show that GradientBoostingRegressor is giving best results.

➢ Review of Literature

- Dataset is extracted by using web scraping techniques for different-different cities from different –different used cars websites.
- Prize will get decided based on below parameters-driven, model year, model and company etc.
- After data scrapping, data is used to build Machine

learning model, as we have to predict prize of used car, which clearly indicating this is a regression problem.

- Various pre-processing techniques are used in this project to give best data to our machine learning model, and at final its giving good accuracy.

➢ Motivation for the Problem Undertaken

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 [2-3]. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

➢ Data Sources and their formats

The data was collected by using web scrapping on used cars websites. In web scrapping I have used selenium. It is hereby used for this model building. In order to predict the prize of customers for the sell, the client wants some predictions that could help them in further investment and improvement in selection of customers.

Also, my dataset was having 6254 rows and 9 columns including target. In this particular datasets I have object, float and integer types of columns. The information about features is as follows.

```
df = pd.read_csv("D:/Bhushan Sharma/Internship/Car Price Prediction model/used_car_scarpped_data.csv")
df.head()
```

	Title	History	Kilometer	Location	Year	Onwer	fuel	transmission	prize
0	2017 Tata TIGOR Revotron XT MANUAL	Non-Accidental	70,895 km	DL	17-May	1st Owner	Petrol	MANUAL	₹4,56,399
1	2013 Ford Figo 1.4 ZXI DURATORQ MANUAL	Non-Accidental	75,454 km	DL	13-Oct	1st Owner	Diesel	MANUAL	₹2,01,799
2	2012 Maruti Swift VDI MANUAL	Non-Accidental	1,20,326 km	DL	12-Nov	2nd Owner	Diesel	MANUAL	₹3,06,599
3	2012 Volvo S60 SUMMUM D5 AUTOMATIC	Non-Accidental	41,361 km	DL	12-Dec	2nd Owner	Diesel	AUTOMATIC	₹9,40,799
4	2012 Maruti Swift Dzire VDI BS IV MANUAL	Non-Accidental	77,730 km	DL	12-Nov	1st Owner	Diesel	MANUAL	₹3,62,399

```
pd.set_option('display.max_rows', None)  # to maximize display of the rows
pd.set_option('display.max_columns', None)  # to maximize display of the columns
```

# Title	:	Car Title
# History	:	Car History (Accidental or not)
# Kilometer	:	Car Driven km
# Location	:	Car registered location
# Year	:	Car purchased year
# Onwer	:	Car owner (1st or 2nd etc.)
# fuel	:	Car fuel type
# transmission	:	Car transmission (Automatic or Manual)
# prize	:	Car (car prize)

Data Information

```
df.info()

 <class 'pandas.core.frame.DataFrame'>
 RangeIndex: 6254 entries, 0 to 6253
 Data columns (total 9 columns):
  #   Column        Non-Null Count  Dtype
 ---  ------        --------------  -----
  0   Title         6254 non-null   object
  1   History       6254 non-null   object
  2   Kilometer     6254 non-null   object
  3   Location      6254 non-null   object
  4   Year          6254 non-null   object
  5   Onwer         6254 non-null   object
  6   fuel          6254 non-null   object
  7   transmission  6015 non-null   object
  8   prize         6254 non-null   object
 dtypes: object(9)
 memory usage: 439.9+ KB
```

Data Dtype:

```
df.dtypes

Title           object
History         object
Kilometer       object
Location        object
Year            object
Onwer           object
fuel            object
transmission    object
prize           object
dtype: object
```

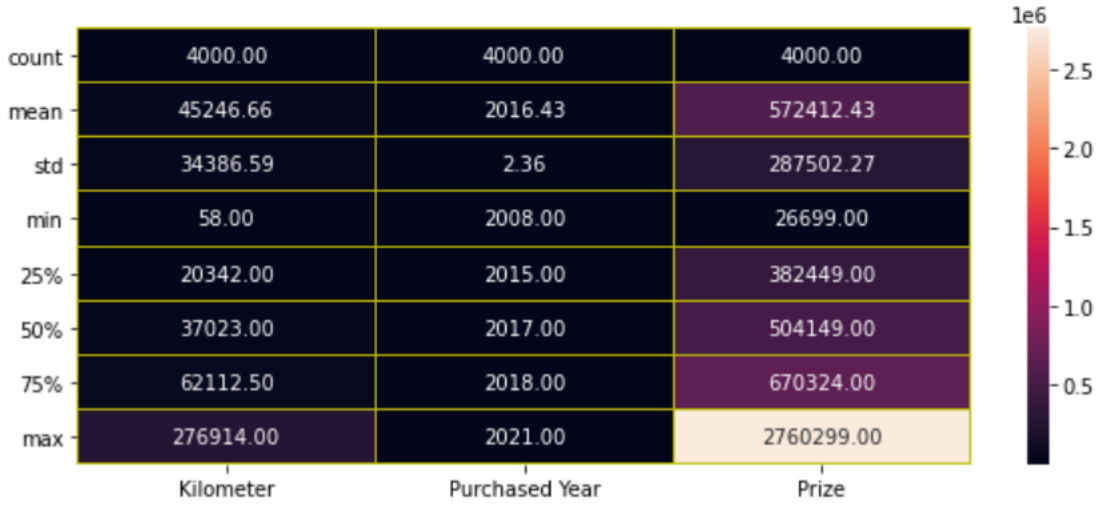
Initially every column is of object type, and transmission column is having null value but that is hide by "-", once I will replace "-" with null value then we would find some missing data.

Total 9 column we have received by data, in which 8 are feature columns and last one is target column (prize).

As we to have find the used car prize, and target variable is in numeric form so it is any regression problem.

Analytical Problem Framing

- **Describe dataset:**



i) As every column is showing high difference between mean and 50% percentile, which indicating the skewness of the column

ii) Very high difference found between min and max value of kilometre and prize columns.

iii) Kilometer and Prize columns min and max value are having high difference, we can say these column are highly spread.

iv) Mean and 50 percentile value is having some value difference, mean these columns are having skewness

v) Count for each column is same mean no null value present.

vi) 25% and 75% is having difference , which also indicating , column spread.

The statistical figure (above mentioned) get to know by the df.describe() so many information, like min, max, standard deviation, 25 percentile, 50th percentile, 75 percentile.

Then by the help of correlation function I get to know the correlation of each columns with each other.

From the heat map one can visualized to see them clearly that they are positive correlated or the negative correlated, the dark side shows the negative correlation among each other and lighter side represents the positive correlation among the each other.

- **Correlation :**

i) Darker cell is having high value of correlation where light colour cell is having less value of correlation.

ii) Numeric columns are not much correlated with each other, as they are showing less value of correlation

iii) As these column are not showing much correlation, it would be helpful to build a good model of machine learning

➢ Data Sources and their formats:

➢ Data Pre-processing:

i) I have checked duplicates rows in the dataset:
Removed them by using drop_duplicates method:

Removing Duplicates

```
: df.drop_duplicates(inplace = True)
  df.shape

: (4152, 9)
```

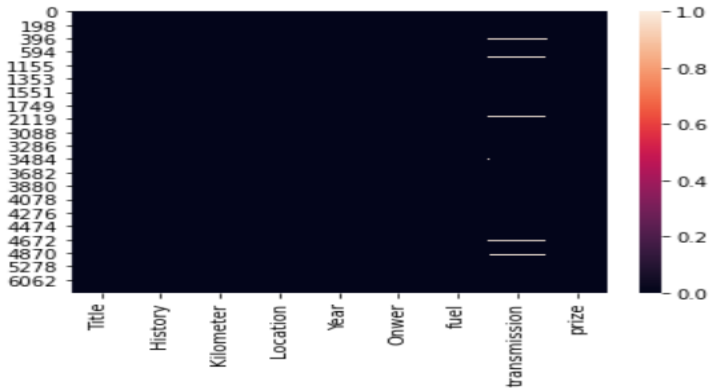
ii) Firstly, I have replace "-" with Null value then found some null values in the Transmission column, by removing null values rows, we are getting very low data loss (less than 4 % of data). Therefore, removed all null values by apply dropna function on dataframe.

```
df.replace('-', '', inplace = True)
df.isnull().sum()
```

```
Title            0
History          0
Kilometer        0
Location         0
Year             0
Onwer            0
fuel             0
transmission   152
prize            0
dtype: int64
```

```
sns.heatmap(df.isnull() )
```

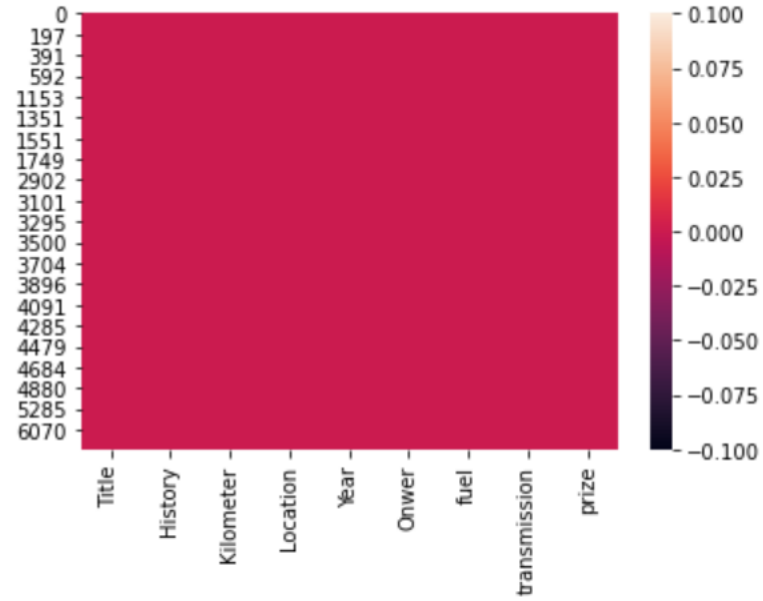
```
<AxesSubplot:>
```




```
df.dropna(inplace = True)
```

```
sns.heatmap(df.isnull()  )
```

```
<AxesSubplot:>
```



Note: **As heatmap is now clear, which indicating that, no null values are present in the dataset.**

iii) Extraction of Car Purchased **Month** and **Year** from the Year column:

```
df['Purchased Month'] = df['Year'].apply(lambda x:x.split("-")[0])
```

```
df['Purchased Year'] = df['Year'].apply(lambda x:x.split("-")[1])
```

```
df['Purchased Month'].unique()
```

```
array(['May', 'Oct', 'Nov', 'Dec', 'Sep', 'Jun', 'Apr', 'Aug', 'Feb',
       'Jan', 'Mar', 'Jul'], dtype=object)
```

```
df['Purchased Year'].unique()
```

```
array(['17', '13', '12', '16', '10', '14', '15', '18', '19', '20', '21',
       '11', '09', '08'], dtype=object)
```

```
n = {'10': 2010, '11': 2011, '12':2012, '13':2013, '14': 2014, '15':2015, '16':2016, '17':2017, '18':2018, '19':2019,'20':2020,
     '21':2021, '08':2008, '09':2009 }
df['Purchased Year']= df['Purchased Year'].replace(n)
```

```
df['Purchased Year'].unique()
```

```
array([2017, 2013, 2012, 2016, 2010, 2014, 2015, 2018, 2019, 2020, 2021,
       2011, 2009, 2008], dtype=int64)
```

Many other pre-processing steps which are applied to our dataset:

```
df['Title'][0][5:]
```

```
'Tata TIGOR Revotron XT MANUAL'
```

```
df['Title'] = df['Title'].apply(lambda x:x[5:])
# df.head()
```

```
df['Purchased Year'] = df['Purchased Year'].astype('int')
```

```
df['prize'][0].replace(',', '')
```

```
'₹456399'
```

```
df['prize'] = df['prize'].apply(lambda x:x.replace(',', ''))
```

```
df['prize'] = df['prize'].apply(lambda x:x[1:])
```

```
# df['prize'] = df['prize'].apply(lambda x: x[0][1:])
df['prize'] = df['prize'].astype('int')
```

```
df['History'].unique() # All rows having non accidental status therefore this column can be deleted from the dataset
```

```
array(['Non-Accidental'], dtype=object)
```

Operation of Kilometre column:

```
df['Location'].unique()
```

```
array(['DL', 'HR', 'UP', 'CH', 'PB', 'GJ', 'MH', 'KA', 'TS', 'AP', 'TN',
       'WB', 'OR', 'JH', 'KL', 'BR', 'RJ'], dtype=object)
```

```
df['Kilometer'] = df['Kilometer'].apply(lambda x:x[:-3])
```

```
df['Kilometer'] = df['Kilometer'].apply(lambda x:x.replace(',', ''))
```

```
df.drop(columns = ['Year', 'History'], inplace = True)
```

```
df['Title'] = df['Title'].apply(lambda x:x.replace('MANUAL',''))
df['Title'] = df['Title'].apply(lambda x:x.replace('AUTOMATIC',''))
```

```
df['Company'] = df['Title'].apply(lambda x:x.split(" ")[0])
```

```
df['Car Name'] = df['Title'].apply(lambda x:x.split(" ")[1])
```

```
title_value = df['Title'].values
```

```
models_lst = []
for i in title_value:
    var = i.split(' ')
    models_lst.append(i.replace(var[0], '').replace(var[1], '')[2:])

len(models_lst)
df['Model'] = models_lst
```

```
df['Model'] = df['Model'].apply(lambda x:x.replace('DIESEL', ''))
df['Model'] = df['Model'].apply(lambda x:x.replace('PETROL', ''))
```

```
df.drop(columns = ['Title'], inplace = True)
```

```
df['Prize']=df['prize']
```

```
df.drop(columns = ['prize'], inplace = True)
```

```
df['Car Name'].unique()
```

```
array(['TIGOR', 'Figo', 'Swift', 'S60', 'Innova', 'New', 'Ertiga', 'S',
       'Ritz', 'Wagon', 'Dzire', 'Etios', 'Q3', 'Ciaz', 'Ecosport',
       'Fortuner', 'Amaze', 'Duster', 'Tiago', 'Vitara', 'Kwid', 'Baleno',
       'X1', 'Alto', '3', 'City', 'Verna', 'Tucson', 'SELTOS', 'Elite',
       'VENUE', 'Creta', 'IGNIS', 'WR-V', 'NEW', 'YARIS', 'Glanza',
       'Corolla', 'i20', 'Benz', 'Vento', 'i10', 'XUV500', '5', 'Grand',
       'Brio', 'Eon', 'Polo', 'Celerio', 'Jazz', 'GRAND', 'A', 'Ameo',
       'Redi', 'Xcent', 'Compass', 'Eeco', 'Superb', 'Civic', 'HECTOR',
       'Bolero', 'Thar', 'NEXON', 'Micra', 'Zen', 'TUV300', 'Rapid',
       'CRV', 'Hexa', 'Go', 'BR-V', 'Octavia', 'Terrano', 'ALTROZ',
       'Nano', 'Scorpio', 'Freelander', 'XF', 'Camry', 'Kuv100',
       'FREESTYLE', 'OMNI', 'Harrier', 'TRIBER', 'Jetta', 'XL6', 'Santa',
       'A3', 'Santro', 'Mobilio', 'Captur', 'MARAZZO', 'AURA', 'Sunny',
       'Rexton', 'A4', 'Endeavour', 'Beat'], dtype=object)
```

```
df['Kilometer']= df['Kilometer'].astype('float')
df['Prize']= df['Prize'].astype('float')
```

```
df.dtypes
```

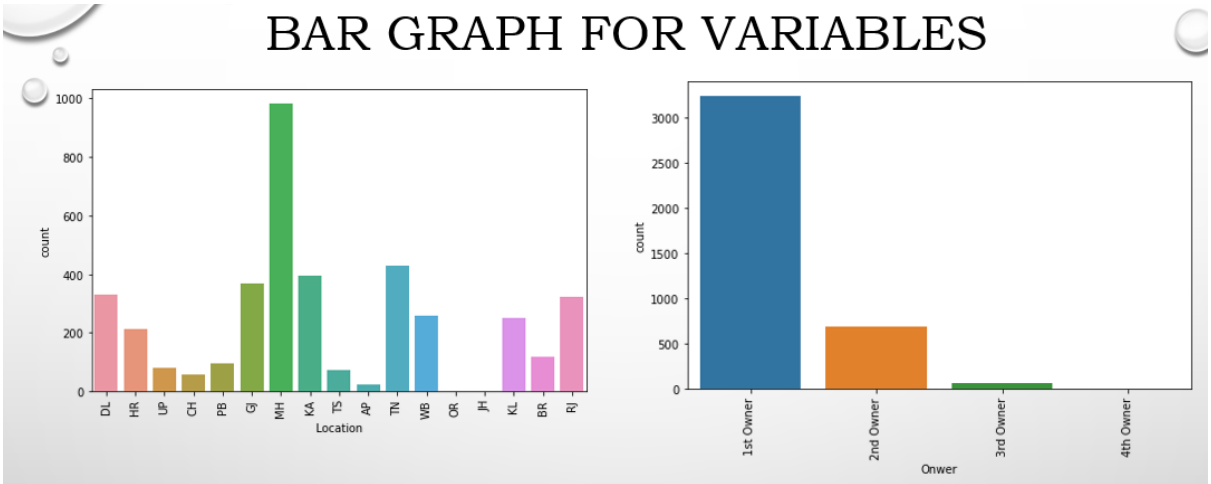
```
Kilometer          float64
Location            object
Onwer               object
fuel                object
transmission        object
Purchased Month     object
Purchased Year       int32
Company             object
Car Name            object
Model               object
Prize              float64
dtype: object
```

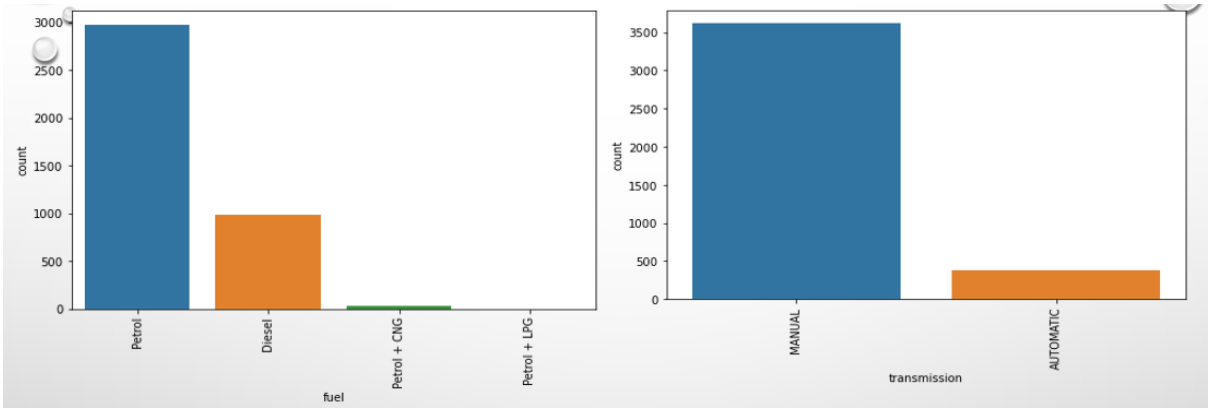
After applying various pre-processing techniques, we are having 11 columns, 10 are feature and one is target column

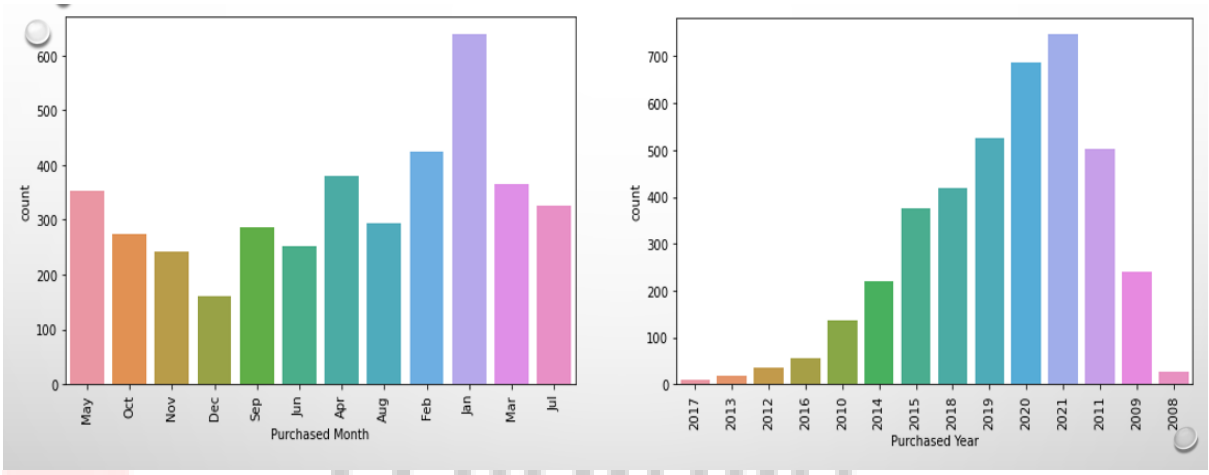
Here, in the dataset 2 columns are of float type and other is of object and int type columns

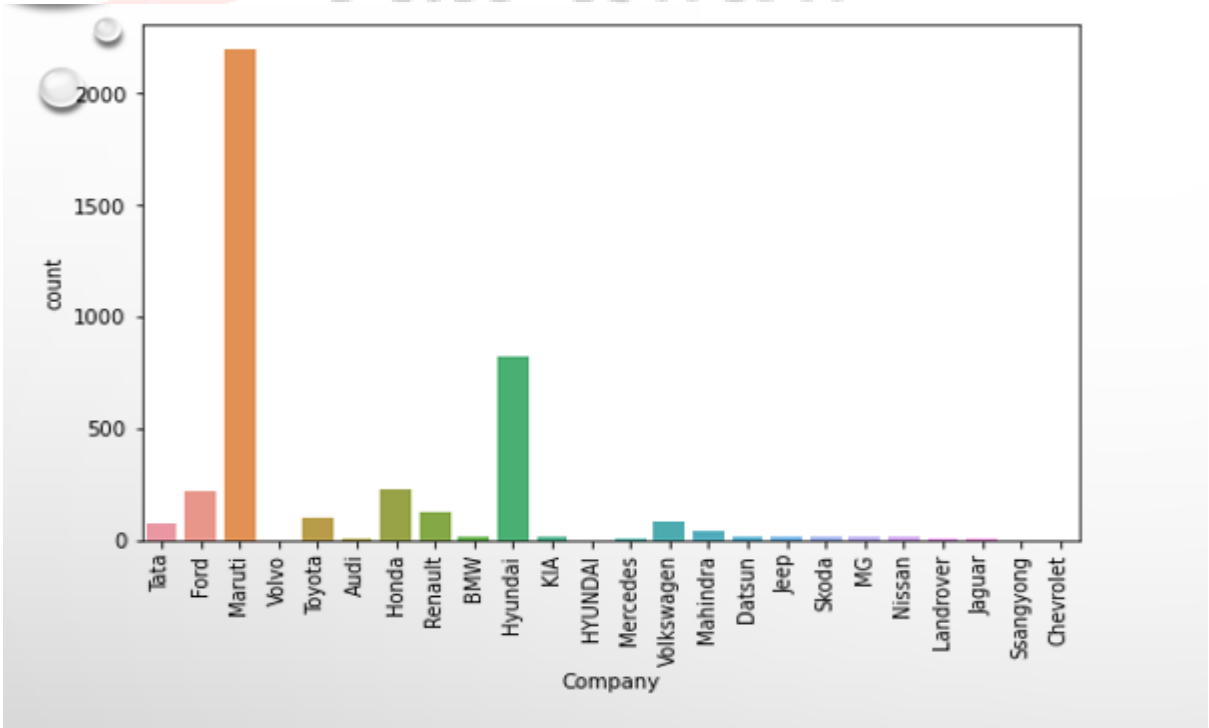
iv) After classifying these column, I have deleted un necessary columns of dataset:

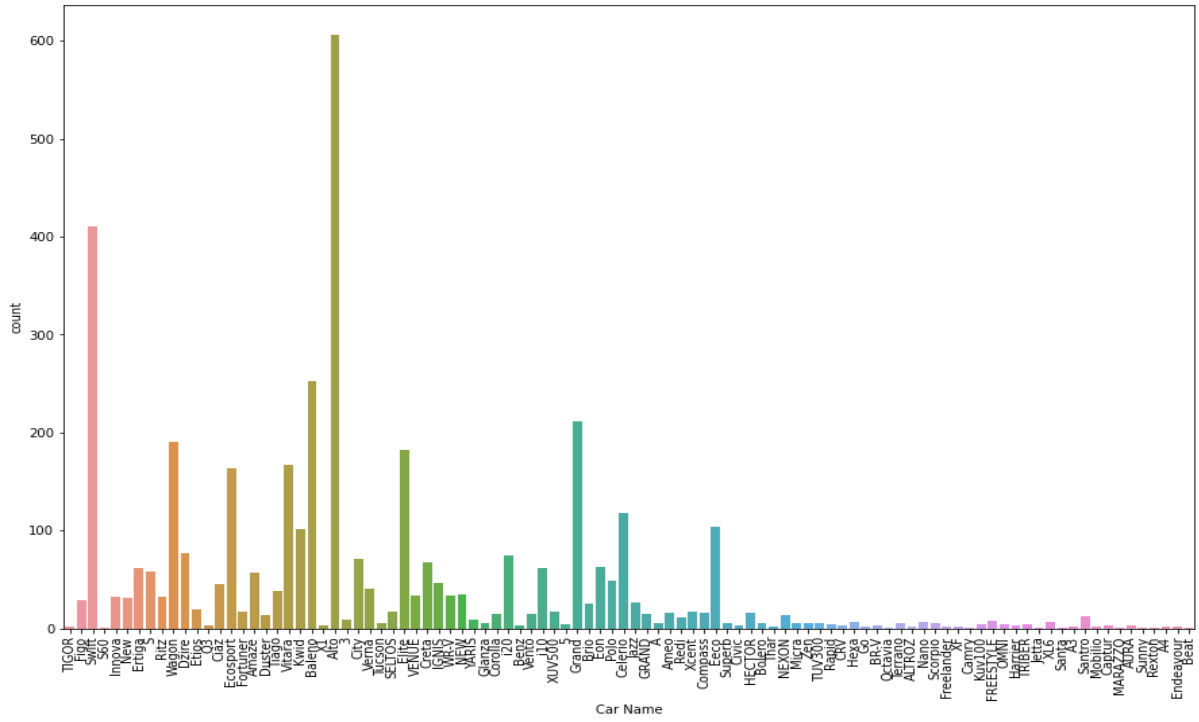
- **Visualization:**




count
fuel
Petrol
Diesel
Petrol + CNG
Petrol + LPG
transmission
MANUAL
AUTOMATIC

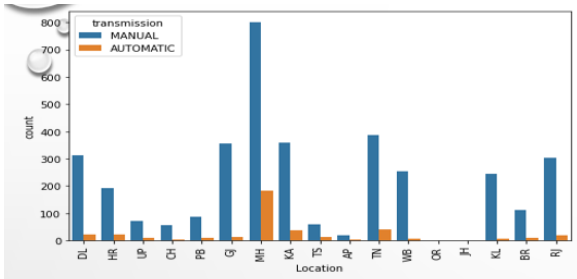

count
Purchased Month
May
Oct
Nov
Dec
Sep
Jun
Apr
Aug
Feb
Jan
Mar
Jul
Purchased Year
2017
2013
2012
2016
2010
2014
2015
2018
2019
2020
2021
2011
2009
2008

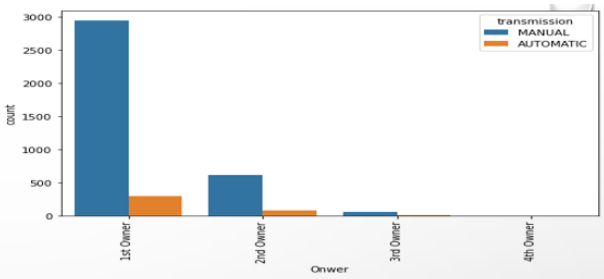

count
Company
Tata
Ford
Maruti
Volvo
Toyota
Audi
Honda
Renault
BMW
Hyundai
KIA
HYUNDAI
Mercedes
Volkswagen
Mahindra
Datsun
Jeep
Skoda
MG
Nissan
Landrover
Jaguar
Ssangyong
Chevrolet

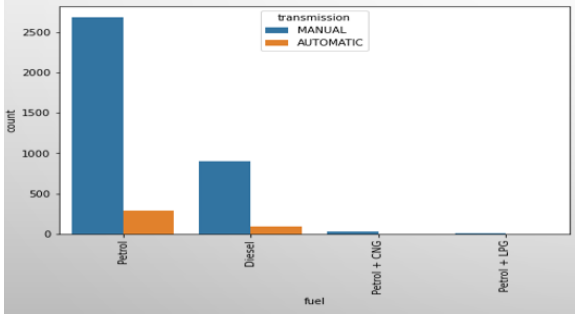


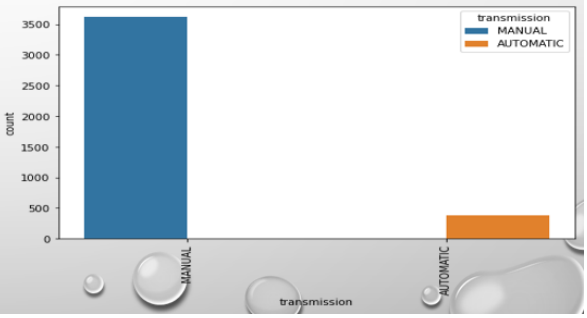
Key Points:

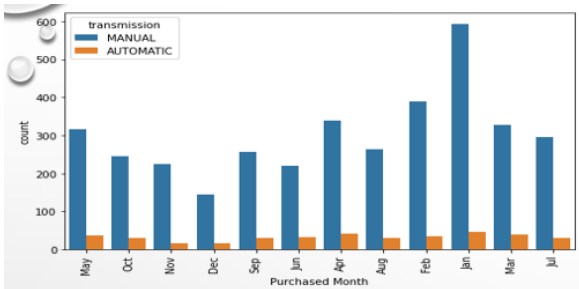
- Majority vehicle is from 'MH' region, and least is from 'JH'
- Maximum used car is of 1st owner and very few car having 4th owner.
- Majority vehicle is of Petrol variant and very less vehicle is having Petrol + LPG
- Maximum car is manual transmission type.
- As maximum vehicle registration month is jan, it may be because of new year month. Accordingly one can apply some special offer in this month for increase sell of vehicle
- Top 3 company cars in market, Maruti, Hyundai, Ford

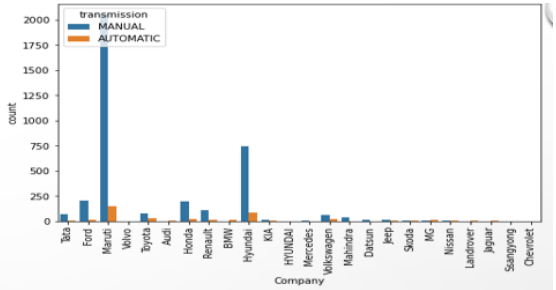

transmission
MANUAL
AUTOMATIC
count
Location
DL
HR
UP
CH
PB
GJ
MH
KA
TS
AP
TN
WB
OR
JH
KL
BR
RJ

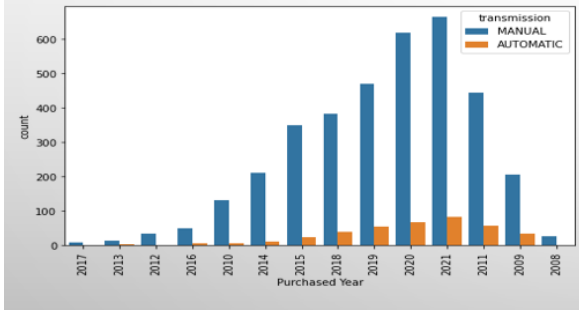

transmission
MANUAL
AUTOMATIC
count
Onwer
1st Owner
2nd Owner
3rd Owner
4th Owner


transmission
MANUAL
AUTOMATIC
count
fuel
Petrol
Diesel
Petrol + CNG
Petrol + LPG

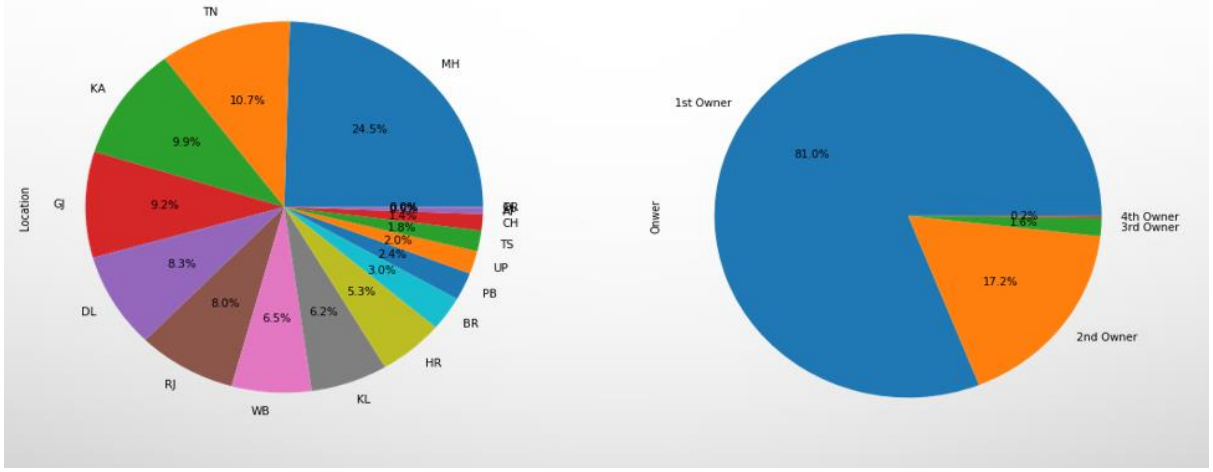

transmission
MANUAL
AUTOMATIC
count
transmission
MANUAL
AUTOMATIC

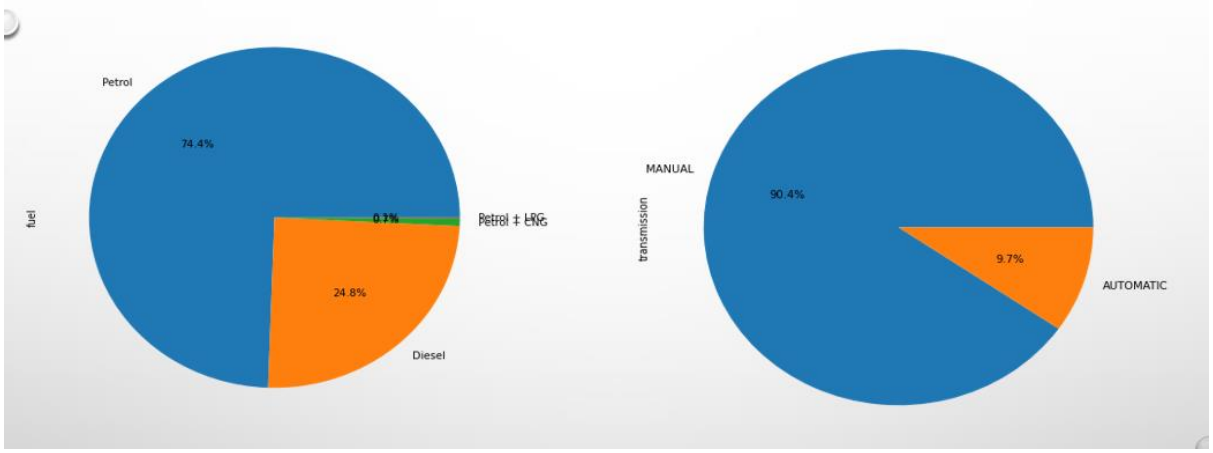

transmission
MANUAL
AUTOMATIC
count
Purchased Month
May
Oct
Nov
Dec
Sep
Jun
Apr
Aug
Feb
Jan
Mar
Jul

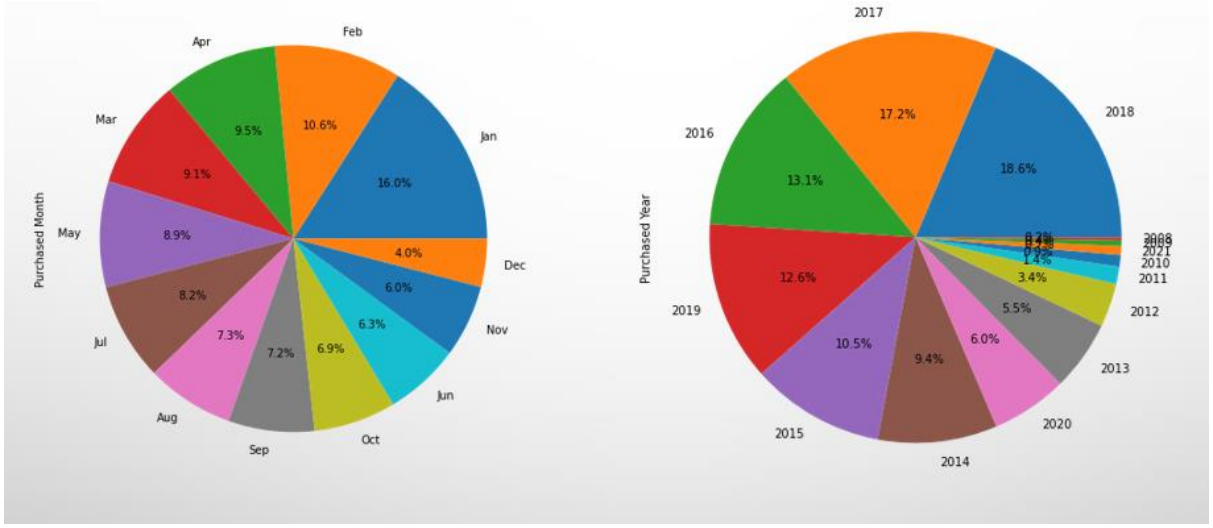

transmission
MANUAL
AUTOMATIC
count
Company
Tata
Ford
Maruti
Volvo
Toyota
Audi
Honda
Renault
BMW
Hyundai
KIA
HYUNDAI
Mercedes
Volkswagen
Mahindra
Datsun
Jeep
Skoda
MG
Nissan
Landrover
Jaguar
Ssangyong
Chevrolet

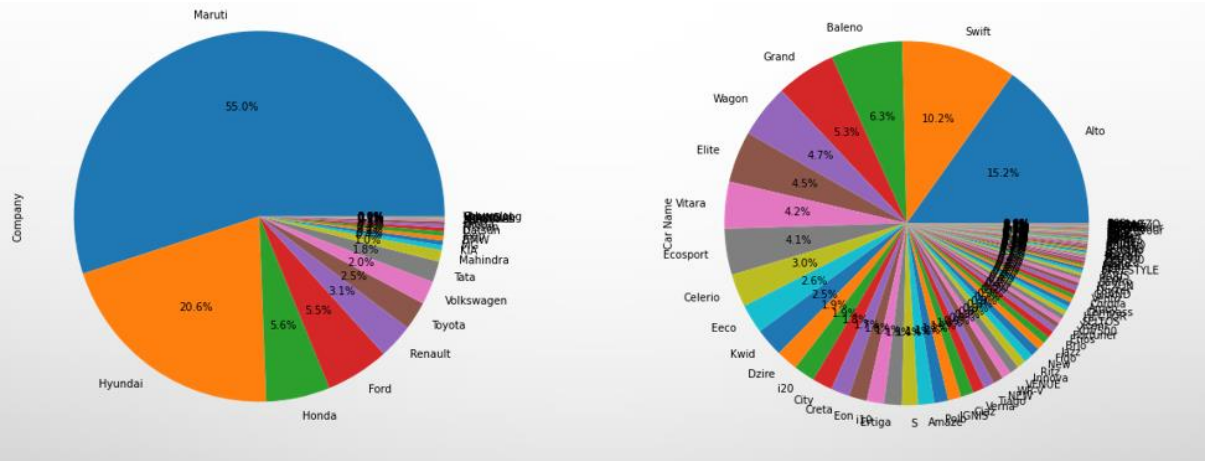

transmission
MANUAL
AUTOMATIC
count
Purchased Year
2017
2013
2012
2016
2010
2014
2015
2018
2019
2020
2021
2011
2009
2008

PIE CHART FOR VARIABLES

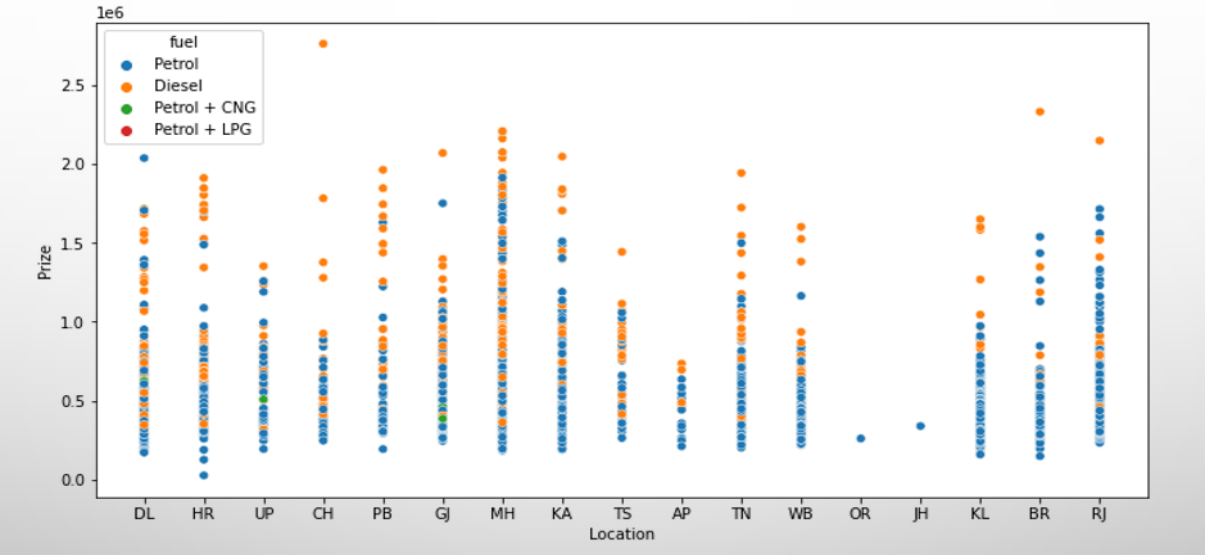


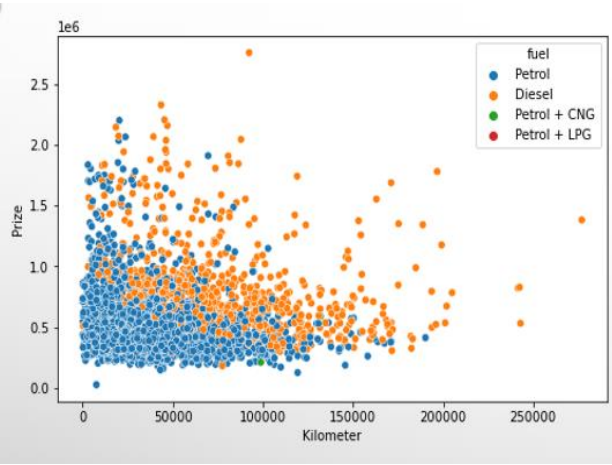


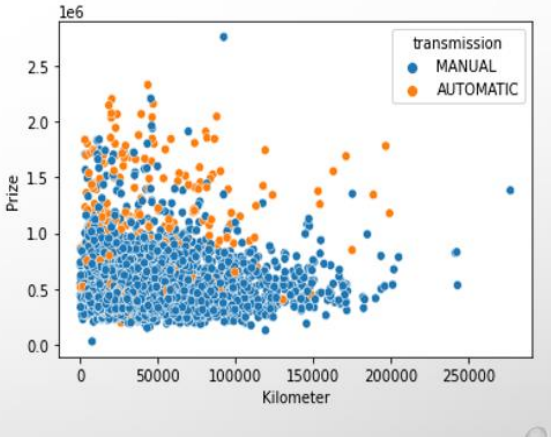


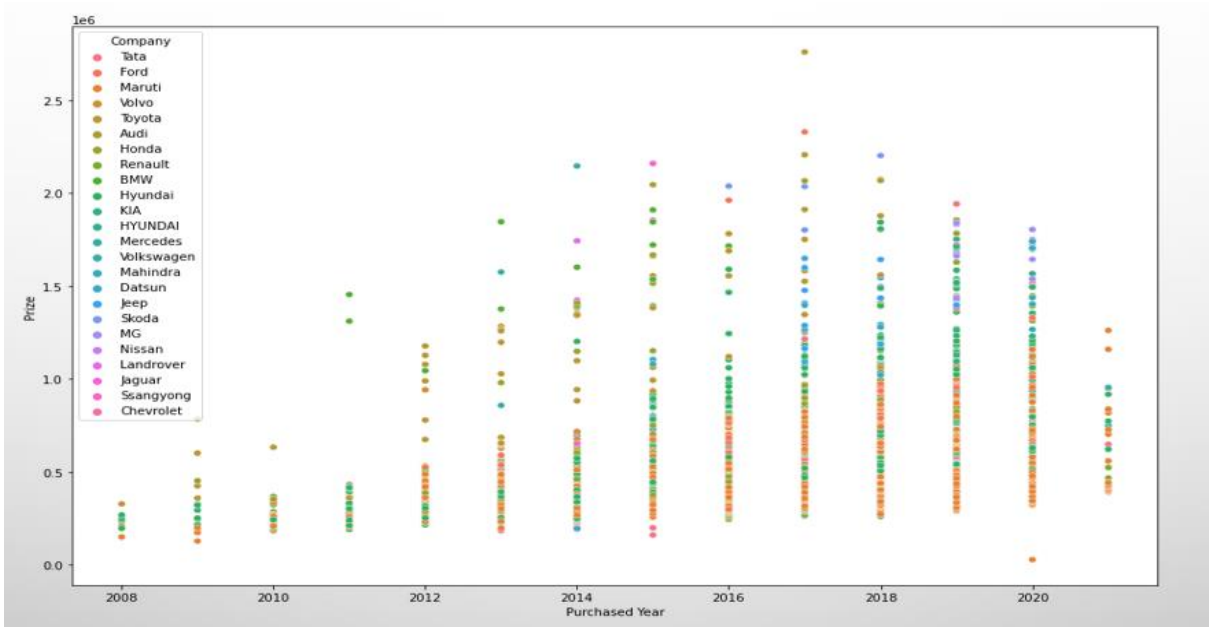


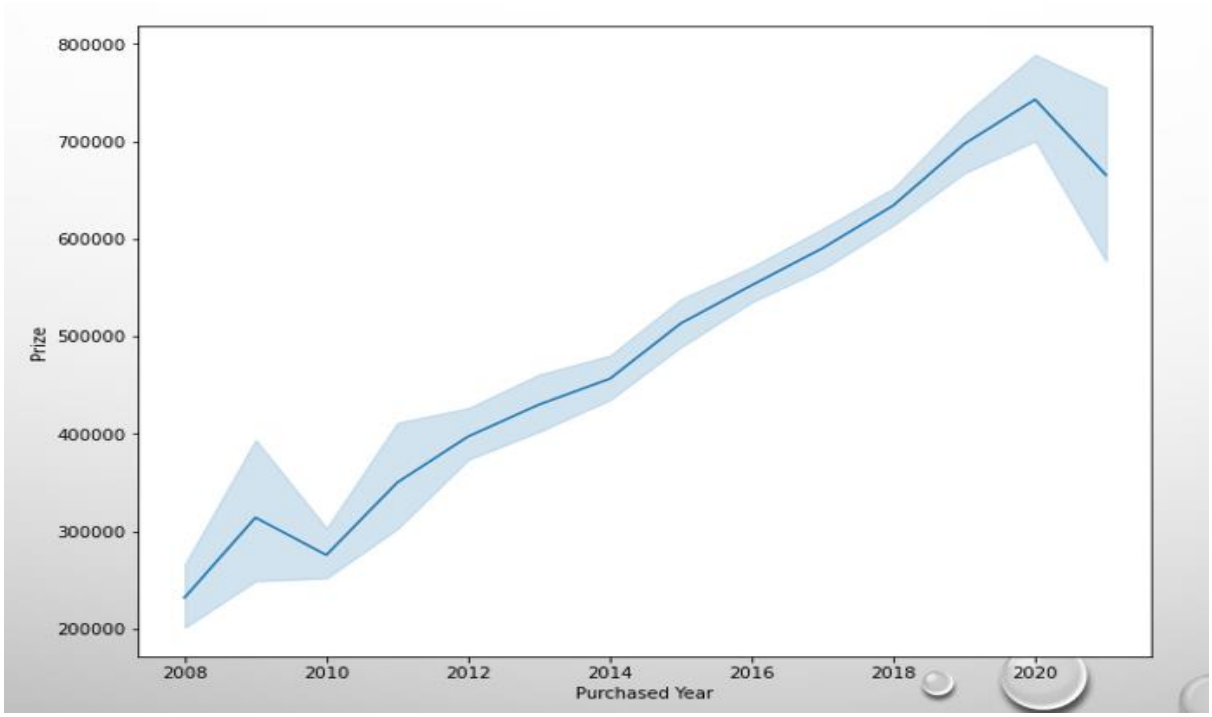
BIVARIATE ANALYSIS

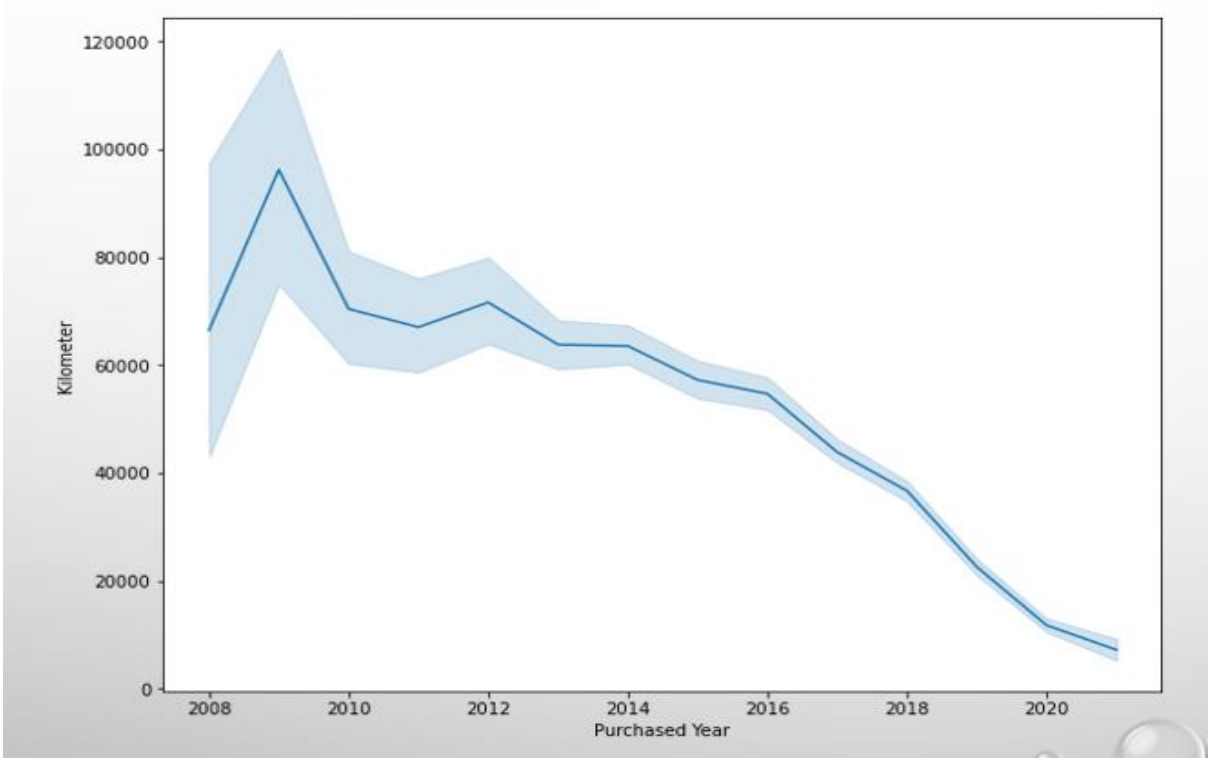







1e6
Company
Tata
Ford
Maruti
Volvo
Toyota
Audi
Honda
Renault
BMW
Hyundai
KIA
HYUNDAI
Mercedes
Volkswagen
Mahindra
Datsun
Jeep
Skoda
MG
Nissan
Landrover
Jaguar
Ssangyong
Chevrolet
Prize
Purchased Year

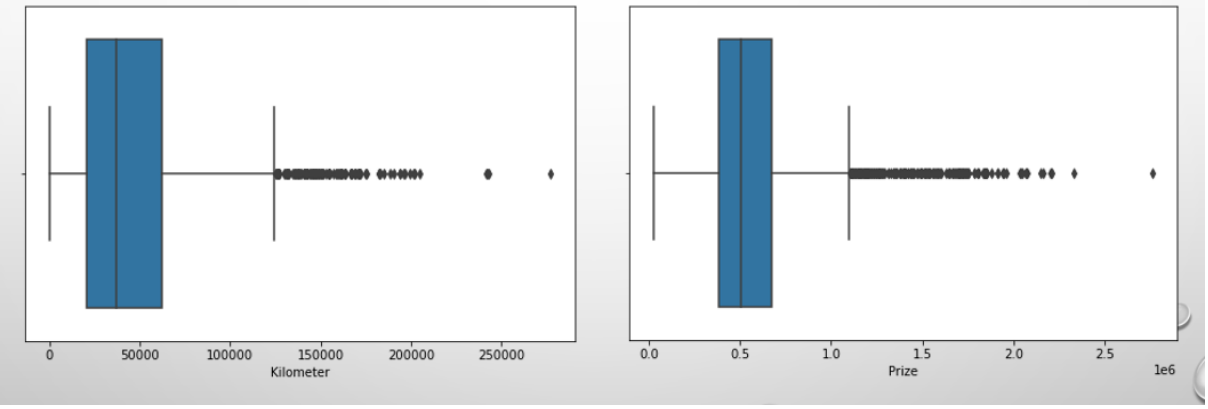

Prize
Purchased Year



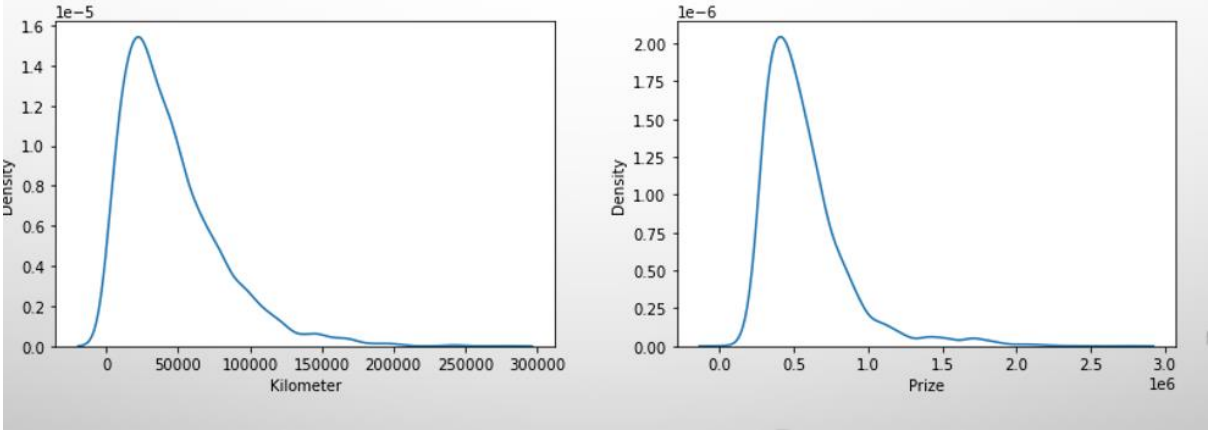
Key points:

- We can see, highest prize is of diesel cars and cars which are having petrol + CNG & LPG are having less low prize
- Automatic and Diesel Variant cars are having higher rate as compare to other variants car.
- Volvo car is having highest value, and maximum car is of Hundai, Maruti and etc.
- Prizes are going up with increase in model year, which is very obvious as newer the model prize will increase accordingly.
- As much newer model car one will choose, that much less driven car would be available.

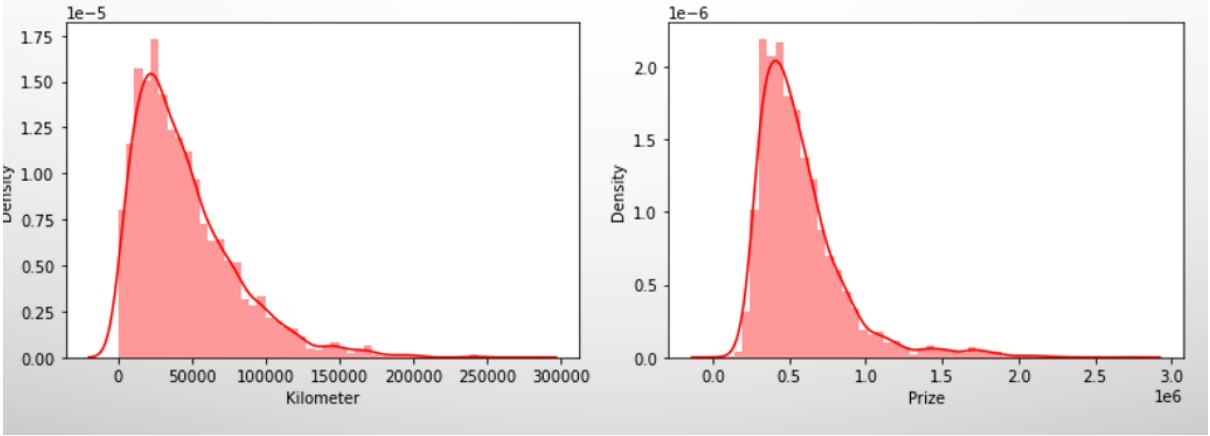
OUTLIERS

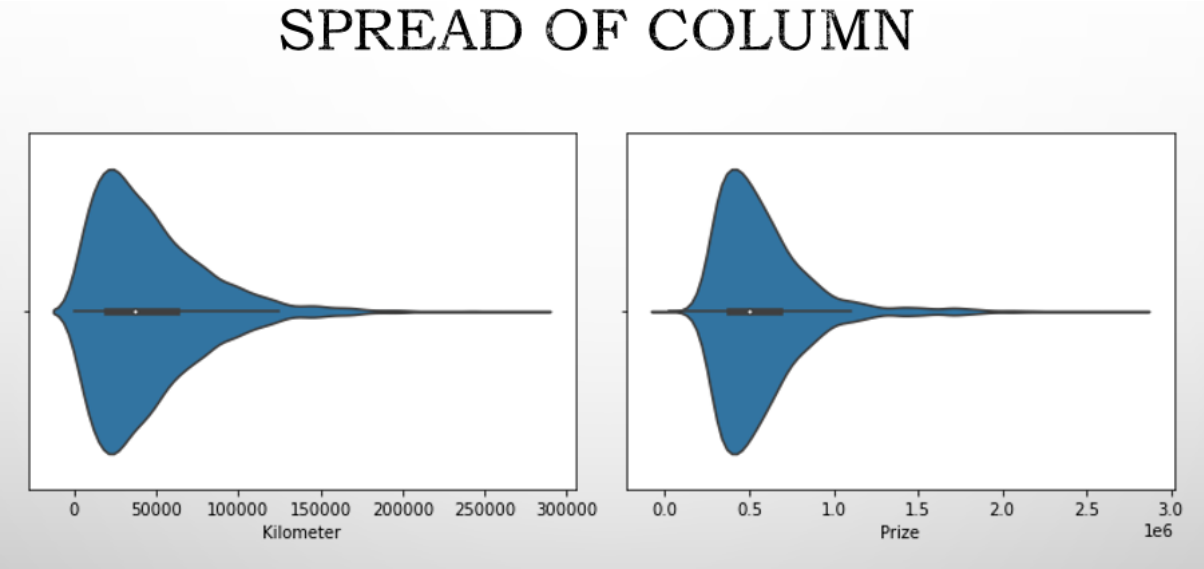


SKEWNESS



DATA DISTRIBUTION

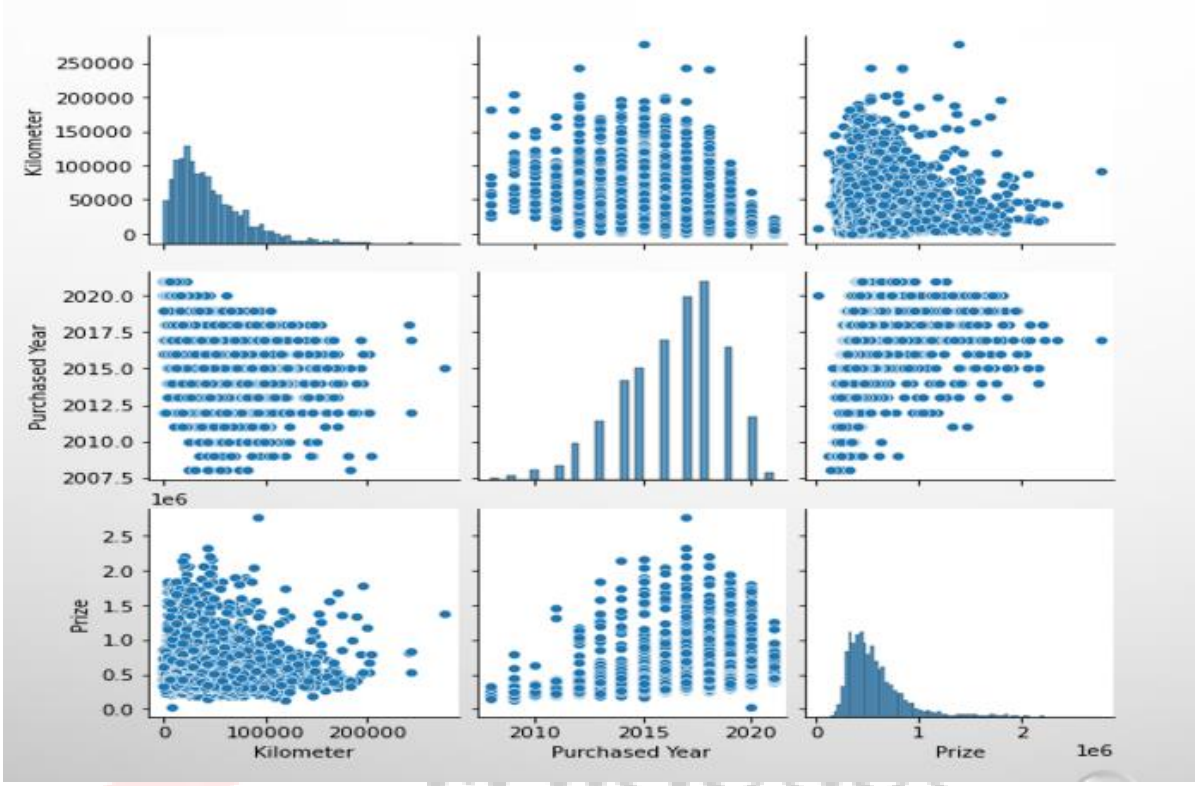




 # Key Points:

- Outlier present in the both columns (Kilometer and Prize)
- These both columns are left skewed
- In the distribution and spread visualization, one can see, where the highest volume is present, accordingly make suitable model

PAIR PLOT OF DATASET



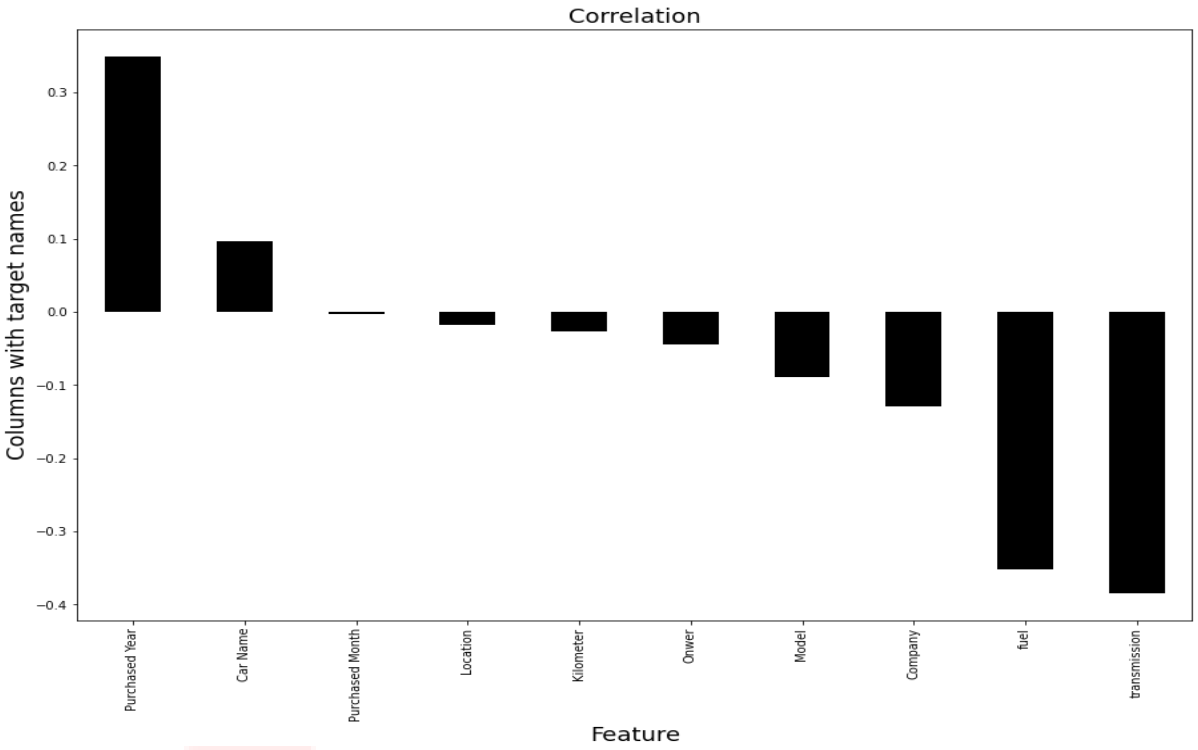
- **Encoding of dataset:**

```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()   # created instance of Ordinal Encoder

oe.fit(encoded_df[object_col])
encoded_df[object_col] = oe.transform(encoded_df[object_col])  # Transforming  df

print(encoded_df.shape)

 (4000, 11)
```

We have to encode feature column, therefore I have used ordinal encoding.

I have also checked model performance by applying OneHotEncoding but ordinal encoding is giving best result as compare to OneHotEncoding.

- **Data Inputs- Logic- Output Relationships**



We can see in above diagram, Purchased year column is positively correlated with target variable and Car Name is also positively correlated with target variable.

Transmission and fuel is negatively correlated with target variable.

By this graph, one can easily see that increase in purchased year will to leads to higher the car prize.

Kilometre column is also say that, as much less driven car available, it will leads to increase the prize of car.

- **State the set of assumptions**

OUTLIERS:

Outliers are not removed from the dataset because it was giving high loss of data; therefore I have worked on existing dataset without removing outliers

1. Try zscore technique

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

```
z = np.abs(zscore(encoded_df) )
df_z = encoded_df[(z < 3).all(axis = 1)]
df_z.shape

# (3392, 11)
```

```
(3392, 11)
```

```
(encoded_df.shape[0] - df_z.shape[0] ) / encoded_df.shape[0]*100
#  15.2
# Heavy data loss we getting by this method
```

```
15.2
```

2. IQR Technique

```
Q1 = encoded_df.quantile(0.25)
Q3 = encoded_df.quantile(0.75)
IQR = Q3 - Q1
```

```
df_IQR = encoded_df[~((encoded_df < (Q1 - 1.5*IQR) ) | (encoded_df > (Q3 + 1.5*IQR) )).any(axis = 1) ]
df_IQR.shape
# (2111, 11)
```

```
(2111, 11)
```

```
(encoded_df.shape[0] - df_IQR.shape[0] ) / encoded_df.shape[0]  * 100
# 47.225 (Heavy dataset)
```

```
47.225
```

As both method of removing outliers, IQR and zscore is giving high loss data, therefore, I choose to work with existing data without removing outliers.

.

Seperating dataset into x1 and y1 form

```
x = encoded_df.drop(columns= ['Prize'])
y = encoded_df['Prize']
print('shape of x', x.shape)
print('shape of y', y.shape)
```

```
shape of x (4000, 10)
shape of y (4000,)
```

Separating Dataset into x and y form, where x is feature and y is target variable

Treatment of Skewness of columns:

Removing Skewness

```
x['Kilometer'].skew()
```

```
1.4615216623851044
```

```
# apply sqrt (1 + x) transfornation
x['Kilometer'] = np.sqrt(1 + x['Kilometer'])
x['Kilometer'].skew()
```

```
0.3807329793335966
```

I have applied many skewness treatment on this dataset for removing skewness. After applying many skewness treatment this applied treatment is working best to remove skewness.

Multicollinearity

Multicollinearity is removed using variance inflation factor,

Here cal_vif is a my created function to calculate vif value of column, this function is created using variance inflation factor imported from statsmodels library

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
# function to calculate VIF
def cal_vif(data):
    vif = pd.DataFrame()
    vif['Columns Name'] = data.columns    # columns name
    vif['VIF'] = [variance_inflation_factor(data.values, i) for i in range(data.shape[1])]
    return (vif)
```

```
cal_vif(x)
```

	Columns Name	VIF
0	Kilometer	7.051615
1	Location	5.268099
2	Onwer	1.249373
3	fuel	4.262860
4	transmission	8.848963
5	Purchased Year	9.476249
6	Purchased Month	3.405281
7	Company	7.772986
8	Car Name	3.172878
9	Model	4.391370

As we can see, kilometre column is showing vif value under the acceptable range.

And other columns are of encoded form.

Scaling:

Scaling of column, this step apply just before applying Machine learning of dataset, by followed this I have applied this Standard Scaling technique to the dataset

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()   # Instance of Standard Scaler
```

```
# Scalling Training dataset
x[x.columns] = ss.fit_transform(x[x.columns])
x.head()
```

	Kilometer	Location	Onwer	fuel	transmission	Purchased Year	Purchased Month	Company	Car Name	Model
0	0.874380	-1.375488	-0.460178	0.534582	0.326813	0.246443	0.827512	1.796298	1.087378	0.374252
1	0.981957	-1.375488	-0.460178	-1.702163	0.326813	-1.387661	1.432805	-1.768891	-0.247770	-1.576696
2	1.903459	-1.375488	1.733759	-1.702163	0.326813	-1.796188	1.130158	0.459352	1.053999	0.804522
3	0.071649	-1.375488	1.733759	-1.702163	-3.059852	-1.796188	-0.988368	2.464771	0.820348	0.586440
4	1.034447	-1.375488	-0.460178	-1.702163	0.326813	-1.796188	1.130158	0.459352	1.053999	-0.492181

```
print(y.shape)
print(x.shape)
```

```
(4000,)
(4000, 10)
```

➢ Model/s Development and Evaluation

In Machine Learning, I have applied 4 base models of regression problem, after this I have also applied boosting techniques over this dataset and then selected one of them based on their performance.

After choosing best model, Hyper Parameter tuning is applied over it to increase the accuracy of model.

In final, finalize the a final model for this dataset based on its performace.

In this machine learning process, I have created a function named "ML_Model" for smooth functioning of machine learning,

This function will give Training and testing data accuracy along with Mean Squared error and Mean Absolute Error and cv score on different-different cross fold values.

 # Libraries used in process of Machine Learning:

Machine learning

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost
from xgboost import XGBRegressor
```

creating instances for ML model and storing them into a list of models.

After storing instances into model list, apply ML_Model function with parameter model and dataset (x, y)

```
lr = LinearRegression()
dtr = DecisionTreeRegressor()
svr = SVR()
knn = KNeighborsRegressor()

models = [lr, dtr, svr, knn]
ML_Model(models, x, y)
```

Here, ML_Model is a my created function which will give performance of the model based on various performing factor accuracy, MSE, MAE and cv score.

 # Linear Regression Model Performance :

```
Training accuracy is :  0.4350592287657771
Testing accuracy is : 0.4255793536491609
--------------------------------------------------------------------------------
Mean Squared Error:  48228697313.85908
Mean Absolute Error:  157029.39984513258
--------------------------------------------------------------------------------
Cross value score
cv score 0.33056575774727853 at 2 cross fold
cv score 0.36654274760044786 at 3 cross fold
cv score 0.37435634843619253 at 4 cross fold
cv score 0.37172094518538434 at 5 cross fold
cv score 0.3704259392173203 at 6 cross fold
cv score 0.3554010298258235 at 7 cross fold
--------------------------------------------------------------------------------
```

This model is giving less accuracy and having more difference between cv score and accuracy.

 # DecisionTreeRegressor Model Performance:

```
Training accuracy is :  1.0
Testing accuracy is : 0.7487004390662537
--------------------------------------------------------------------------------
Mean Squared Error:  21085165372.728333
Mean Absolute Error:  70363.96833333334
--------------------------------------------------------------------------------
Cross value score
cv score 0.6169203774149563 at 2 cross fold
cv score 0.6960307228277852 at 3 cross fold
cv score 0.6665195851724816 at 4 cross fold
cv score 0.6913733121271134 at 5 cross fold
cv score 0.7266191753969157 at 6 cross fold
cv score 0.7430886710007455 at 7 cross fold
--------------------------------------------------------------------------------
```

This model is performing well as compare to previous one, but it is giving overfitted model.

 # Support Vector Regressor Model Performance:

```
Training accuracy is :  -0.050073128347731144
Testing accuracy is : -0.0479902666478802
--------------------------------------------------------------------------------
Mean Squared Error:  87989882813.44049
Mean Absolute Error:  194539.22195303207
--------------------------------------------------------------------------------
Cross value score
cv score -0.10465866852104222 at 2 cross fold
cv score -0.0717667479502609 at 3 cross fold
cv score -0.08237110313923168 at 4 cross fold
cv score -0.08578988675032351 at 5 cross fold
cv score -0.09253776444830998 at 6 cross fold
cv score -0.103825337394894 at 7 cross fold
--------------------------------------------------------------------------------
```

This model is not giving suitable for this dataset because it is working anonymously.

 # KNeighborsRegressor Model Performance

```
Training accuracy is :  0.7427100565451255
Testing accuracy is : 0.6392435340680314
--------------------------------------------------------------------------------
Mean Squared Error:  28118665431.6241
Mean Absolute Error:  107217.79550000001
--------------------------------------------------------------------------------
Cross value score
cv score 0.5007349757502145 at 2 cross fold
cv score 0.5344867656676786 at 3 cross fold
cv score 0.5380710390379686 at 4 cross fold
cv score 0.5481975595098747 at 5 cross fold
cv score 0.5519031666847227 at 6 cross fold
cv score 0.5363457375256294 at 7 cross fold
--------------------------------------------------------------------------------
```

This model is also giving over fitted model, hence cannot be selected as final model

 # Applied Boosting Techniques:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
rfr = RandomForestRegressor()
gbr = GradientBoostingRegressor()
abr = AdaBoostRegressor()

models = [rfr, gbr, abr]
ML_Model(models, x, y)
```

Here, RandomForestRegressor is a bagging technique and Gradient and AdaBoost are boosting techniques.

 # RandomForestRegressor Model Performance:

```
Training accuracy is :  0.9763372544561276
Testing accuracy is : 0.8944821994695854
--------------------------------------------------------------------------------
Mean Squared Error:  8167540452.821001
Mean Absolute Error:  50545.04429166667
--------------------------------------------------------------------------------
Cross value score
cv score 0.7741528062498255 at 2 cross fold
cv score 0.8075872765636823 at 3 cross fold
cv score 0.8151560491811716 at 4 cross fold
cv score 0.8332866759121828 at 5 cross fold
cv score 0.8383370422251527 at 6 cross fold
cv score 0.8352435635298693 at 7 cross fold
--------------------------------------------------------------------------------
```

Random Forest model is giving good accuracy but giving overfitted model, hence can't consider this model as final model.

 # AdaBoostRegressor Model Performance:

```
Training accuracy is :  0.56461782792539
Testing accuracy is : 0.49781864481647464
--------------------------------------------------------------------------------
Mean Squared Error:  43241940912.88443
Mean Absolute Error:  165998.6962162819
--------------------------------------------------------------------------------
Cross value score
cv score 0.18536591862101337 at 2 cross fold
cv score 0.20477608286167226 at 3 cross fold
cv score 0.1727543756907936 at 4 cross fold
cv score 0.24505550906302562 at 5 cross fold
cv score 0.2390700636164127 at 6 cross fold
cv score 0.21676227221989633 at 7 cross fold
--------------------------------------------------------------------------------
```

This model is giving very low accuracy as like linear model, hence cannot consider for final model.

 # XGBoostRegressor Model Performance:

```
Training accuracy is :  0.9959833963659347
Testing accuracy is : 0.9151101491368179
--------------------------------------------------------------------------------
Mean Squared Error:  6570846695.758522
Mean Absolute Error:  46898.805572916666
--------------------------------------------------------------------------------
Cross value score
cv score 0.8090434036845229 at 2 cross fold
cv score 0.8398739360945041 at 3 cross fold
cv score 0.8520468091702995 at 4 cross fold
cv score 0.869388373380637 at 5 cross fold
cv score 0.8684655069411166 at 6 cross fold
cv score 0.8692838309053256 at 7 cross fold
--------------------------------------------------------------------------------
```

Not much, but this model is also giving over fitted model.

 # GradientBoostingRegressor Model Performance:

```
Training accuracy is :  0.8800591995910687
Testing accuracy is : 0.8615330347882328
--------------------------------------------------------------------------------
Mean Squared Error:  11677872635.260414
Mean Absolute Error:  69358.43958569485
--------------------------------------------------------------------------------
Cross value score
cv score 0.7605642251638078 at 2 cross fold
cv score 0.7785083951503461 at 3 cross fold
cv score 0.787201497024516 at 4 cross fold
cv score 0.7899967002260997 at 5 cross fold
cv score 0.7985380866304629 at 6 cross fold
cv score 0.8004503447417923 at 7 cross fold
--------------------------------------------------------------------------------
```

Here, we are getting almost equal accuracy of training and testing dataset. And also not having much difference in cv value and accuracy of testing data.

This model can be consider as final model.

Models	Training Accuracy	Testing Accuracy	CV Score	Difference
# RandomForestRegressor	0.976337	0.894482	0.838337	0.056145
# GradientBoosting Regressor	0.880059	0.861533	0.80045	0.061083
# AdaBoostRegressor	0.564617	0.497818	0.245055	0.252763
# XGBRegressor	0.995983	0.91511	0.869388	0.045722

 # RandomeForestRegressor: Overfitted (train acc. > test acc.)

 # GradientBoostingRegressor: Good

 # AdaBoostingRegressor: Less accuracy

 # XGBRegressor: Less Overfitted (train acc. > test acc.)

 # Ensemble Techniques :

Based on model performance, we can see, GradientBoosting is performing best as compare to other ML Models, Its cv value is also very close to accuracy of model.

Hence I had applied hyper parameter tuning for this Model to increase the accuracy of model

```
from sklearn.model_selection import GridSearchCV
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 34)
```

```
parameter = {'alpha' : [0.9, 0.09, 0.1],
             'learning_rate' : [0.1, 0.01],
             'max_depth' : [ 3, 4, 5],
             'min_samples_leaf' : [1, 2, 3],
            'min_samples_split': [2,3,4],
            'n_estimators': [100, 50, 10]}
```

```
gcv = GridSearchCV(estimator = GradientBoostingRegressor(), param_grid = parameter, cv = 7)
gcv.fit(x_train, y_train)
```

```
GridSearchCV(cv=7, estimator=GradientBoostingRegressor(),
             param_grid={'alpha': [0.9, 0.09, 0.1],
                         'learning_rate': [0.1, 0.01], 'max_depth': [3, 4, 5],
                         'min_samples_leaf': [1, 2, 3],
                         'min_samples_split': [2, 3, 4],
                         'n_estimators': [100, 50, 10]})
```

```
gcv.best_params_
```

```
{'alpha': 0.9,
 'learning_rate': 0.1,
 'max_depth': 5,
 'min_samples_leaf': 3,
 'min_samples_split': 4,
 'n_estimators': 100}
```

```
gbr = GradientBoostingRegressor(alpha = 0.9, learning_rate = 0.1, max_depth = 5, min_samples_leaf=3,min_samples_split=4 ,
                                n_estimators = 100)

models = [gbr]
ML_Model(models, x, y)
```

```
_____________________________________________________________________________
GradientBoostingRegressor(max_depth=5, min_samples_leaf=3, min_samples_split=4)  is giving best accurancy 0.8982800377021298 on
random state of 10
Training accuracy is :  0.9599412808872572
Testing accuracy is : 0.9083251353168631
-------------------------------------------------------------------------------
Mean Squared Error:  8239306946.623017
Mean Absolute Error:  52627.709933710146
-------------------------------------------------------------------------------
Cross value score
cv score 0.8180102342450999 at 2 cross fold
cv score 0.8350164421398629 at 3 cross fold
cv score 0.8417221212146166 at 4 cross fold
cv score 0.8542132539134897 at 5 cross fold
cv score 0.8541844069219681 at 6 cross fold
cv score 0.8575650467925049 at 7 cross fold
-------------------------------------------------------------------------------
```

After applying hyper parameter tuning, we can see, accuracy is increased from 86% to 90%.

Final Model:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 10)

final_model = GradientBoostingRegressor(alpha = 0.9, learning_rate = 0.1, max_depth = 5, min_samples_leaf=3,min_samples_split=4
final_model.fit(x_train, y_train)
final_pred = final_model.predict(x_test)
final_pred_train = final_model.predict(x_train)

train_accuracy = r2_score(y_train, final_pred_train )
test_accuracy = r2_score(y_test, final_pred )
print('Training accuracy: ', train_accuracy)
print('Testing accuracy: ', test_accuracy)
print('_____________________________________________')
print('Mean squared error: ', mean_squared_error(y_test, final_pred ) )
print('Mean absolute error: ', mean_absolute_error(y_test, final_pred) )
```

```
Training accuracy:  0.9599412808872573
Testing accuracy:  0.9086278762724012
_____________________________________________
Mean squared error:  8212098009.182926
Mean absolute error:  52578.25530144247
```

```
Cross value score
cv score 0.8180102342450999 at 2 cross fold
cv score 0.8350164421398629 at 3 cross fold
cv score 0.8417221212146166 at 4 cross fold
cv score 0.8542132539134897 at 5 cross fold
cv score 0.8541844069219681 at 6 cross fold
cv score 0.8575650467925049 at 7 cross fold
--------------------------------------------------------------------------------
```

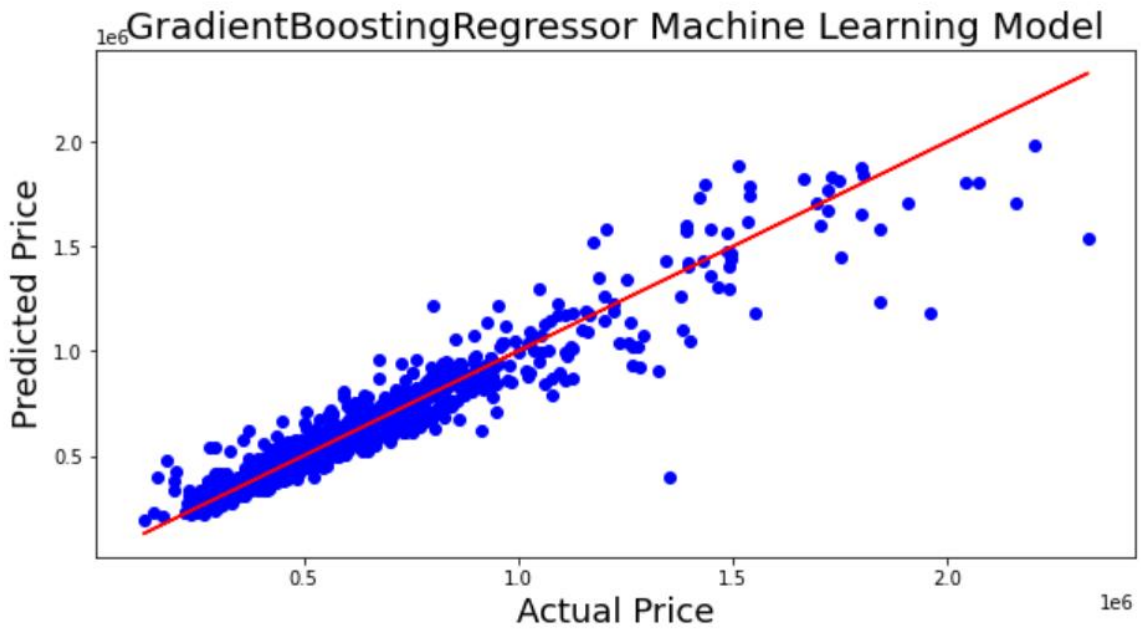
Graph for Model Performance:

```
plt.figure(figsize = (10, 5))
plt.scatter(x = y_test, y = final_pred, color = 'b')
plt.plot(y_test, y_test, color = 'r')
plt.xlabel('Actual Price', fontsize= 18 )
plt.ylabel('Predicted Price', fontsize = 18)
plt.title('GradientBoostingRegressor Machine Learning Model', fontsize = 20)
```

In below mentioned chart, we can see how model is performing,

As our model is having 90% accuracy based on this, its chart is visible to us

```
Text(0.5, 1.0, 'GradientBoostingRegressor Machine Learning Model')
```



Deploying & Loading Model:

Deploy Model

```
import pickle
filename = 'car_price_predictor.pkl'
pickle.dump(final_model, open(filename, 'wb'))
```

Loading model

```
load_model =  pickle.load(open('car_price_predictor.pkl', 'rb'))
result = load_model.score(x_test, y_test)
print(result)
```

```
0.9086278762724012
```


Conclusion:

```
predicted = np.array(load_model.predict(x_test))
original = np.array(y_test)
# convert columns in to np.array
```

```
round(predicted[0],2)
for i in predicted:
```

Conclusion DataFrame

```
conclusion = pd.DataFrame({'Actual Price': original, 'Predicted Price': predicted},
                          index = range(len(original)))
```

```
conclusion['Predicted Price']= conclusion['Predicted Price'].apply(lambda x: round(x, 2))
```

```
conclusion.sample(10)
```

	Actual Price	Predicted Price
479	308199.0	304422.46
171	698799.0	685515.40
1032	843999.0	849936.76
472	434399.0	416783.03
22	611999.0	615455.59
1102	469599.0	450540.38
198	317099.0	350318.05
857	593099.0	589082.49
250	385599.0	406587.78
953	439099.0	517453.05

➢ Hardware and Software Requirements and Tools Used

All used libraries :

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost
from xgboost import XGBRegressor
```

Pandas: This library used for dataframe operations

Numpy: This library gives statistical computation for smooth functioning

Matplotlib: Used for visualization

Seaborn: This library is also used for visualization

Sklearn: This library having so many machine learning module and we can import them from this library

Pickle: This is used for deploying the model

Imblearn: This library is import to get SMOTE technique for balance the data

Scipy: It is import to perform outlier removing technique using zscore

Warning: To avoid unwanted warning shows in the output

I am giving this requirement and tool used, based on my laptop configuration.

Operating System:	**Window 11**
RAM:	**8 GB**
Processor :	**i5 10th Generation**
Software:	**Jupyter Notebook,**

➢ Observations from the whole problem.

i) Few columns are created from title column, as it was containing various details, like, car name, company, which model.

ii) Dataset was having duplicate records.

iii) Missing cell was hide by "-", I have removed all missing value by replace it with null value.

iv) Multicollinearity was not present in the dataset

v) Skewness is removing by using square root method

vi) Support Vector Repressor is not working on this dataset and etc.

➢ Learning Outcomes of the Study in respect of Data Science

My learnings :- the power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important step to remove missing value or null value.

Various algorithms I have used in this dataset and to get out best result and save that model. The best algorithm is GradientBoostingClassifier.

The challenges I faced while working on this project is outliers removing techniques.

As it is very large dataset, to scrape, from websites.

 # For scraping the data, it had taken almost 7 to 8 hours' time for execution.

.

➢ Limitations of this work and Scope for Future Work

Need to train this same model with data in Lacs or more than that, then only it would be become best model and it can perform in any region of India.