Car Price Prediction Project

Submitted by:

Harsh Prasad

ACKNOWLEDGMENT

Working on Car Price prediction project was interesting. I learnt a lot in from this project, especially, extracting data from the multiple sources and then cleaning them in the model building phase. The data set was collected from open-source websites like olx, cardekho and cars24. This data is used to build a model that will help our client in predicting new car price of used cars, which got affected due to covid-19 outbreak.

I would like to express my sincere gratitude to our internship coordinator, Mr Sajid Choudhary, who have given their valuable time and given me chance to learn something despite having their busy schedule and his great guidelines for internship.

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.

  The pandemic-induced lockdown resulted in the shutting down of production at original equipment manufacturers (OEM). It also led to disruption of the entire value chain of major industries in India, and therefore negatively affected production of auto spare parts in micro, small and medium-sized industries. In addition, the reduction in consumer demand for passenger vehicles contributed to a loss in revenue and a severe liquidity crisis in the sector.

  According to the Society of Indian Automobile Manufacturers, the sector registered negative growth in sales of all vehicle categories in FY21 (2.24% decline in sales of passenger vehicles, 13.19% fall in sales of two-wheelers, 20.77% fall in sales of commercial vehicles, and 66.06% fall in sales of three-wheelers).

  Used cars will continue to see a faster recovery in demand and much will depend on the supply catching up to the same.

  After facing back-to-back setbacks because of demonetisation and GST, the used car market has witnessed healthy demand momentum and has outperformed new car sales in the last two years.

  With the change in market due to covid 19 impact, previous car price valuation machine learning models are not working efficiently. So, we are looking for new machine learning models from new data

We propose to use machine learning algorithms and artificial intelligence techniques to develop an algorithm that can predict used car prices based on certain input features.

- ## Conceptual Background of the Domain Problem

The automotive and mobility industries have certainly been among the hardest hit during the COVID-19 pandemic. The picture is improving, however. Car dealerships are getting busier, and many are eagerly seeking more inventory to sell. Overall mobility is picking up steadily, although not to pre-COVID-19 levels. Usage of shared mobility services and public transit is picking up significantly, while regions where many commuters and their employers accept the practicalities of working from home are recovering more slowly.

Prospective buyers are less inclined to want to interact with sellers at car dealerships. That decline in preference is falling across all regions and age groups—especially for consumers between 55 and 70 years of age, who now consider online buying as a relevant alternative to visiting dealers.

Also, Public transport and shared mobility modes are considered more or less safe again with regard to COVID-19 infection. People are wary of taking public transport for commuting. Small family cars are in prime demand.

The increased preference for personal mobility due to safety concerns amid the Covid-19 pandemic has led to supply constraints in the used car market, as people are holding on to their cars longer.

Meanwhile, demand for used cars has increased rapidly compared to pre-Covid times, and this mismatch between demand and supply has led to a 2-7 per cent increase in the price of used cars, said experts.

The demand for used cars has certainly increased from pre-pandemic times. The used vehicle market is facing supply constraints due to three factors: customers are holding on to their used vehicles and

not selling them as they were doing before the pandemic, exchanges have been impacted due to continued challenges in the new car market, and repossessions have virtually stopped since April due to the ongoing loan moratorium. Due to supply constraints and unpredictable nature of the lockdown, sales were impacted during the first few months.

While the pandemic has led to an increased requirement of personal mobility, this has been met with lower expected income amid the economic distress.

This has led to a downward telescoping of demand amid the pandemic, resulting in greater demand for used cars, whilst also leading people to not sell their old cars in the market, which is the source of supply in the pre-owned car market.

This mismatch between demand and supply has led to a 6-7 per cent increase in the buying price of used cars, while the selling price has increased by a bit more.

- Review of Literature

An article entitled "The Determinants of Used Rental Car Prices" reports predictably that both car age and mileage contribute negatively to resale value. I The article further mentions car make and quality and the impact these variables have on used car prices. These findings support the predicted signs of the coefficients in a regression model equation. Both age and mileage are mentioned as negatively contributing variables to resale price of used rental cars. Further, there is mention of seasonal differences in resale prices, something that I had not considered to be a major variable in the process.

A study found in a 2003 issue of McKinsey Quarterly briefly covers car quality and its impact on consumers. The authors address product appeal, and thus demand, and how incentives can identify the appeal. The article states that "U.S. companies in 2002 have increased their incentives to a staggering 14 percent of sale prices-

twice as high as those offered by foreign competitors, on average. Even so, the U.S. market share of the Big Three has slipped over this period, by a combined 1. 5 percent a year.One way to view the role of the incentives in this case is that they serve to even out demand. A car company that offers more incentives in the form of discounts and rebates is likely doing so to make its products more appealing to purchase. Therefore, a company offering more incentives than another is likely receiving a lower level of demand. It can be inferred, then, that US companies face a smaller and even declining demand against foreign producers and need to offer the incentives to equalize demand.

Beyond how old a car is, how many miles it has been driven, and its make, a record of how well the car has been treated will carry considerable importance when consumers look to purchase the car. Kelly Blue Book offers market values for a given vehicle at up to four different condition levels. A review of studies conducted in the past will help provide confidence as to how title history and the condition of a vehicle will p influence how well a car maintains its value.

- Motivation for the Problem Undertaken

  An accurate used car price evaluation is a catalyst for the healthy development of used car market. Data mining has been applied to predict used car price in this project. This machine learning algorithm benefits different stakeholders.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Mathematical, statistical and analytics modelling done in the project are as follows:

a. Linear regression:

Equation used for linear regression model is

$$Y = bX + a$$

linear regression uses a traditional slope-intercept form, here *a* and *b* are the coefficients that we try to "learn" and produce the most accurate predictions. *X* represents our input data and *Y* is our prediction.

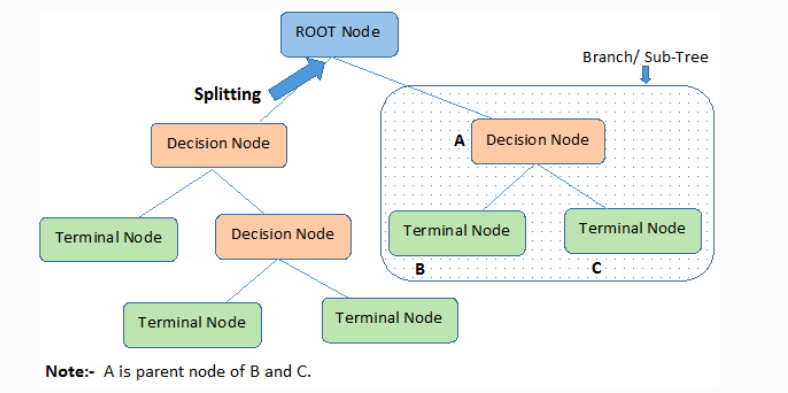
Since we have multiple variables present in the data set so the following equation was used:

$$Y(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0$$

Where x1, x2 and x3 are different independent variables and y is dependent variable. w1, w2 and w3 are similar to b and W0 is similar to a.

b. Decision Tree Regressor:

Equation used by this regressor model:



A **decision tree** is an efficient algorithm for describing a way to traverse a dataset while also defining a tree-like path to the expected outcomes. This branching in a tree is based on control statements or values, and the data points lie on either side of the **splitting node,** depending on the value of a specific feature.

1. **Information Gain**:

$$Info(D) = -\sum_{i}^{m} p_i \log_2(p_i),$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

pi is the probability that an arbitrary tuple in dataset D belongs to class Ci and is estimated by|Ci, D|/|D|**. Info(D) is simply the mean amount of information needed to identify the class/category of a data point in D.** A log function to base 2 is used, since in most cases, information is encoded in bits.

Information gain is calculated by:

$$Gain(A) = Info(D) - Info_A(D).$$

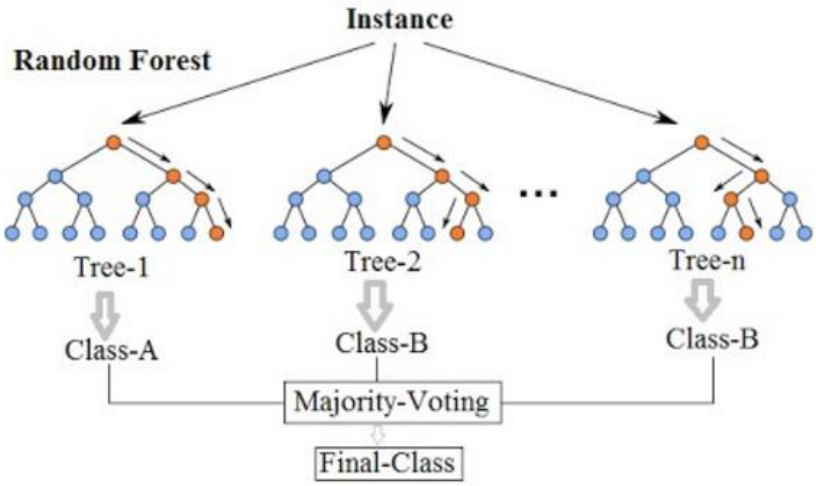
2. Gini Index:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

where pi is the probability that a tuple in D belongs to class Ci and is estimated by |Ci,D|/|D|. The sum is computed over m classes.

c. Random Forest Regressor:

Random-forest does both row sampling and column sampling with Decision tree as a base. Model h1, h2, h3, h4 are more different than by doing only bagging because of column sampling.



Random forest= DT (base learner) + bagging (Row sampling with replacement) + feature bagging (column sampling) + aggregation (mean/median, majority vote)

Steps taken to implement Random Forest regressor:

1. Suppose there are N observations and M features in the training data set. First, a sample from the training data set is taken randomly with replacement.
2. A subset of M features is selected randomly and whichever feature gives the best split is used to split the node iteratively.
3. The tree is grown to the largest.
4. The above steps are repeated and prediction is given based on the aggregation of predictions from n number of trees.

d. KNeighborsRegressor:

KNN uses a similarity metric to determine the nearest neighbors. This similarity metric is more often than not the Euclidean distance between

our unknown point and the other points in the dataset. The general formula for Euclidean distance is:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

How KN Neigbors algorithm works:
For each entry in our test set we will iterate over all the entries in the training set and calculate the Euclidean distance. Thus, we are calculating the Euclidean distances between each entry in our test set and all entries in our training set. After the Euclidean distance has been calculated, we make a 'distance' column in our training set, shuffle the resulting set and sort them in ascending order of distance. Choose the first 3(if K=3) or first 5(if K=5) entries of the sorted dataset and find the mean of their 'price' column.

e. Gradient Boosting Regressor:

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Statistical modelling done during the project:

The following statistical functions were used in the project:

1. Mean, Median, Mode and Variance.

$$\text{Mean } \bar{x} = \frac{\sum xi}{N} \qquad \text{Variance} = \frac{\sum (xi - \bar{x})^2}{N}$$

$$\text{Median} = \begin{cases} \frac{(N+1)^{th}}{2} \text{ term; when N is even} \\ \frac{N^{th}}{2} \text{ term} + \left(\frac{N}{2} + 1\right) \text{term} \\ \hline 2 \end{cases} \text{; when N is even}$$

$$\text{Mode} = \text{The value in the data set that occurs most frequently}$$

2. Standard Deviation

$$\sigma = \sqrt{\frac{\sum (X - \mu)^2}{n}}$$

where,

σ = population standard deviation
$\sum$ = sum of...
μ = population mean
n = number of scores in sample.

3. Percentile

$$P_k = k\left(\frac{n+1}{100}\right)^{th} item$$

$k = k^{th}$ *percentile*
$n =$ *no of items in given observation*

4. Z- score

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score
x = observed value
μ = mean of the sample
σ = standard deviation of the sample

5. Mean absolute error

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

$x_i = actual\ value$ $\quad$ $n = sample\ size$
$y_i = predictions$

6. Mean squared error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

- Data Sources and their formats

The Data was collected using web scrapping technique from different websites like OLX,CarDekho and Cars24.

The following data has been collected:

Brands: This column contains the name of the Brands of the cars.

Models: This column contains the name of the Models of the cars.

Variants: This column contains the Variants of different models of the cars.

Kilometres driven: This column shows the kilometres driven of the cars.

Location: This column contains the location of the seller.

Fuel Type: This column contains the fuel type of the car.

Transmission: This column contains the Transmission type of the car.

Manufacturing year: This column contains the manufacturing year of the car.

Number of owners: This column contains the number of owners the car had in past.

Price: This column contains the price of the used car. (This is our target/dependent variable)

Attaching the snapshot of the cleaned data:

```
In [4]: dataset
```

Out[4]:

	Unnamed: 0	Brand1	Model1	Variant	Kilometers driven(in kms)	Fuel type	Location1	Transmission	year of manufacturing	Number of owners	price
0	0	maruti	alto k10	vxi	27017.0	petrol	NaN	manual	2013.0	1st	285468.0
1	1	honda	city	i vtec v	14815.0	petrol	NaN	manual	2017.0	1st	814432.0
2	2	maruti	alto k10	vxi	32053.0	petrol	NaN	manual	2016.0	1st	302208.0
3	3	honda	city	i vtec cvt vx	79502.0	petrol	NaN	automatic	2015.0	1st	690772.0
4	4	ford	ecosport	1.5 ti vct mt ambiente bsiv	30953.0	petrol	NaN	manual	2018.0	1st	743820.0
...	...	...	...	...	...	...	...	...	...	...	...
8773	8773	maruti	swift	zxi plus	57148.0	petrol	2018	manual	2018.0	1st	650000.0
8774	8774	hyundai	creta	1.6 vtvt at sx plus	44088.0	petrol	2017	automatic	2017.0	1st	1135000.0
8775	8775	bmw	3 series	320d sport	39400.0	diesel	2016	automatic	2016.0	1st	2090000.0
8776	8776	tata	new safari	dicor 2.2 ex 4x2 bs iv	70000.0	diesel	2010	manual	2010.0	2nd	240000.0

All of these columns were in object format or string format, initially. So, we have to change it to the desirable formats.

```
In [5]: dataset.dtypes

Out[5]: Unnamed: 0                      int64
        Brand1                         object
        Model1                         object
        Variant                        object
        Kilometers driven(in kms)     float64
        Fuel type                      object
        Location1                      object
        Transmission                   object
        year of manufacturing         float64
        Number of owners               object
        price                         float64
        dtype: object
```

- Data Preprocessing Done

The following steps were taken in the data pre-processing phase:

1. Cleaning the data: The data which we extracted from different websites was containing lot of junk information so we removed those junk data and kept the useful information.

```
In [4]: dataset
```

Out[4]:

	Unnamed: 0	Brand1	Model1	Variant	Kilometers driven(in kms)	Fuel type	Location1	Transmission	year of manufacturing	Number of owners	price
0	0	maruti	alto k10	vxi	27017.0	petrol	NaN	manual	2013.0	1st	285468.0
1	1	honda	city	i vtec v	14815.0	petrol	NaN	manual	2017.0	1st	814432.0
2	2	maruti	alto k10	vxi	32053.0	petrol	NaN	manual	2016.0	1st	302208.0
3	3	honda	city	i vtec cvt vx	79502.0	petrol	NaN	automatic	2015.0	1st	690772.0
4	4	ford	ecosport	1.5 ti vct mt ambiente bsiv	30953.0	petrol	NaN	manual	2018.0	1st	743820.0
...	...	...	...	...	...	...	...	...	...	...	...
8773	8773	maruti	swift	zxi plus	57148.0	petrol	2018	manual	2018.0	1st	650000.0
8774	8774	hyundai	creta	1.6 vtvt at sx plus	44088.0	petrol	2017	automatic	2017.0	1st	1135000.0
8775	8775	bmw	3 series	320d sport	39400.0	diesel	2016	automatic	2016.0	1st	2090000.0
8776	8776	tata	new safari	dicor 2.2 ex 4x2 bs iv	70000.0	diesel	2010	manual	2010.0	2nd	240000.0

2. Standardization of data: After cleaning the data we observed that some columns contain same information but either in upper case or lower case. The Price column had two values one with numeric values and other with string (e.g., 870050 and 5 lakhs). So, we used different methods to bring the data in desirable formats.
3. Handling the missing values: Removing and handling the null values in the data set using appropriate algorithms.

```
In [15]: #checking the null values present in each feature
         dataset.isnull().sum()
```

```
Out[15]: Brand1                      30
         Model1                      32
         Variant                    127
         Kilometers driven(in kms)   30
         Fuel type                   30
         Location1                  783
         Transmission               249
         year of manufacturing      197
         Number of owners            47
         price                       42
         dtype: int64
```

Observations:Here we can see that there are lot of null values present in Location1,Transmission ,year of manufacturing and variant.Almost each attribute has some null values,we we'll be removing them in further steps.

```
In [16]: #lets see the percentage of null values present in the dataset.
         percent_missing = dataset.isnull().sum() * 100 / len(dataset)
         percent_missing
```

```
Out[16]: Brand1                      0.341763
         Model1                      0.364548
         Variant                     1.446799
         Kilometers driven(in kms)   0.341763
         Fuel type                   0.341763
         Location1                   8.920027
         Transmission                2.836637
         year of manufacturing       2.244247
         Number of owners            0.535429
         price                       0.478469
         dtype: float64
```

Since variant had 1.44% of null values and we do not have any appropriate method to fill those null values so we decided to drop all these rows having null values.

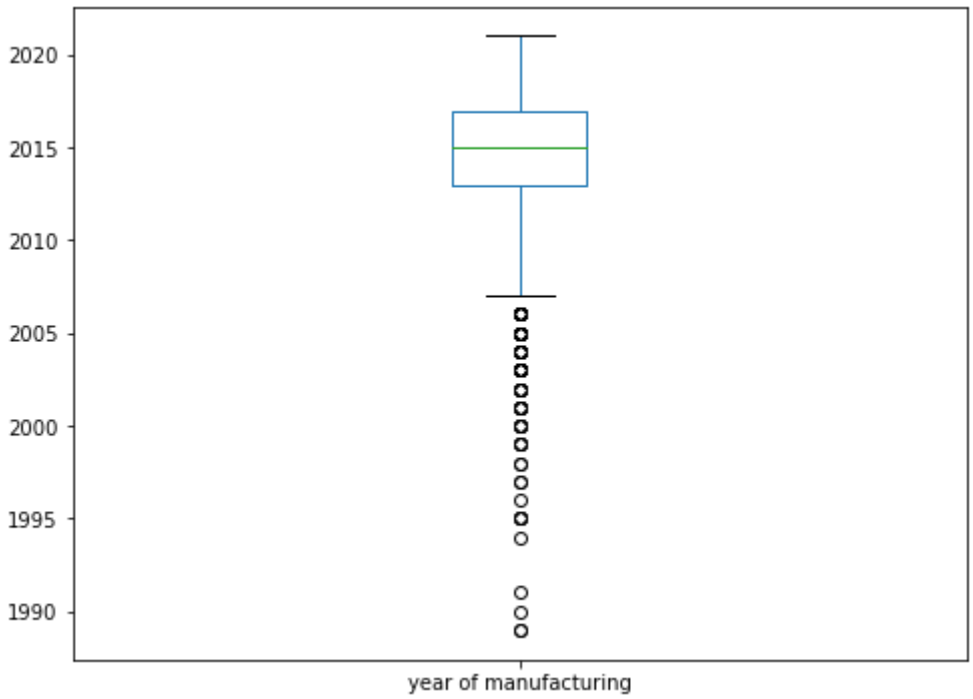
```
In [19]: #droping the null value from the attributes
         #dropping null values from variant
         dataset = dataset.dropna(subset =['Variant'])

In [30]: #using replace function to replace the nan values with mode of the attribute
         dataset["Model1"] = dataset["Model1"].replace(np.nan , 'swift')

In [36]: #using replace function to replace the nan values with mode of the attribute
         dataset["Transmission"] = dataset["Transmission"].replace(np.nan , 'manual')

In [34]: #using fillna funtion to replace null values with median of the attribute
         dataset["year of manufacturing"] = dataset["year of manufacturing"].fillna(dataset["year of manufacturing"].median())

In [38]: #using replace function to replace the nan values with mode of the attribute
         dataset["Number of owners"] = dataset["Number of owners"].replace(np.nan , '1st')

In [40]: #using fillna funtion to replace null values with median of the attribute
         dataset["price"] = dataset["price"].fillna(dataset["price"].median())

In [42]: #using fillna funtion to replace null values with median of the attribute
         dataset["Location1"] = dataset["Location1"].replace(np.nan , 'new delhi')
```

Using the above technique, we replaced / removed the null values from the data set.

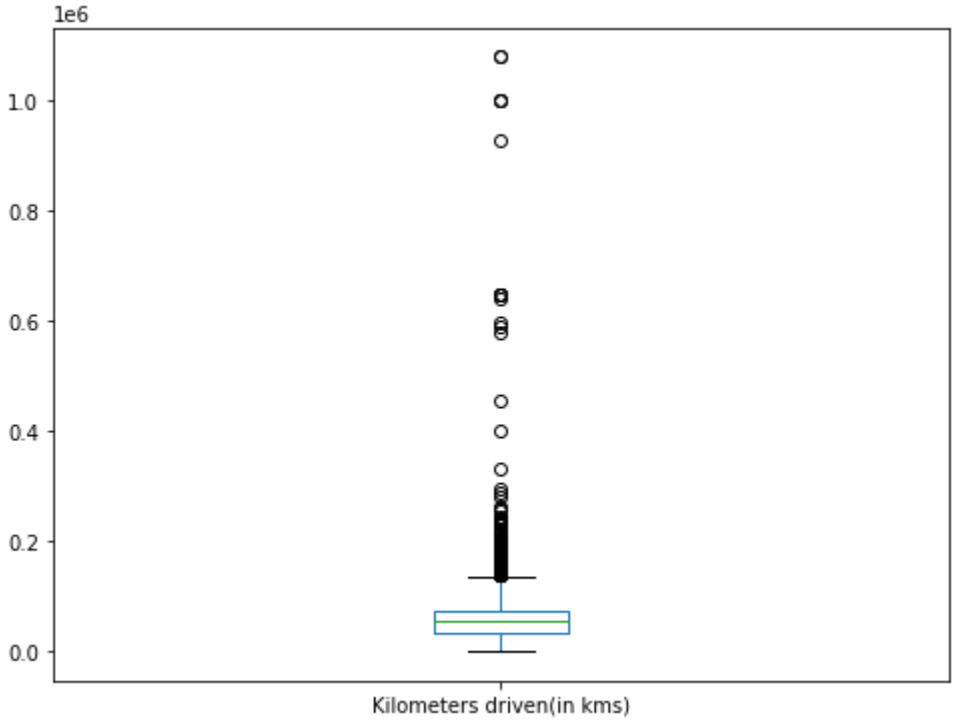
```
In [43]: dataset.isnull().sum()

Out[43]: Brand1                      0
         Model1                      0
         Variant                     0
         Kilometers driven(in kms)   0
         Fuel type                   0
         Location1                   0
         Transmission                0
         year of manufacturing       0
         Number of owners            0
         price                       0
         dtype: int64
```

So, we successfully replaced and removed the null values from the dataset

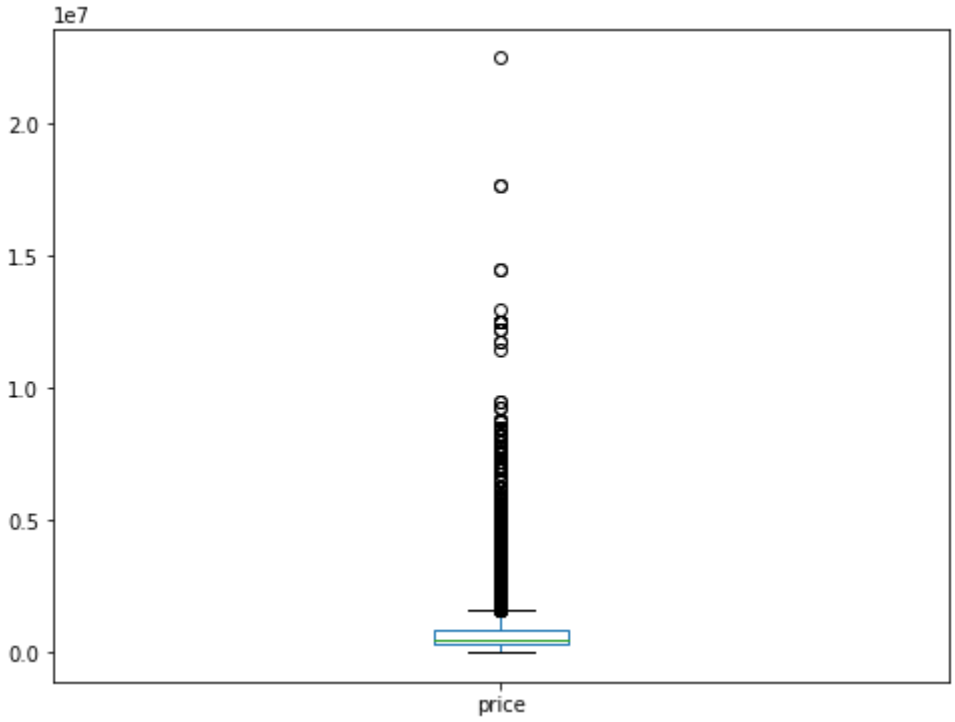
4. Removing Outliers from the dataset: So, first we checked if outliers are present in the data set or not. Since we only have 3 numerical features so we'll perform this process on these 3 features using box plot.



Observations: most of the outliers are present at the lower whisker.



Observations: Most of the outliers are present near the upper whisker.



Observations: Here we saw that lots of outliers present near the upper whisker of the boxplot. Since, its our target variable we'll not remove any outlier from it.

We observed that lots of outliers were present in the data set. So, we removed the outliers using Z Score function. But before that we encoded the categorical features using ordinal encoder.

```
#Encoding the data and then we'll be removing outliers

#using ordinal encoder and label encoder to encode categorical features
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
for i in dataset_categorical:
    dataset[i] = oe.fit_transform(dataset[[i]])
```

```
#removing outliers using z score

dataset_outliers_remove = dataset[['year of manufacturing','Kilometers driven(in kms)']]


from scipy.stats import zscore

z = np.abs(zscore(dataset_outliers_remove))
threshold = 3
print(np.where(z>3))
```

```
In [146]: for i in dataset_outliers_remove:
              dataset[i] = dataset_outliers_remove[i]

          dataset_new = dataset[(z<3).all(axis = 1)]

In [147]: dataset_new.shape

Out[147]: (8544, 10)

In [148]: data_lose = ((dataset.shape[0] - dataset_new.shape[0])/dataset.shape[0])*100
          data_lose

Out[148]: 1.2368512310715525
```

Total data lost after removing the outliers is 1.24 percent.

After removing the outliers, we observed that 1.24% percent of data was lost, which is acceptable.

Now, we'll be removing skewness from the data set for that we first split the data into dependent and independent variable.

```
#let's split the data into dependent and independent variables
x = dataset_new.drop(['price'],axis  = 1)
y = dataset_new['price']
```

```
#checking for the skewness present in the features
x.skew()
```

```
Brand1                       0.387187
Model1                       0.112119
Variant                     -0.416705
Kilometers driven(in kms)    0.864676
Fuel type                   -0.345112
Location1                   -2.197258
Transmission                -1.102452
year of manufacturing       -0.411459
Number of owners             2.916922
dtype: float64
```

Then we remove the skewness from the features having skewness more than 0.5 and less than -0.5 by using power transformer function.

```
#We will remove the skewness from the numerical features only and will try to keep the keep the skewness between -0.5 and +0.5
#power transformer to remove skewness
from sklearn.preprocessing import PowerTransformer
power = PowerTransformer()

x['Kilometers driven(in kms)'] = power.fit_transform(x[['Kilometers driven(in kms)']])
```

```
x.skew()
```

```
Brand1                     0.387187
Model1                     0.112119
Variant                   -0.416705
Kilometers driven(in kms) -0.041842
Fuel type                 -0.345112
Location1                 -2.197258
Transmission              -1.102452
year of manufacturing     -0.411459
Number of owners           2.916922
dtype: float64
```

After removing the skewness, we scaled our dataset before start building our model.

5. Scaling the data: In this step we used scaling feature to standardize our data before training our model.

```
dataset_numerical = x[['Kilometers driven(in kms)', 'year of manufacturing']]
```

```
#scaling the data
#Using standard scaler to scale the data.
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
for i in dataset_numerical:
    x[i]= sc.fit_transform(x[[i]])
```

```
x.head()
```

	Brand1	Model1	Variant	Kilometers driven(in kms)	Fuel type	Location1	Transmission	year of manufacturing	Number of owners
0	21.0	37.0	1937.0	-0.827806	5.0	95.0	2.0	-0.583298	1.0
1	11.0	89.0	1417.0	-1.398193	5.0	95.0	2.0	0.711758	1.0
2	21.0	37.0	1937.0	-0.628613	5.0	95.0	2.0	0.387994	1.0
3	11.0	89.0	1413.0	0.803445	5.0	95.0	1.0	0.064230	1.0
4	9.0	128.0	309.0	-0.670822	5.0	95.0	2.0	1.035522	1.0

```
x.shape
```

```
(8544, 9)
```

For scaling the dataset we used imported standardscaler function from sklearn.preprocessing library.

After scaling the data, we can now build our model.

- Data Inputs- Logic- Output Relationships

To see the relationship of Input variables with the output(dependent) variables we used correlation matrix table and we got the following results:

```
In [132]: #lets use the correlation matrix to see the correlation between the features
          corr_matrix =dataset_encoder.corr()
          corr_matrix
```

Out[132]:

	Brand1	Model1	Variant	Kilometers driven(in kms)	Fuel type	Location1	Transmission	year of manufacturing	Number of owners	price
Brand1	1.000000	0.177999	0.044744	0.078851	-0.086106	-0.038263	0.095331	-0.005866	-0.013667	-0.064052
Model1	0.177999	1.000000	0.271469	0.041758	-0.126519	-0.065301	0.096370	-0.034395	-0.004508	-0.051626
Variant	0.044744	0.271469	1.000000	-0.053186	0.025222	0.029943	0.118497	0.011018	-0.037295	-0.060704
Kilometers driven(in kms)	0.078851	0.041758	-0.053186	1.000000	-0.258812	-0.055459	0.103526	-0.372458	0.146285	-0.154612
Fuel type	-0.086106	-0.126519	0.025222	-0.258812	1.000000	0.011512	0.041944	-0.006354	-0.014101	-0.152760
Location1	-0.038263	-0.065301	0.029943	-0.055459	0.011512	1.000000	0.063943	0.206040	-0.150591	0.029503
Transmission	0.095331	0.096370	0.118497	0.103526	0.041944	0.063943	1.000000	-0.150100	-0.016338	-0.476349
year of manufacturing	-0.005866	-0.034395	0.011018	-0.372458	-0.006354	0.206040	-0.150100	1.000000	-0.337366	0.317753
Number of owners	-0.013667	-0.004508	-0.037295	0.146285	-0.014101	-0.150591	-0.016338	-0.337366	1.000000	-0.066635
price	-0.064052	-0.051626	-0.060704	-0.154612	-0.152760	0.029503	-0.476349	0.317753	-0.066635	1.000000

Using the above code, we got the correlation of all the features with each other. So, to get the correlation of input variables with output variable we used the following code and got the following result:

```
In [137]: corr_matrix['price'].sort_values(ascending = False)

Out[137]: price                          1.000000
          year of manufacturing          0.317753
          Location1                      0.029503
          Model1                        -0.051626
          Variant                       -0.060704
          Brand1                        -0.064052
          Number of owners              -0.066635
          Fuel type                     -0.152760
          Kilometers driven(in kms)     -0.154612
          Transmission                  -0.476349
          Name: price, dtype: float64
```

After analysing the output, we concluded that year of manufacturing shows positive relationship with the output variable and transmission have negative impact on it.

The following observations were also made:

1. fuel type and kilometre driven are negatively correlated to each other, which means if one increase other decrease and vice versa.
2. Same with year of manufacturing and kilometres driven.
3. transmission and price are also negatively correlated to each other.
4. whereas, year of manufacturing and price are positively co related which means if one increases other also increases and vice versa.
5. variant and model are also positively correlated to each other.
6. Most of the features do not show much correlation with each other.
7. As we saw earlier in heatmap that year of manufacturing is positively correlated to price and transmission is negatively correlated to the car price

- ## Hardware and Software Requirements and Tools Used

Hardware requirement:

Type of hardware	Hardware requirements
Hardware	Dual core Intel Pentium compatible processor or multiprocessor-based computer with a 2Ghz or greater processor • 64-bit system • Network interface card
Memory	4 GB or more

Software requirements:

Type of software	software requirements
Operating System	Windows 10
Web browser	chrome
Text Editor	Jupyter notebook

Also, we used following libraries and packages in the project:

In [1]:
```
import numpy as np
import pandas as pd
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

a.) NumPy- for scientific calculation in Python.
b.) Pandas- for data manipulation and analysis.
c.) Matplotlib- used to plot charts in Python.
d.) Sklearn – it provides efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

We first use the basic statistical methods, i.e., calculating the Mean, Median, Mode, Standard deviation, Variance, Minimum and Maximum value and Percentiles of the data set and made analysis on these factors.

To get the basic information of the dataset we used function like dataset.info ()

To check the null values present in the feature we used dataset.isnull().sum() function.

To get the datatypes of the features we used dataset.dtype.

After statistical analysis and getting basic information of the dataset We saw there were lot of uncertainties present in the data set so to remove them we used following methods.

1.) **Z score**: to remove the outliers from the data set.

2.) **Power transformation**: to remove skewness from the data set.

To check the relationship among the feature we used different visualisation technique. We got lots of insights from these plots which helped us in choosing our features effectively.

After that we scaled our data so that our model does not get biased for a particular feature. After that we gave the data for training and testing.

- Testing of Identified Approaches (Algorithms)

  First, we split the data,

  Data Splitting:

  We used 80-20% train-test split of data.

  After that we used following Algorithms for testing and training:

  1. Linear Regression
  2. Decision Tree Regressor.
  3. Random Forest Regressor
  4. KNeighbors Regressor
  5. Gradient Boosting Regressor

- Run and evaluate selected models

1. **Linear Regressor**:

   Linear regression model assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y

can be calculated from a linear combination of the input variables (x). When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

Input given to the model and output obtained:

In [188]:
```
#first using Linear regression to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absolute
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
prdlr = lr.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prdlr))
print("Mean Squared error :",mean_squared_error(y_test,prdlr))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prdlr))
```

```
Accuracy of the model : 0.42815608992500553
Mean Squared error : 559365572603.6576
Mean Absolute eroor : 428528.17495832127
```

Observations : As we can see that the accuracy of the Linear regression model is 42.82% which is very low.And the mean absolute error is 428528, which is quite high.it means that our model is not learning efficiently.

Observations: As we can see that the accuracy of the Linear regression model is 42.82% which is very low. And the mean absolute error is 428528, which is quite high.it means that our model is not learning efficiently.

2. **Decision Tree Regressor**:

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

Input given to the model and output obtained:

In [189]:
```
#first using DecisionTreeRegressor to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absol
from sklearn.tree import DecisionTreeRegressor
dtc = DecisionTreeRegressor()
dtc.fit(x_train,y_train)
prdtc = dtc.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prdtc))
print("Mean Squared error :",mean_squared_error(y_test,prdtc))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prdtc))
```

```
Accuracy of the model : 0.7562356816371416
Mean Squared error : 238445081112.25403
Mean Absolute eroor : 145832.79481178077
```

Observations : As we can see that the accuracy of the Decision tree regression model is 75.62% which is satisfactory.And the mean absolute error is 145832, which is low as compared to previous model.Also, our model is learning quite efficiently.

Observations: As we can see that the accuracy of the Decision tree regression model is 75.62% which is satisfactory. And the mean absolute error is 145832, which is low as compared to previous model. Also, our model is learning quite efficiently.

3. **Random Forest Regressor:**

Random forest is a bagging technique and not a boosting technique. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees.

It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Input given to the model and output obtained:

In [190]:
```
#first using RandomForestRegressor to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absol
from sklearn.ensemble import RandomForestRegressor
rfc = RandomForestRegressor()
rfc.fit(x_train,y_train)
prrfc = rfc.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prrfc))
print("Mean Squared error :",mean_squared_error(y_test,prrfc))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prrfc))
```

```
Accuracy of the model : 0.8876597033250474
Mean Squared error : 109888893225.79056
Mean Absolute eroor : 131045.09236924963
```

Observations : As we can see that the accuracy of the Random forest regression model is 88.77% which is a good accuracy.And the mean absolute error is 131045, which is lowest as compared to previous models.Also, our model is learning effectively.

Observations: As we can see that the accuracy of the Random Forest regression model is 88.77% which is a good accuracy. And the mean absolute error is 131045, which is lowest as compared to previous models. Also, our model is learning effectively.

4. KNNeighbors Regressor:

The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

Input given to the model and output obtained:

In [191]:
```
#first using KNeighborsRegressor to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absolut
from sklearn.neighbors import KNeighborsRegressor
KNN = KNeighborsRegressor(n_neighbors=5)
KNN.fit(x_train,y_train)
prKNN = KNN.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prKNN))
print("Mean Squared error :",mean_squared_error(y_test,prKNN))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prKNN))
```

```
Accuracy of the model : 0.6200126490617288
Mean Squared error : 371695559565.9975
Mean Absolute eroor : 241956.0168519602
```

Observations : As we can see that the accuracy of the KNeighbors regression model is 62.00% which is quite low.And the mean absolute error is 241956, which is high as compared to previous models.Also, our model is not learning efficiently.

Observations: As we can see that the accuracy of the KNeighbours regression model is 62.00% which is quite low. And the mean absolute error is 241956, which is high as compared to previous models. Also, our model is not learning efficiently.

5. Gradient Boosting Regressor:

"Boosting" in machine learning is a way of combining multiple simple models into a single composite model. This is also why boosting is known as an additive model, since simple models (also known as weak learners) are added one at a time, while keeping existing trees in the model unchanged. As we combine more and more simple models, the complete final model becomes a stronger predictor. The term "gradient" in "gradient boosting" comes from the fact that the algorithm uses gradient descent to minimize the loss.

Input given to the model and output obtained:

In [192]:
```
#using boosting technique to build model
from sklearn.ensemble import GradientBoostingRegressor
gbc = GradientBoostingRegressor()
gbc.fit(x_train,y_train)
prdgbc = gbc.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prdgbc))
print("Mean Squared error :",mean_squared_error(y_test,prdgbc))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prdgbc))
```

```
Accuracy of the model : 0.8211102810372415
Mean Squared error : 174986125265.17557
Mean Absolute eroor : 220802.10555285713
```

Observations : As we can see that the accuracy of the Gradient boosting regression model is 82.11% which is a good accuracy.And the mean absolute error is 220802, which is quite high as compared to previous models.Also, our model is also learning effectively.

Observations: As we can see that the accuracy of the Gradient boosting regression model is 82.11% which is a good accuracy. And the mean absolute error is 220802, which is quite high as compared to previous models. Also, our model is also learning effectively.

We concluded the result on three metrices that are:

R2_score, **Mean Absolute error, Mean squared error**.

- Key Metrics for success in solving problem under consideration

  The four main metrices used to predict accuracy that are, r2_score, Mean Absolute error, mean squared error and cross validation score.

1. Mean Squared Error:

   We used mean squared error as the metric because the mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. It helps in selecting the model with least mean squared error.

2. Mean absolute Error:

   We used mean absolute error because mean absolute error is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of

measurement.it tells us the deviation between the actual values and the predicted values

3. R2 Score:

   We used r2 score or the coefficient of determination, denoted R^2 or r^2 a is the proportion of the variance in the dependent variable that is predictable from the independent variable. This tells the actual accuracy of the model.

4. Cross validation score:

   We used cross validation score so that we can check whether there is biasness present in the model or not.

Cross validation score of different models are as follows:

In [194]:
```
#creating a function that will return the firrence of the cross validation score and accuracy of the model
def model_accuracy(accuracy,cross_validation_score):
    difference = accuracy - cross_validation_score
    print("The difference between the accuracy and cross validation score is : ",difference*100)
```

In [195]:
```
from sklearn.model_selection import cross_val_score

#Linear Regressor
cross_val1 = cross_val_score(LinearRegression(),x,y,cv = 5)
print("Cross Validation score for Linear regression : ",cross_val1.mean())
```

```
Cross Validation score for Linear regression :  0.1973329108925334
```

In [196]:
```
model_accuracy(r2_score(y_test,prdlr),cross_val1.mean())
```

```
The difference between the accuracy and cross validation score is :  23.082317903247212
```

In [197]:
```
#calculating cross validation score of Random forest regressor

cross_val2 = cross_val_score(RandomForestRegressor(),x,y,cv = 5)
print("Cross Validation Score for Random Forest Regressor",cross_val2.mean())
```

```
Cross Validation Score for Random Forest Regressor 0.7521595260921684
```

In [198]:
```
model_accuracy(r2_score(y_test,prrfc),cross_val2.mean())
```

```
The difference between the accuracy and cross validation score is :  13.5500177232879
```

In [199]:
```
#calculating cross validation score of Decision Tree regressor

cross_val3 = cross_val_score(DecisionTreeRegressor(),x,y,cv = 5)
print("Cross Validation Score for Decision tree regressor",cross_val3.mean())
```

```
Cross Validation Score for Decision tree regressor 0.5374107935729794
```

In [200]:
```
model_accuracy(r2_score(y_test,prdtc),cross_val3.mean())
```

```
The difference between the accuracy and cross validation score is :  21.882488806416223
```

In [201]:
```
#calculating cross validation score of KNeighborsRegressor regressor

cross_val4 = cross_val_score(KNeighborsRegressor(),x,y,cv = 5)
print("Cross Validation Score K Neighbors ",np.abs(cross_val4).mean())
```

```
Cross Validation Score K Neighbors  0.36135324572367394
```

In [202]:
```
model_accuracy(r2_score(y_test,prKNN),cross_val4.mean())
```

```
The difference between the accuracy and cross validation score is :  35.49546878051705
```

In [203]:
```
#calculating cross validation score of Gradien boosting regressor

cross_val5 = cross_val_score(gbc,x,y,cv = 5)
print("Cross validation score of Gradient Boosting classifier is ",cross_val5.mean())
```

```
Cross validation score of Gradient Boosting classifier is  0.6771914253003555
```

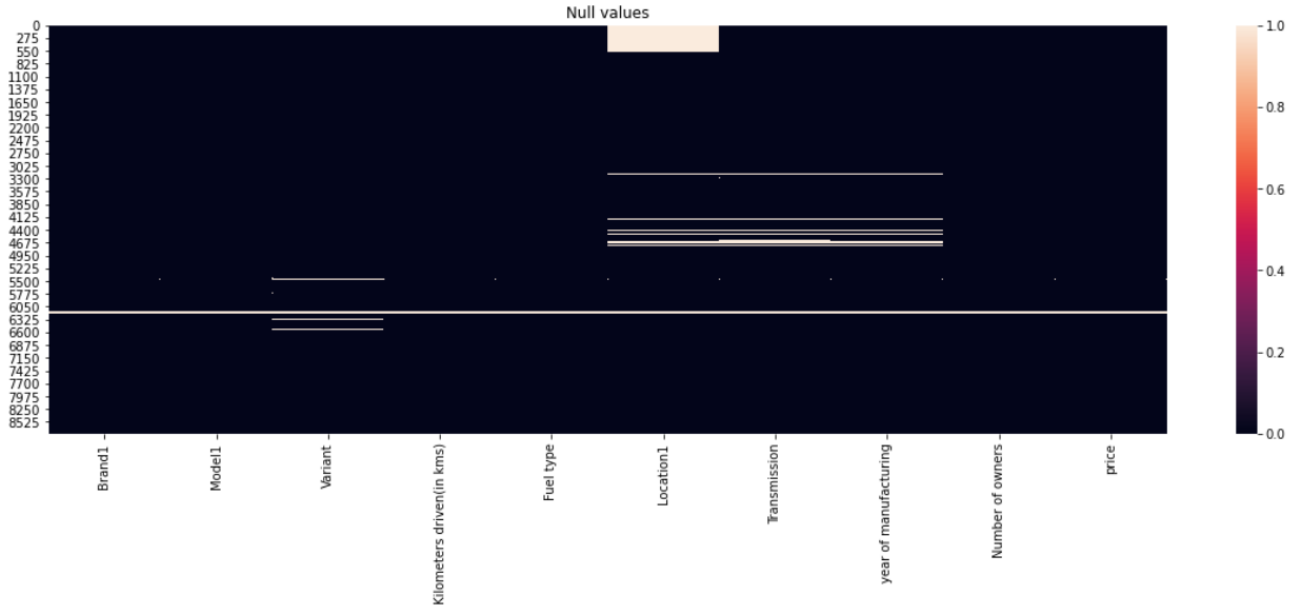
In [204]:
```
model_accuracy(r2_score(y_test,prdgbc),cross_val5.mean())
```

```
The difference between the accuracy and cross validation score is :  14.3918855736886
```

- Visualizations

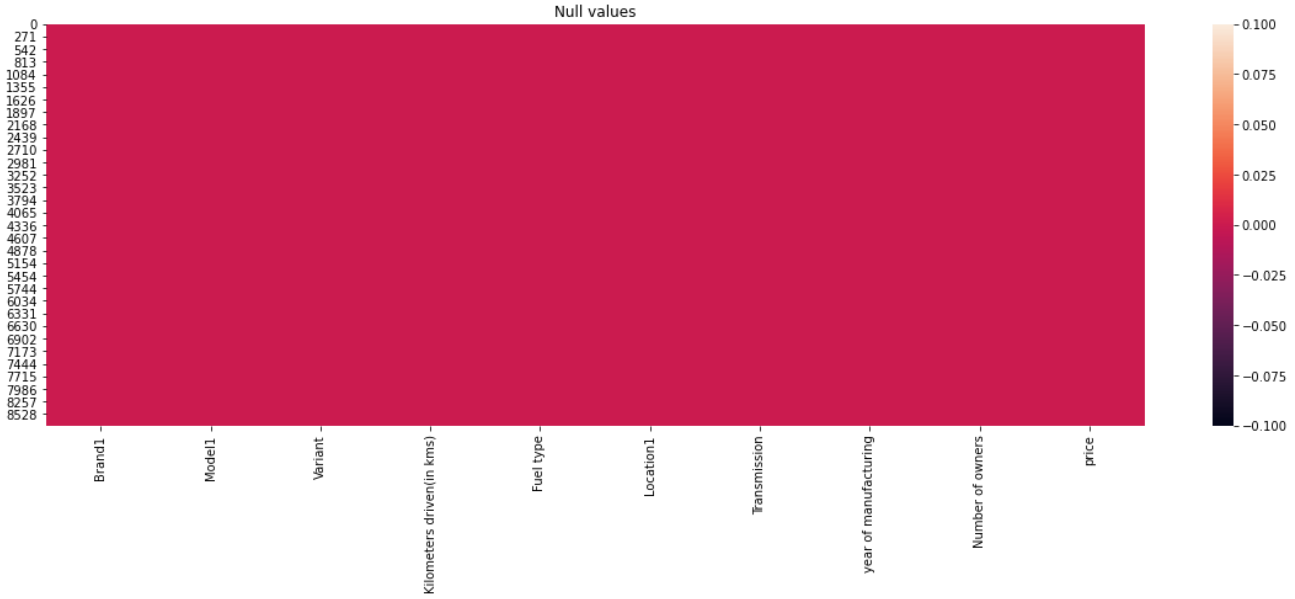
Visualizations and observations

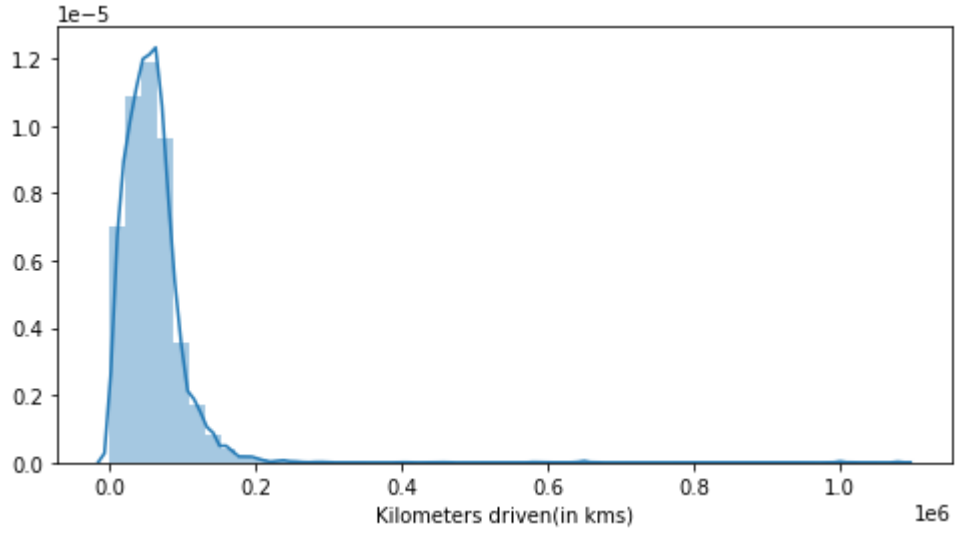
In [17]:
```
#we'll be using heatmap to see the null values
plt.figure(figsize = [20,6])
sns.heatmap(dataset.isnull())
plt.title('Null values')
plt.show()
```

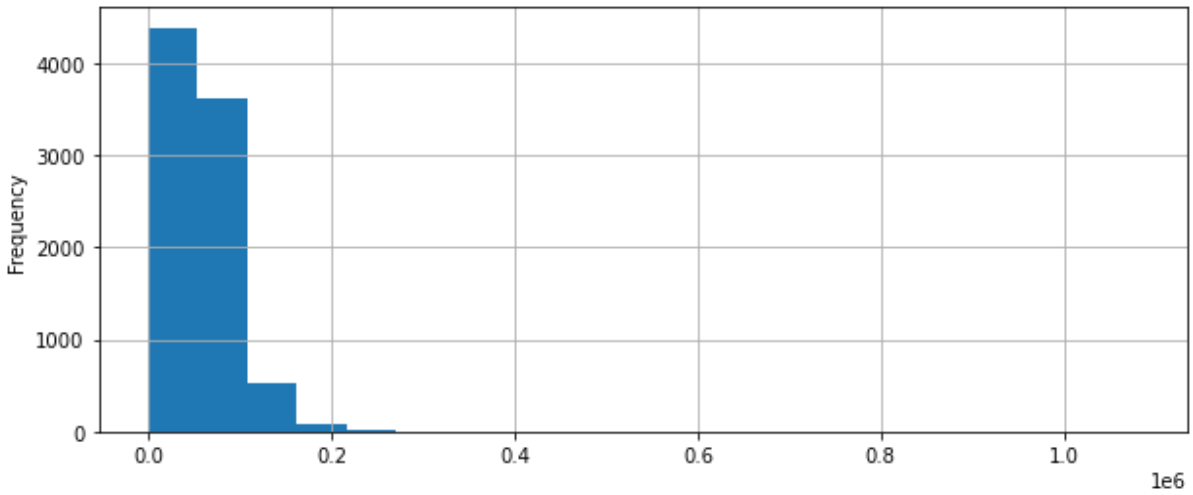


Observations: from here we can see that the location1 contains the greatest number of null values in the beginning of the dataset.

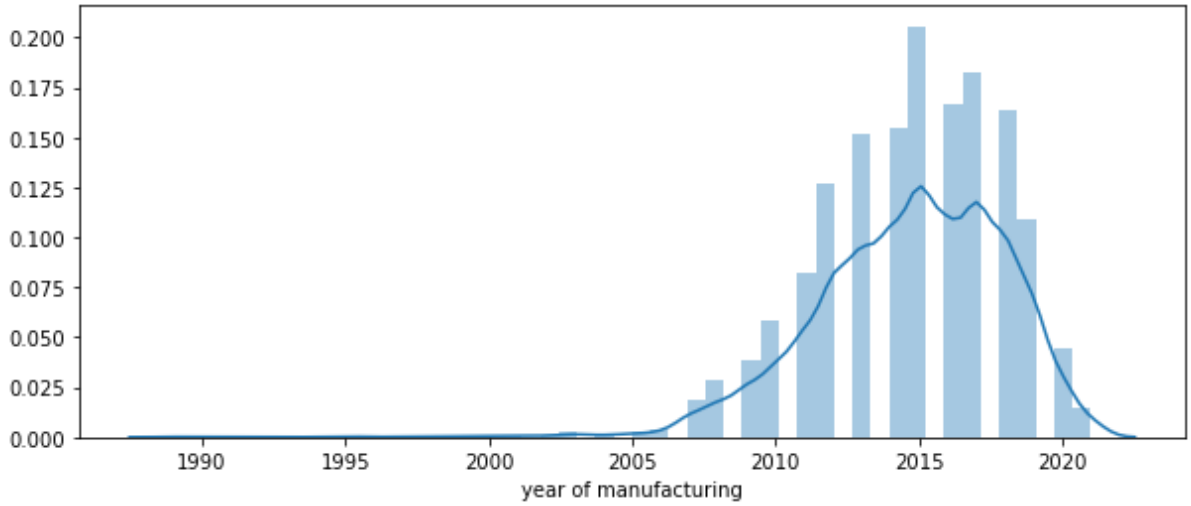
After Removing null values we obtained the following heatmap.

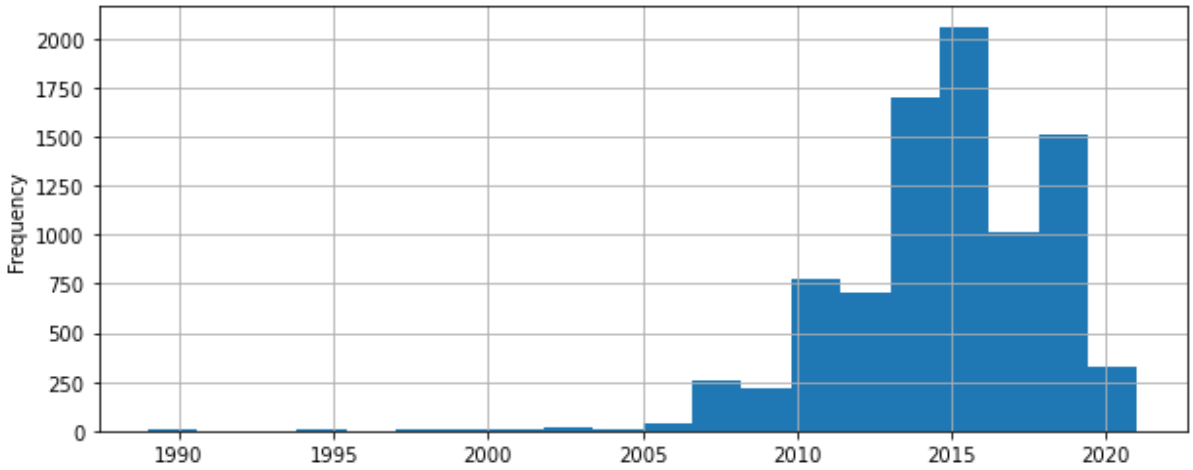




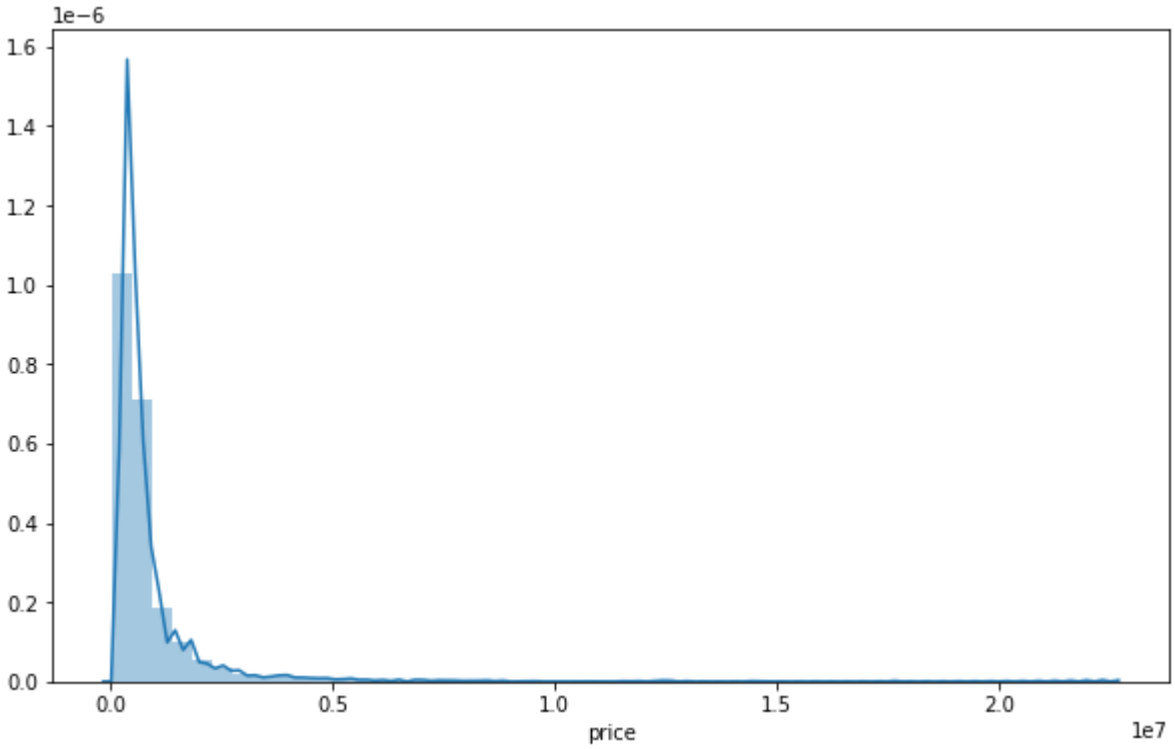


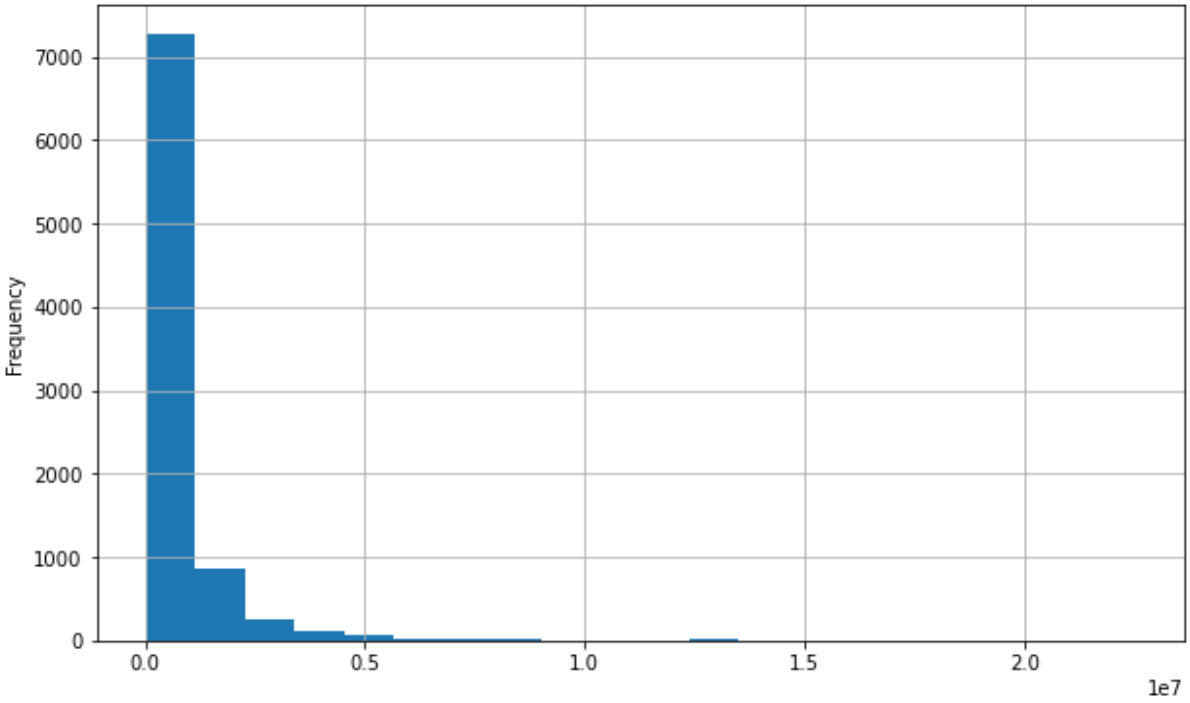
Observations: Here we observed that more than 4000 cars have driven kilometres more than 50 thousand.



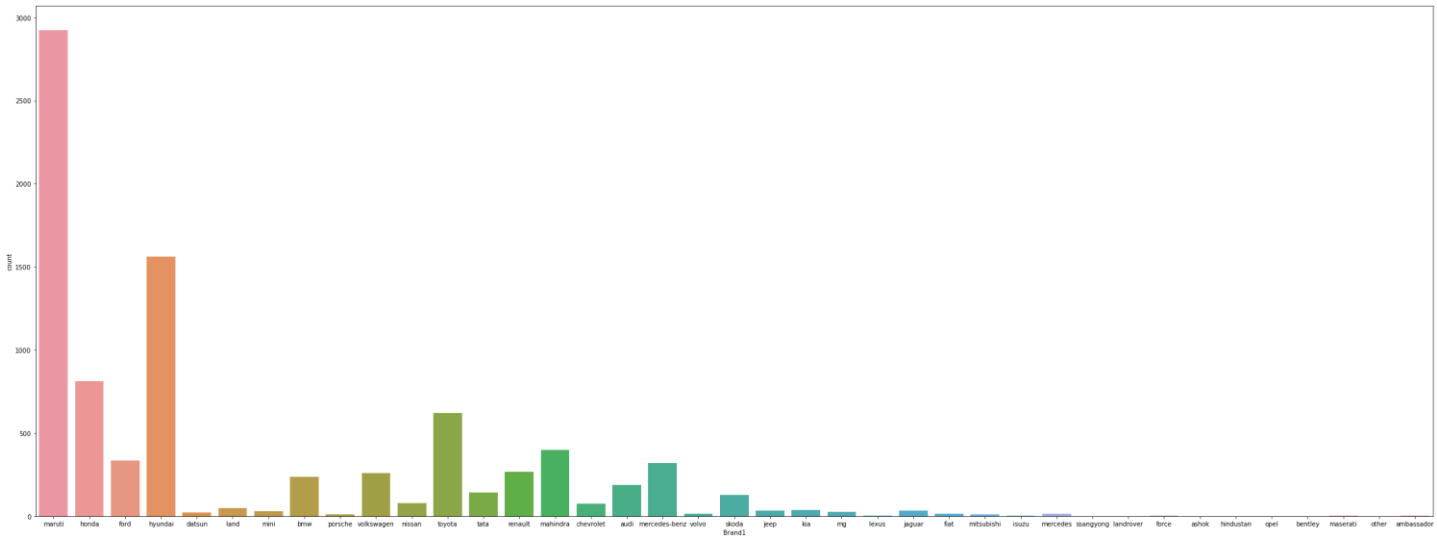


Observation: Here we observed that most the of the cars were manufactured between year 2014 and 2018 i.e., more than 80% of the total cars.

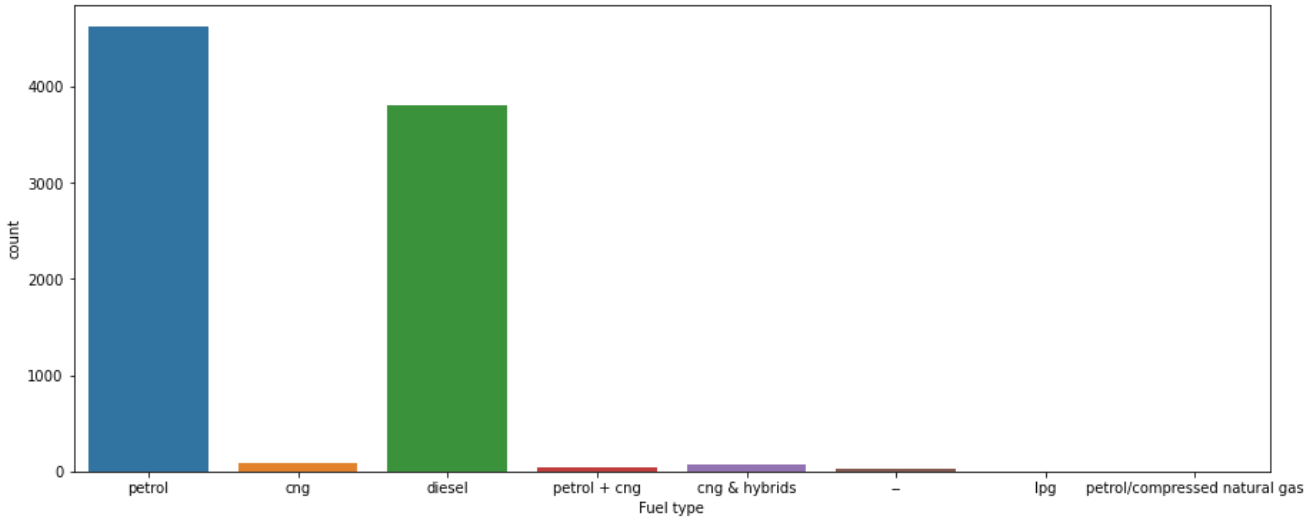




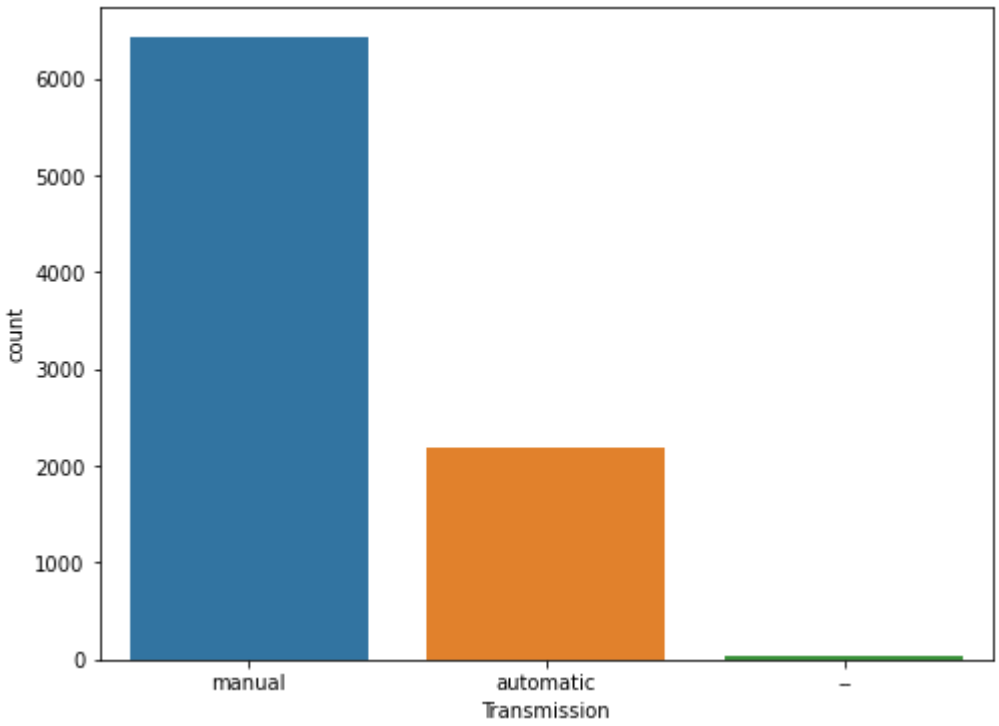
Observations: More than 7000 cars have a price range of 4.95 lakhs to 8 lakhs.



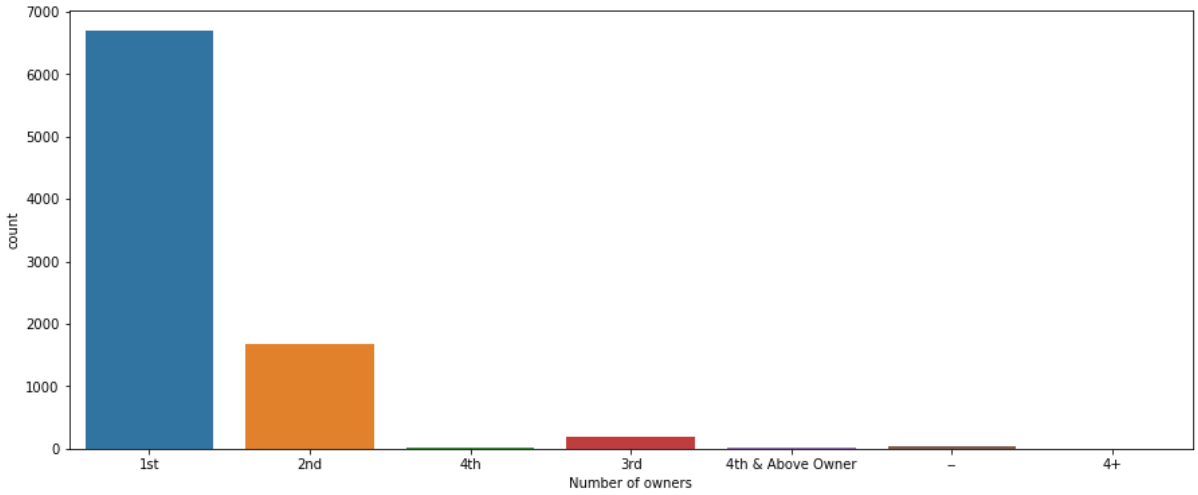
Observations: Brand attribute contains 39 different brands where the greatest number of cars has brand named Maruti in the dataset, followed by Hyundai and Honda.



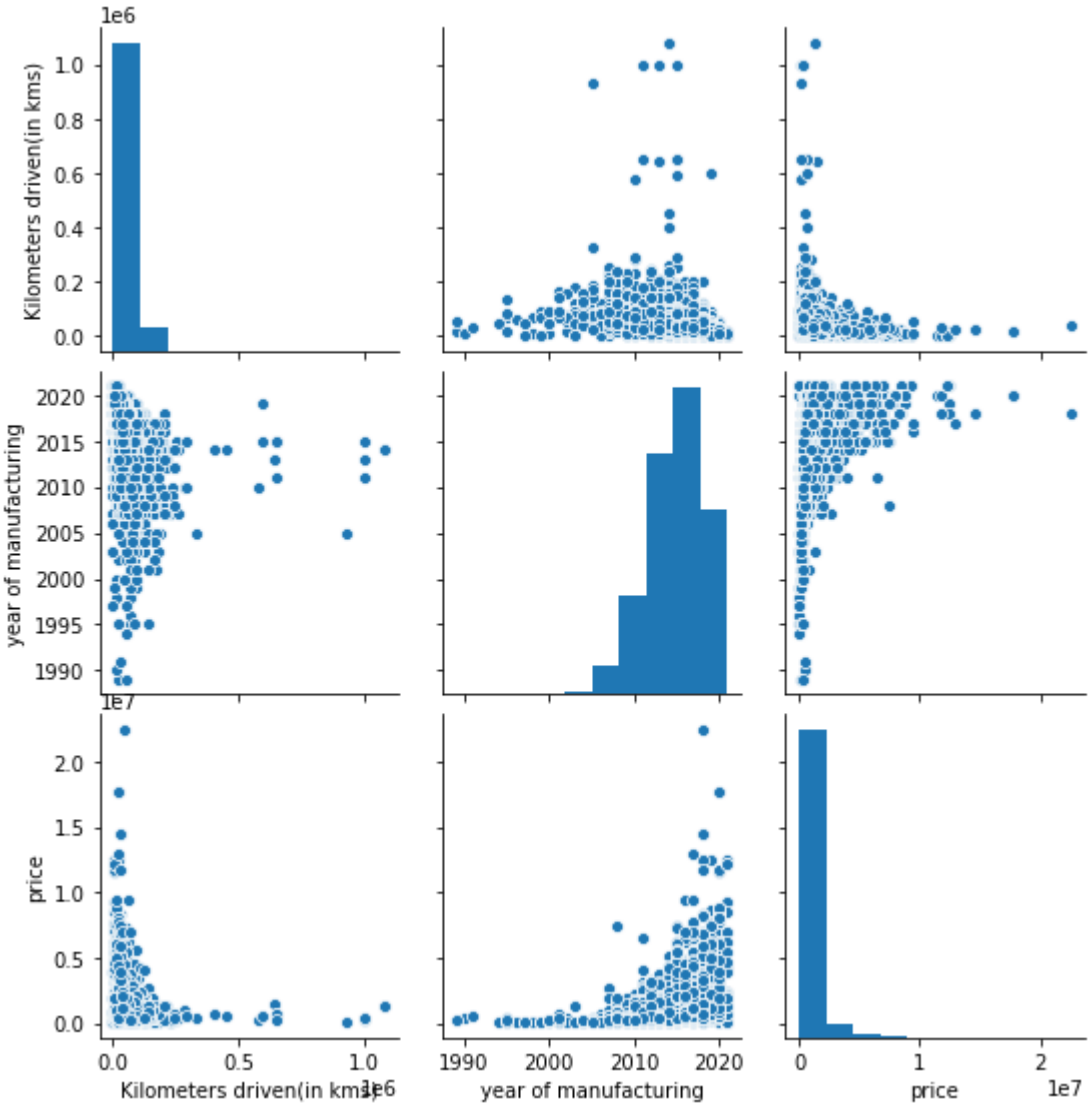
Observations: Fuel type attribute contains 8 different fuels type out of which one is "--" we'll replace it in further steps. Here the greatest number of cars petrol fuel type followed by diesel.



Observations: Transmission attribute contains 3 different type of transmission.it also contains "--" which we'll treat as null values and replace it. most number of cars has transmission type as Manual.

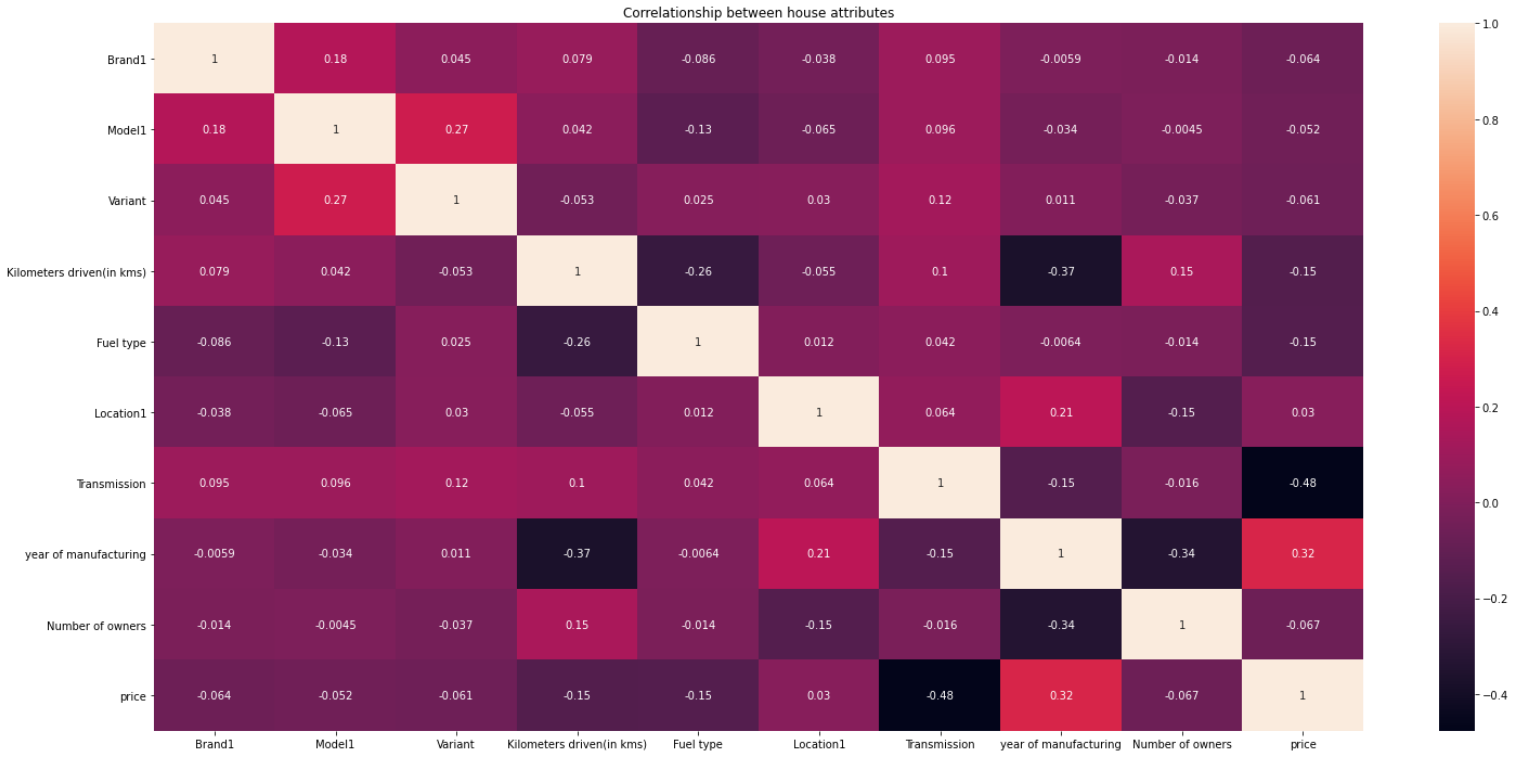


Observations: Number of owners attribute contains 7 different categories.it also contains "--" which we'll treat as null values and replace it. Here most of the cars belongs to 1st owner category in the dataset.



Observations:

1. Here we observed that as the manufacturing year increases means newer the car more is the price of the car.
2. as the kilometre driven increases price of the car decreases.



Observations: Here we observed the following things:

1. fuel type and kilometre driven are negatively correlated to each other, which means if one increase other decrease and vice versa. 2.same with year of manufacturing and kilometres driven.
2. transmission and price are also negatively correlated to each other.
3. whereas, year of manufacturing and price are positively co related which means if one increases other also increases and vice versa.
4. variant and model are also positively correlated to each other.
5. Most of the features do not show much correlation with each other.

- ## Interpretation of the Results

  The interpretation made was that there were null values present in each attribute of the data set. As we saw earlier in heatmap that year of manufacturing is positively correlated to price and transmission is negatively correlated to the car price.

  The final model selected:

  After checking the accuracy of all the models, we observed that Random Forest regressor model shows Maximum accuracy i.e.,

around 88 percent. The next step is to check the cross-validation score of these models in order to check whether there is any biasness in the model or not.

After comparing the cross-validation scores of the models we saw that Random Forest Regressor model shows least difference and KNeighbors regressor model shows the maximum difference. But we saw earlier that Random Forest regressor model has the highest accuracy and has least Mean Absolute error compared to another model. Also, the difference between the cross-validation score and accuracy of this model is lowest as compared to other models. So, we'll be using Random Forest regressor model as our final model.

Hyperparameter tuning of Random Forest Regressor

In [221]:
```
from sklearn.model_selection import GridSearchCV
#creating parameter list to pass in GridSearchCV
parameters = {'bootstrap': [True, False],
              'n_estimators': [1,100],
             'criterion': ['squared_error','mse','absolute_error','mae'],
             'min_samples_leaf': [1, 2, 4],
             'min_samples_split' : [1,10],
             'max_depth' : [1,10],
             'max_features' : ['auto', 'sqrt', 'log2']}
```

In [222]:
```
GVC = GridSearchCV(RandomForestRegressor(),parameters,cv = 5)
GVC.fit(x_train,y_train)
```

Out[222]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'bootstrap': [True, False],
                         'criterion': ['squared_error', 'mse', 'absolute_error',
                                       'mae'],
                         'max_depth': [1, 10],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [1, 10],
                         'n_estimators': [1, 100]})
```

In [223]:
```
GVC.best_params_
```

Out[223]:
```
{'bootstrap': False,
 'criterion': 'mse',
 'max_depth': 10,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 10,
 'n_estimators': 100}
```

In [224]:
```
Boosted_model = RandomForestRegressor(bootstrap= 'False',criterion = 'mse',max_depth = 10,max_features = 'sqrt',
                                      min_samples_split = 10 ,n_estimators = 100,min_samples_leaf = 1)
Boosted_model.fit(x_train,y_train)
pred = Boosted_model.predict(x_test)
print("accuracy score of the final model is :",r2_score(y_test,pred)*100)
```

```
accuracy score of the final model is : 85.2607468074242
```

Conclusion The final model has an accuracy of the 85.26 % ,after using the following best parameters :

<u>Conclusion</u>: The final model has an accuracy of the 85.26 % ,after using the following best parameters :

'criterion': 'mse',

'max_depth': 10,

'max_features': 'sqrt',

'min_samples_split': 10,

'n_estimators': 100,

'bootstrap': 'False'

,'min_samples_leaf : 1'

Now we'll use this model to predict the car price.

CONCLUSION

By performing different models, it was aimed to get different perspectives and eventually compared their performance. With this study, it purpose was to predict prices of used cars by using a dataset that has 10 predictors and 8868 observations. With the help of the data visualizations and exploratory data analysis, the dataset was uncovered and features were explored deeply. The relation between features were examined. At the last stage, predictive models were applied to predict price of cars in an order: random forest, linear regression, KNeighbors regression, Decision Tree Regression, Gradient Boosting regression.

During this project I learnt a lot about new algorithms, data cleaning process, data scaling, normalization of data using various techniques. the data visualization process, especially count plot in the beginning helped me understanding the dataset. Further by using scatter plot I got to know relationship of different feature with target variables in less time and more effectively. The other visualisation techniques, like box plot, histogram, distribution plot, heat maps help in getting more information in less time and are effective and efficient also. While finalising the model for the project I used basic parameters to check the efficiency of each model. However, Random Forest algorithm was best among the others. After doing hyperparameter tuning of the model, I found that the random forest regressor works best in the parameters{'bootstrap': False, 'criterion': 'mse', 'max_depth': 10,'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}.

The main challenge faced during the project was Cleaning the data and extracting the useful data from the junk data. The Web scrapping and Data cleaning process took the most time. But I learned a lot from this project.