CAR PRICE PREDICTION PROJECT

Submitted by:

SHILPA KEWALARAMANI

ACKNOWLEDGMENT

I would like to acknowledge the contributions of the following people without whose help and guidance this report would not have been completed. I acknowledge the counsel and support of our training coordinator, **Mr. Shubham Yadav**, SME of Internship14 and **Mr. Sajid Chaudhary SME** of Internship15 with respect and gratitude, whose expertise, guidance, support, encouragement, and enthusiasm has made this report possible. Their feedback vastly improved the quality of this report and provided an enthralling experience.

I am indeed proud and fortunate to be supported by them. I am also thankful to **Miss. Vaishali Singh** HR **of FlipRobo Technologies**, and **Miss Neha** Mentor of Batch No 1827 of **DataTrained Institute, Noida** for her constant encouragement, valuable suggestions and moral support.

Although it is not possible to name individually, I shall ever remain indebted to the faculty members **of Data Trained Institute of Technology & Management, Noida for** their persistent support extended during this work.

This acknowledgement will remain incomplete if I fail to express our deep sense of obligation to my parents and God for their consistent blessings and encouragement

INTRODUCTION

- ## Business Problem Framing

Describe the business problem and how this problem can be related to the real world.

Determining the Price of New Car cannot be a tedious task as the manufacturer company can sum up the cost, profit and various other charges but the most tedious task is to determine the price of used car, like how many kilometres car has run, what kind of fuel it uses, how many times the car has been resold and many other things , It is difficult to predict. Sometimes the price may vary because of places , like in some location the used cars with same features is high, but the price of same cars with same features at different location is less. Sometimes the values of cars depends on the Brands also like used luxurious car cannot be of same price as of normal economic car, car price may vary on feature like its engine, sometimes colour , sometimes headlights mainly on variant and model , Number of seats and many things. Also the price of car depends upon how old the car is, i.e., Manufacturing Year also matters most in deciding the car price.

Nowadays there are many company that deal with second hand products no matter it can be car also, Owner of the car may quote a high price to the second_hand_products_buyer_Company . But the company also need a perfect knowledge that buying the product at the car owner's quoted price will definitely affect them. To overcome this a generalized model need to make that will help not only buyer but the seller of the car. that they buy and sell at an appropriate rate.

Conceptual Background of the Domain Problem

Describe the domain related concepts that you think will be useful for better understanding of the project.

The domain of this project is the sale industry. The Used Car Price Prediction Project is similar to House Price Prediction Project. Like in the housing project we were given many parameters that usually matters in predicting the price of a House like area, number of rooms, floor, street, alley, city, location and many things. Here in this project we have to collect the data of the used cars using various websites then clean the data and then train the model and then predict the price of the used cars using model.

As the project is related to the sales industry it can be related with any product that comes in the markets for reselling, like people resell their laptops, cameras other expensive items sometimes vehicles or even the artistic things and much more.

- **Review of Literature**

This is a comprehensive summary of the research done on the topic. The review should enumerate, describe, summarize, evaluate and clarify the research done.

For this projects researches on various websites were made like the olx.com, carsdekho.com, and cars24.com the researches were made on the car model, brands, variant, number of owners of the car and also the cities , like in cities like Delhi, Chennai, Banglore, Ahmedabad there are Thousands plus cars offered for reselling, there were huge varieties of cars available and also some parameters of cars shared were common and others were shared as an additional information, It was also observed that more the info available more the views were on the cars

- **Motivation for the Problem Undertaken**

  Describe your objective behind to make this project, this domain and what is the motivation behind.

  The main Motivation behind building this project was to make a generalized model from the used cars data that will help the car reseller companies to predict the appropriate price for the cars offered for reselling by sharing some features like, number of owners of cars, manufacturing year, fuel type, kilometres driven and few more. This price prediction project is really helpful , generalized and dynamic

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem**

Describe the mathematical, statistical and analytics modelling done during this project along with the proper justification.

Correlation-

correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). It s a common tool for describing simple relationships without making a statement about cause and effect. Here are several types of correlation coefficient, but the most popular is Pearson's correlation the formula for coefficient is given below, the coefficient of each column is calculated automatically in python through corr() function

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[\, n\sum x^2 - (\sum x)^2 \,][\, n\sum y^2 - (\sum y)^2 \,]}}$$

Outlier Removal using InterQuartile (IQR)

The IQR is a measure of variability, based on dividing a data set into quartiles. Quartiles divide a rank-ordered data set into four equal parts. The values that separate parts are called the first, second, and third quartiles; and they are denoted by Q1, Q2, and Q3, respectively.

In descriptive statistics, the Interquartile Range (IQR) also called the mid spread, middle 50% or H-spread, is a measure of statistical dispersion, being equal to the difference between 75^{th} and 25^{th} percentiles, or between upper and lower quartiles,

IQR= $Q_3 - Q_1$. In other words, the IQR is the first quartile subtracted from the third quartile; these quartiles can be clearly seen on a box plot on the data It is a trimmed estimator defined as the 25% trimmed range and is a commonly used

In almost all the columns there were huge Outliers and they need to be cured because extra data can lead to the bias data. To cure outliers z_score was not effective because it was deleting almost all the rows because in each rows any one of the column had z_score > 3.

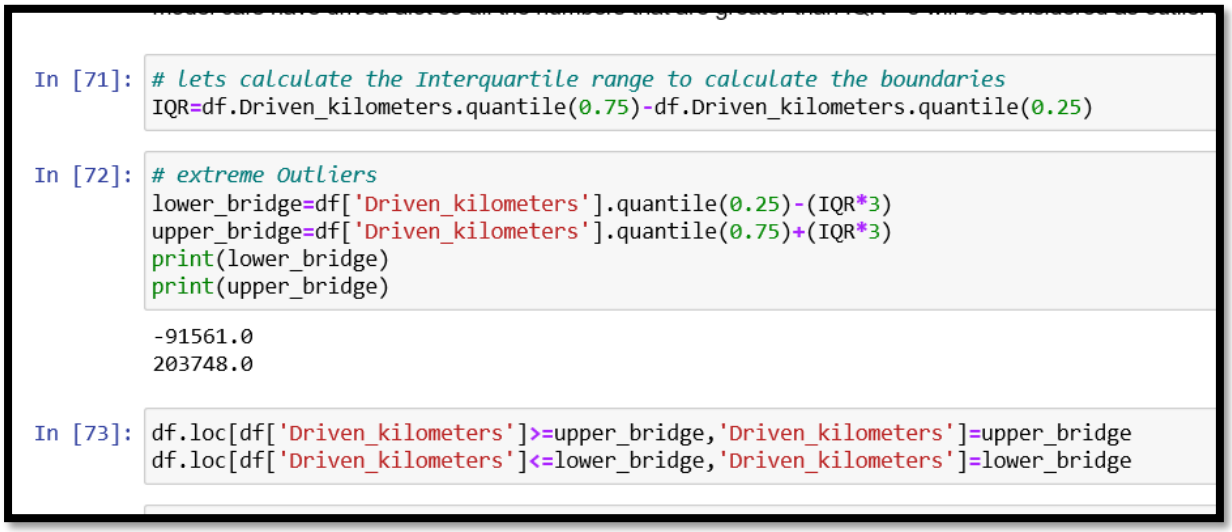
So IQR method was adopted. For IQR method we defined two functions Outlier_removal and Extreme_Outlier_removal .

extreme_outlier_removal -The same procedure is used for extreme_outlier_removal the only thing that differ is lower bridge is calculated as

3 * 25^{th} percentile. and upper_bridge is calculated as 3 * 75^{th} percentile

The question is now when to use which function.

So by looking at the boxplot if the outliers are at extreme distance from the 75^{th} percentile then extreme_outlier_removal function is used and if the outliers are not at extreme distance then outlier_removal function is used.



In [71]:
```
# lets calculate the Interquartile range to calculate the boundaries
IQR=df.Driven_kilometers.quantile(0.75)-df.Driven_kilometers.quantile(0.25)
```

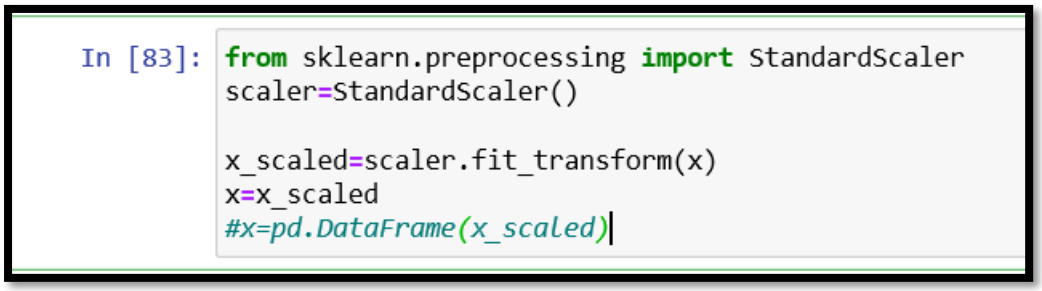
In [72]:
```
# extreme Outliers
lower_bridge=df['Driven_kilometers'].quantile(0.25)-(IQR*3)
upper_bridge=df['Driven_kilometers'].quantile(0.75)+(IQR*3)
print(lower_bridge)
print(upper_bridge)
```

```
-91561.0
203748.0
```

In [73]:
```
df.loc[df['Driven_kilometers']>=upper_bridge,'Driven_kilometers']=upper_bridge
df.loc[df['Driven_kilometers']<=lower_bridge,'Driven_kilometers']=lower_bridge
```

Standard Scaler

Standardize features by removing the mean and scaling to unit variance . We use standard scaler to bring all the data to the same scale like in the column of brands the numbers after label encoding vary between 0-25 or 30, in variants it vary between 0-500 or 600 location 1 to 4 and in driven_kiloemters from thousands to lakhs so to bring the whole data on the same scaler standardization process is used . and StandardScaler is the library that performs this transformation.



Converting the Lakh written in words in the proper number format

It was observed that in the columns like Price and Driven_kilometers people had input the data as 7 lakh , 3.25 lakh, or 4,00,000 or so it was necessary to convert the lakh into number format like 00,000 but if we replace lakh into 00,000 it was okay with 7 lakh type inputs but not with 3.25 lakhs as it converts 3.25 lakh into 3,25,00,000 it became 3 crore so I have written a code it checks the condition properly and converts and I have written it separately for each dataset because passing dataset through function was not working properly. So same code is written for each dataset separately

In [6]:

```
import regex as re
 # if price is 2.3 lakh* converting it as 2,30 000
# if price is 2.45 lakh* then converting it as 2,45,000
# if price is 7 lakh* converting it as 7,00,000

for i in range(0,len(df1['Price'])):
    string=df1['Price'][i]
    pattern='[0-9]+'
    result1=re.findall(pattern,string)
    #print(result1)
    pattern='\W'
    result2=re.findall(pattern,string)
    if result2[0]=='.':
        if int(result1[1]) > 9:
            string=string.replace('Lakh*','000')
            df1['Price'][i]=string
            #print('1')
            #print(string)
        else:
            string=string.replace('Lakh*','0 000')
            df1['Price'][i]=string
            #print('2')
            #print(string)

    else:
        string=string.replace('Lakh*','00000')
        df1['Price'][i]=string
        #print(string)
```

And for Driven kilometres same code was written

In [7]:

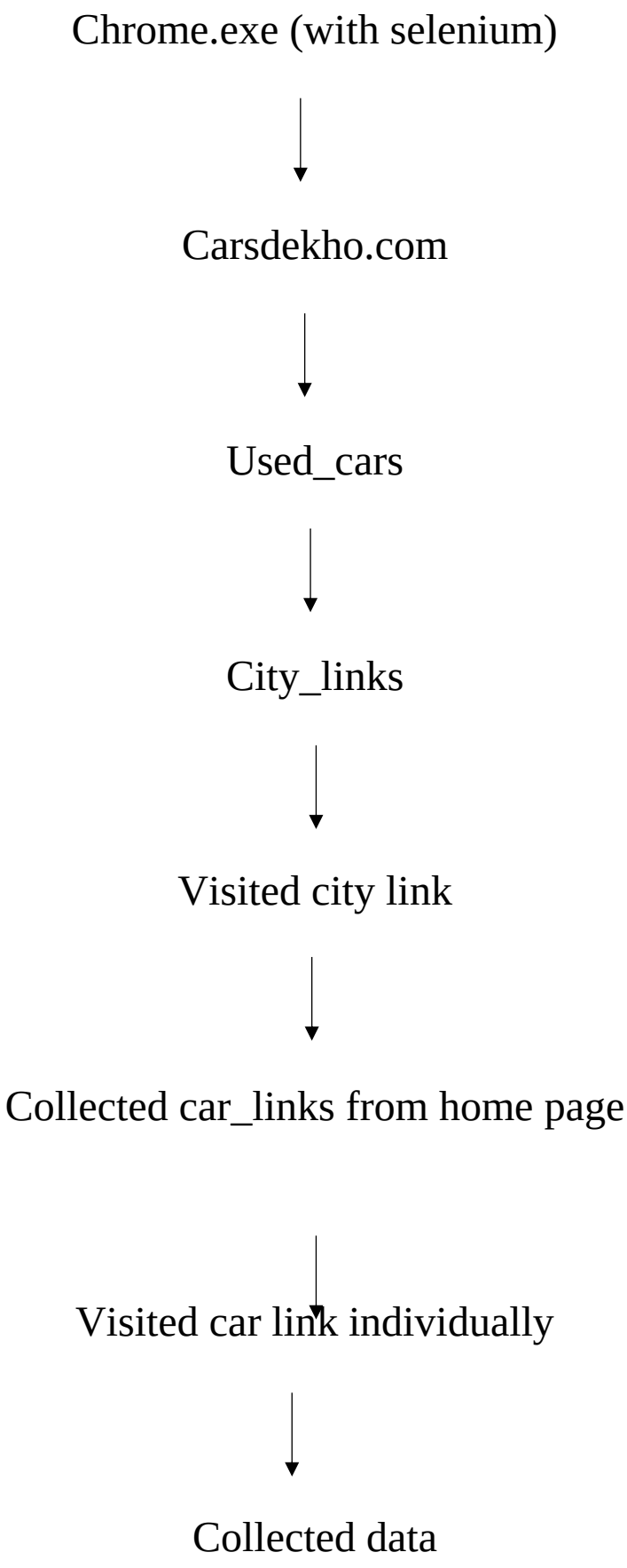
```
# just like prices converting the driven kilometers
for i in range(0,len(df1['Driven_kilometers'])):
    string=df1['Driven_kilometers'][i]
    pattern='[0-9]+'
    result1=re.findall(pattern,string)
    #print(result1)
    pattern='\W'
    result2=re.findall(pattern,string)
    if result2[0]=='.':
        if int(result1[1]) > 9:
            string=string.replace('Lakh*','000')
            df1['Driven_kilometers'][i]=string
            #print('1')
            #print(string)
        else:
            string=string.replace('Lakh*','0 000')
            df1['Driven_kilometers'][i]=string
            #print('2')
            #print(string)

    else:
        string=string.replace('Lakh*','00000')
        df1['Driven_kilometers'][i]=string
        #print(string)
```

- **Data Sources and their formats**

What are the data sources, their origins, their formats and other details that you find necessary? They can be described here. Provide a proper data description. You can also add a snapshot of the data.

The data was collected through website like carsdekho.com then through the websites links were collected to visit cars sold in cities like Ahmedabad, Chennai, Banglore, Delhi individually then collected the car links displayed on the home page of each city. Then each car link was opened and car data was collected .

You can visualize the above flow in the following diagram


Chrome.exe (with selenium)
Carsdekho.com
Used_cars
City_links
Visited city link
Collected car_links from home page
Visited car link individually
Collected data

The Data like Car_Brand, Model, Variant, Price, Fuel_tye, No of owners, Driven_kilometers, location , Manufacturing year etc were collected.

The website was having the data displayed in two types of webpages, so to avoid Exceptions of NoSuchElementException the error was catched and the web scrapping code of both type of webpages were written to collect data you can see in following screenshots

```
try:
            vrnt=driver.find_element_by_xpath('//div[@class="variant-name"]')
            variant.append(vrnt.text)
except NoSuchElementException:
            name=driver.find_element_by_xpath('//div[@class="vdp-head"]/h1[2]')
            n=name.text.split(' ')
           # brand_name.append(n[0])
          #  model_name.append(n[1])
            print(n)
            if len(n) >=3:
                v=n[2:]
            else:
                v=n
            variant.append(' '.join(v))
try:
            yr=driver.find_element_by_xpath('//ul[@class="gsc_row detailsList"]/li[1]/div/div[2]')
            year.append(yr.text)
except NoSuchElementException:
            yr=driver.find_element_by_xpath('//ul[@class="gsc_row clearfix"]/li[1]')
            year.append(yr.text)
           # year.append(' - ')


try:
            fuel=driver.find_element_by_xpath('//ul[@class="gsc_row detailsList"]/li[3]/div/div[2]')
            fuel_type.append(fuel.text)
except NoSuchElementException:
            fuel=driver.find_element_by_xpath('//ul[@class="gsc_row clearfix"]/li[5]')
            fuel_type.append(fuel.text)
            #fuel_type.append(' - ')

try:
            transmission=driver.find_element_by_xpath('//ul[@class="gsc_row detailsList"]/li[8]/div/div[2]')
            key_transmission.append(transmission.text)
except NoSuchElementException:
            transmission=driver.find_element_by_xpath('//ul[@class="gsc_row clearfix"]/li[6]')
            key_transmission.append(transmission.text)
            #key_transmission.append(' - ')

try:
            km=driver.find_element_by_xpath('//ul[@class="gsc_row detailsList"]/li[4]/div/div[2]')
            km_driven.append(km.text)
except NoSuchElementException:
            km=driver.find_element_by_xpath('//ul[@class="gsc_row clearfix"]/li[3]')
            km_driven.append(km.text)
            #km_driven.append(' - ')

try:
            owner=driver.find_element_by_xpath('//ul[@class="gsc_row detailsList"]/li[6]/div/div[2]')
            no_of_owner.append(owner.text)
except NoSuchElementException:
            owner=driver.find_element_by_xpath('//ul[@class="gsc_row clearfix"]/li[4]')
            no_of_owner.append(owner.text)
            #no_of_owner.append(' - ')

try:
            pr=driver.find_element_by_xpath('//div[@class="priceSection"]')
            price.append(pr.text)
except NoSuchElementException:
            pr=driver.find_element_by_xpath('//span[@class="amount"]')
            price.append(pr.text)
            #price.append(' - ')
```

The data then collected was combined together in the DataFrame. For each cities the different DataFrames were created and exported to a csv file for later use also

```
In [152]: banglore_cars=pd.DataFrame({})

In [153]: banglore_cars['Brand']=brand_name
          banglore_cars['Model']=model_name
          banglore_cars['Variant']=variant
          banglore_cars['Manufacturing_year']=year
          banglore_cars['Driven_kilometers']=km_driven
          banglore_cars['Fuel']=fuel_type
          banglore_cars['number_of_owners']=no_of_owner
          banglore_cars['location']=location
          banglore_cars['Price']=price

In [154]: banglore_cars.to_csv('banglore_cars_data')

In [155]: banglore_cars.shape
Out[155]: (1767, 9)
```

At the time of model building all the DataFrames were concatenated in single DataFrame.

concatinating all the frames in the single dataframe

```
In [30]: frames=[df1,df2,df3,df4]

In [31]: df=pd.concat(frames)

In [32]: df.shape
Out[32]: (7349, 10)
```

The collected data was having datatypes like

```
In [34]: # checking the datatypes
         df.dtypes

Out[34]: Unnamed: 0            int64
         Brand                object
         Model                object
         Variant              object
         Manufacturing_year    int64
         Driven_kilometers    object
         Fuel                 object
         number_of_owners     object
         location             object
         Price                object
         dtype: object
```

In this way data was collected and further processed for Model Building.

- **Data Preprocessing Done**

What were the steps followed for the cleaning of the data? What were the assumptions done and what were the next actions steps over that?

The data collected was cleaned to a great extent like

1) While checking for the datatypes of the dataframe it can be observed that

* Unnamed:0 is the count column so we can drop it.

* Driven_kilometers should be in integer format but here it is in object format

* price column should also be in integer format but here it is in object format

2)Dropping the column

* the columns like **Unnamed:0** and **Location** were dropped as they were not contributing in the model building.

3) Cleaning the Driven_kilometer column and type casting it as Integer

At the Time of loading the csv files the Driven_kilometer column was treated with respect to lakh at suffix (screenshot is shared at the start) then also he column Driven_kilometer was having suffixes like **kms, Kms, ‘ , ’ , ‘ . ’ lakh*, and white space** so it was all replaced with no space and the column was then converted into integer type you can see it in the following screenshot.

```
In [41]: # Cleaning Driven kilometer column

In [42]: # removing Kms from the column
         df['Driven_kilometers']=df['Driven_kilometers'].str.replace('kms',' ')
         df['Driven_kilometers']=df['Driven_kilometers'].str.replace('Kms',' ')
         df['Driven_kilometers']=df['Driven_kilometers'].str.replace(',','')
         df['Driven_kilometers']=df['Driven_kilometers'].str.replace('Lakh','00000')
         df['Driven_kilometers']=df['Driven_kilometers'].str.replace(' ','')

In [43]: # converting  object column to integer  column
         df['Driven_kilometers']=df['Driven_kilometers'].astype(int)

In [44]: df['Driven_kilometers'].value_counts()

Out[44]: 70000     250
         80000     241
         60000     233
         90000     140
         120000    137
                  ...
         95100       1
         48001       1
         64325       1
         9110        1
         8200        1
         Name: Driven_kilometers, Length: 1759, dtype: int64
```

4) Cleaning the Price column and type casting it as Integer

At the Time of loading the csv files the Price column was treated with respect to lakh at suffix (screenshot is shared at the start) then also the column **price** was having suffixes like **₹ , ‘ , ’ , ‘ . ’ , lakh* , ‘ Cr ’**

```
In [51]: # cleaning the price column
         df['Price']=df['Price'].str.replace('Lakh*','000')
         df['Price']=df['Price'].str.replace('.','')
         df['Price']=df['Price'].str.replace('*','')
         df['Price']=df['Price'].str.replace('₹','')
         df['Price']=df['Price'].str.replace(',','')
         df['Price']=df['Price'].str.replace(' ','')
         df['Price']=df['Price'].str.replace('Cr','000000')

In [52]: # typecasting the price column
         df['Price']=df['Price'].astype('int64')

In [53]: df['Price'].head()

Out[53]: 0    448500
         1    474500
         2    415500
         3    273500
         4    539000
         Name: Price, dtype: int64

In [ ]:
```

, and white space so it was all replaced with no space and the column was then converted into integer type you can see it in the following screenshot.

5)cleaning number_of_owner column

The column number of owner was having the same data but written in different format for the first owner the data was **‘First Owner’ and also ‘1 st Owner’** similarly for second owner and third owner so all were bring in the same format As shown in the following screenshot

```
In [46]: df['number_of_owners'].value_counts()

Out[46]: First Owner             4991
         Second Owner            1602
         1st Owner                344
         Third Owner              271
         2nd Owner                 83
         Fourth & Above Owner      50
         Test Drive Car             6
         3rd Owner                  1
         4th Owner                  1
         Name: number_of_owners, dtype: int64

In [47]: #renaming values
         df['number_of_owners']=df['number_of_owners'].str.replace('1st Owner','First Owner')
         df['number_of_owners']=df['number_of_owners'].str.replace('2nd Owner','Second Owner')
         df['number_of_owners']=df['number_of_owners'].str.replace('3rd Owner','Third Owner')

In [48]: df['number_of_owners'].value_counts()

Out[48]: First Owner             5335
         Second Owner            1685
         Third Owner              272
         Fourth & Above Owner      50
         Test Drive Car             6
         4th Owner                  1
         Name: number_of_owners, dtype: int64
```

The columns like **Brand , Model, Variant, Manufacturing_year and Fuel Type** were having cleaned data so no need to clean them.

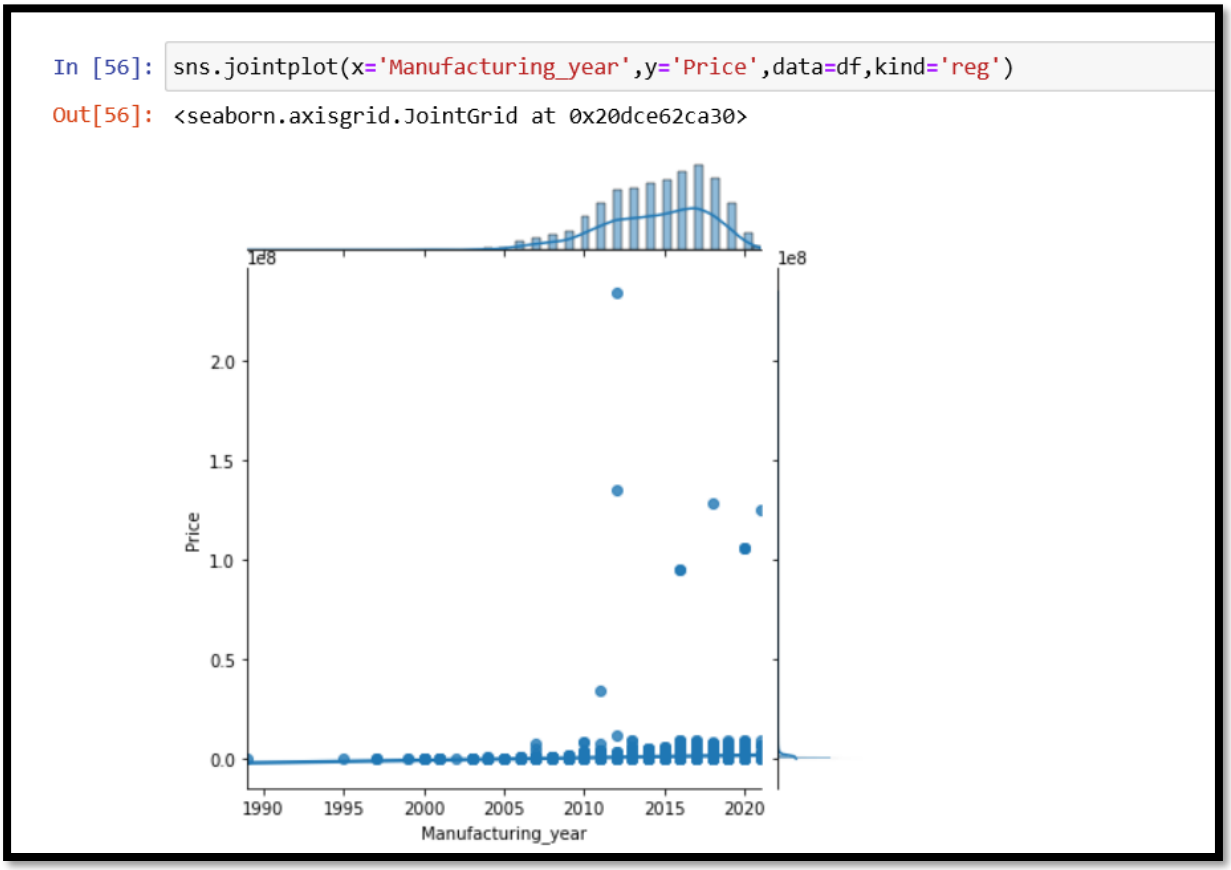
- **Data Inputs- Logic- Output Relationships**

Describe the relationship behind the data input, its format, the logic in between and the output. Describe how the input affects the output.

The relation between the Input Data and The output data was found when Bivariate analysis was done separately between Numerical column and Categorical column

Bivariate analysis with Numerical Column :

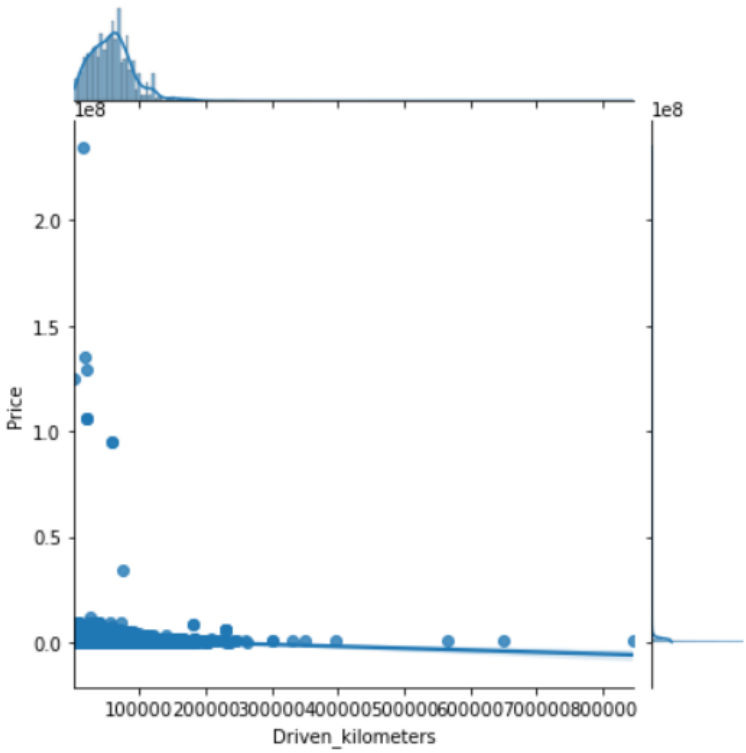
1) Manufacturing Column



Manufacturing Year is showing left skewed data and it is highly posssible as we have seen in the data that old car models are also offered for sale and as it showing no direct relation with the sale means both are increasing with same ratio

2) Driven_Kilometer

```
In [57]: sns.jointplot(x='Driven_kilometers',y='Price',data=df,kind='reg')
Out[57]: <seaborn.axisgrid.JointGrid at 0x20dd40c6640>
```

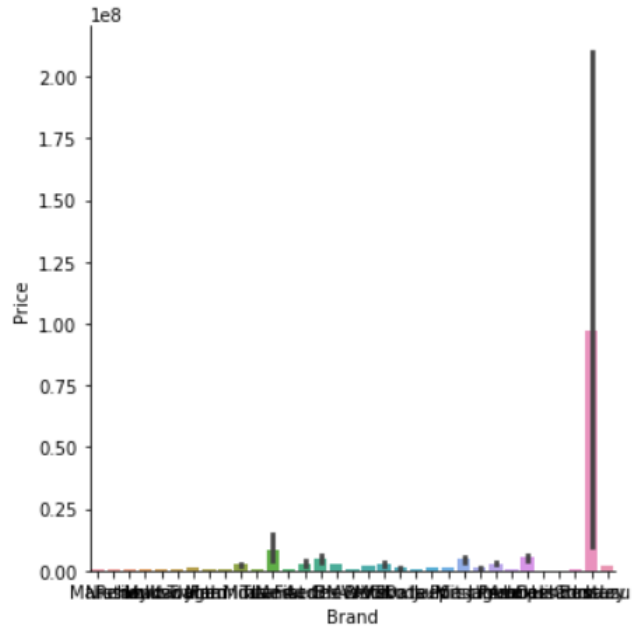


It can be observed that many cars have driven kilometer if 0 to 2,00,000 and further from it less cars have driven kilometer of 250000 and more and it is showing no positive relation with output data price

Bivariate analysis with Categorical Column :

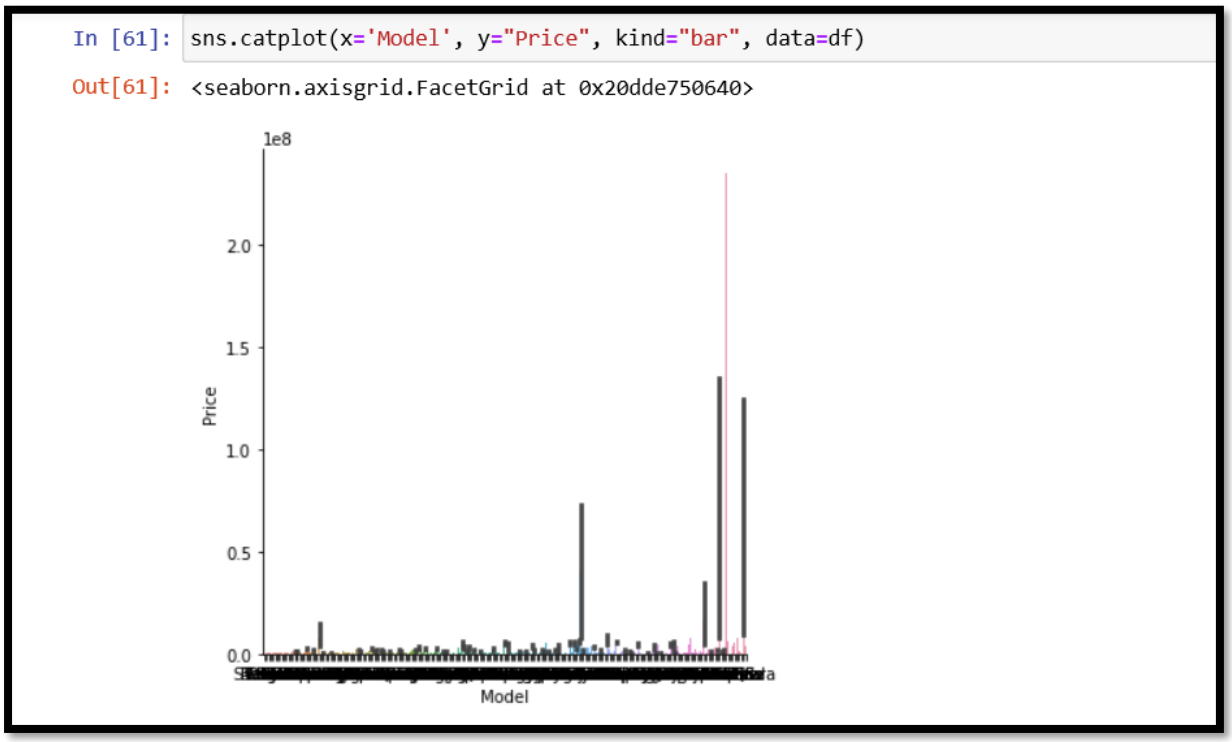
1) Brand

```
In [60]: p=sns.catplot(x='Brand', y="Price", kind="bar", data=df)
```



As there are many brands it cannot be said that which brand has highest selling price by looking at the figure

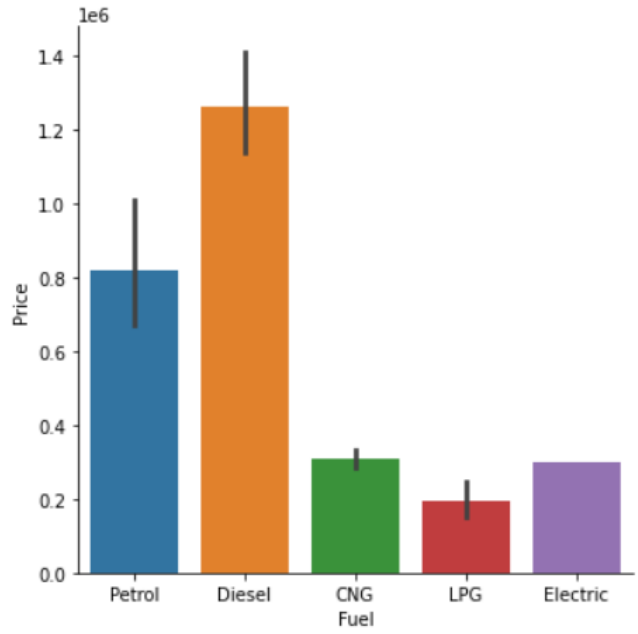
2)Model



Just like Brand models are also many in numbers and it cannnot be figured out which has highest reselling value ust like Brand models are also many in numbers and it cannnot be figured out which has highest reselling value

3) Fuel

```
In [62]: p=sns.catplot(x='Fuel', y="Price", kind="bar", data=df)
```

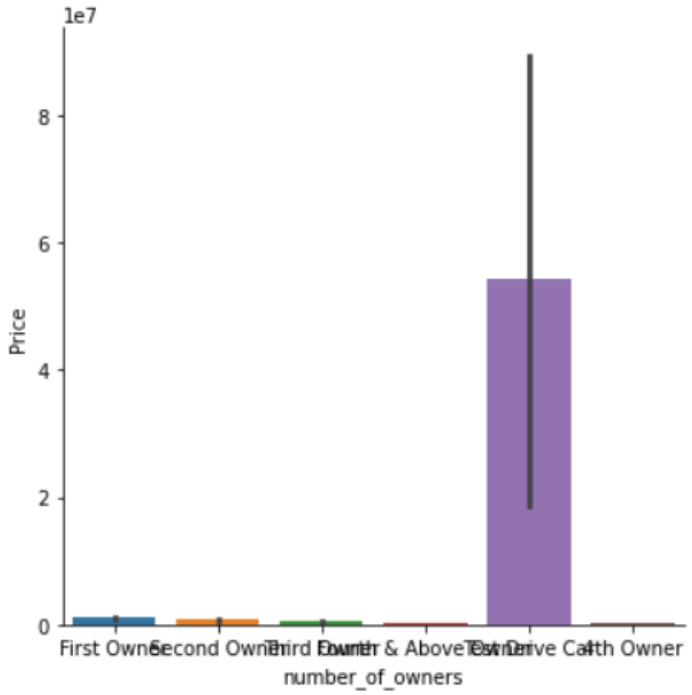


The car which have fuel type as Diesel has highest reselling value followed by Petrol then electric , then CNG and finally LPG

4) Number_of_owners

```
In [63]: p=sns.catplot(x='number_of_owners', y="Price", kind="bar", data=df)
         print(df['number_of_owners'].value_counts())
```

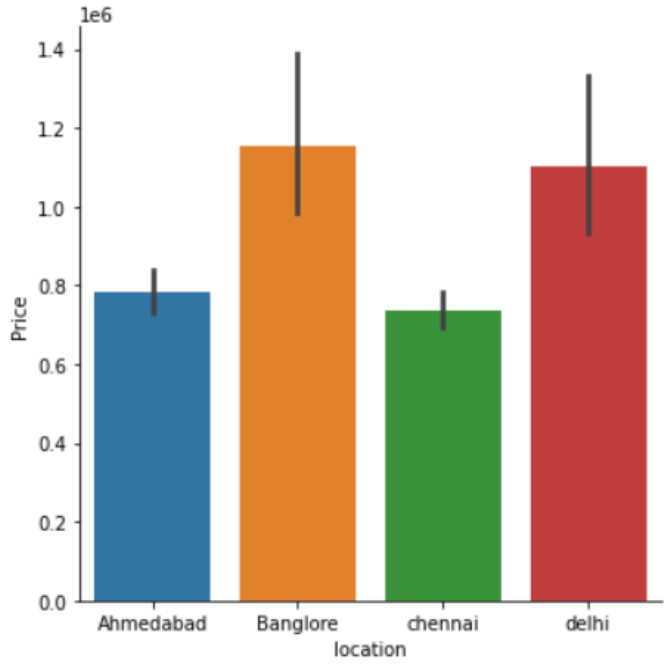
```
First Owner             5335
Second Owner            1685
Third Owner              272
Fourth & Above Owner      50
Test Drive Car             6
4th Owner                  1
Name: number_of_owners, dtype: int64
```



it can be observed that though there is highest number of First owner second owner little less number of Third owner, very few number of fourth and above owner still Test drive car has highest reselling sale value.

5) Location

```
In [64]: p=sns.catplot(x='location', y="Price", kind="bar", data=df)
```



It can be observed that cars sold in Delhi and Bangalore have highest reselling value as compared to then Ahmedabad and Chennai.

- **State the set of assumptions (if any) related to the problem under consideration**

  Here, you can describe any presumptions taken by you.

  The presumptions taken by me where as follows

  1) Manufacturing_year
     While plotting the Outliers through boxplot it was found that there are many outliers but the assumption was :- There may be possibility that the old cars are offered for sale and some models of cars can be old so we will not remove this outlier, as we aim to make dynamic and generalized model so for that our model should know about the old cars their sale value so not removing the outliers

  2) Driven-Kilometers :- the same in plotting the outliers using boxplot many outliers were found at that time assumption made was:- As driven_kilometer is showing some outlier we need to cure it to some extent. we will take iqr * 3 for curing outlier becausae there may be possibility that old model cars have drived alot so all the numbers that are greater than IQR * 3 will be considered as outlier and replaced with IQR * 3

- ## Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows to launch applications and easily manage conda packages , environments , and channels without using command-line commands. Navigators are searching for packages on Anaconda.org or in a Anaconda Repository.

Anaconda Navigator comprises of many applications like Jupyter Notebook, Jupyter lab, Pyshell and many more which are more essential for the Data Scientist.

Jupyter Notebook is an open-source web application that allows to create and share document that contain live code, equations, visualiazations and narrative text.

we used jupyter notebook for importing libraries, loading dataset, data cleaning and transformation, numerical simulation,statistical modelling, visualization of data, building the models and much more

Libraries :-

Pandas Library is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

In our project we use pandas for various purposes like for importing dataset we used read_csv() , as there were 36 columns we were not able to see all, it became possible using pd.set_options('display.max_rows'None) function.

At last for separating the features and target variable pandas library was used like drop function to drop target from feature and to create new feature column again dataframe of target was created using library function.

Numpy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and metrics. NumPy was created in 2005 by Travis Oliphant.

In our project we used numpy for various things like for converting negative into positive values we used absolute abs() function. Besides min,max ,Iqr are also derived from Numpy library.

Matplotlib and Seaborn is a plotting library for the Python programming language and its numerical mathematics extension NumPy. Using matplotlib library we plot various graphs like histogram, countplot, boxplot, distplot. Using seaborn we plot heatmap of the correlation

Scikit-learn is a library in python that provides many unsupervised and supervised learning algorithms. Its built upon some of the technology we used like NumPy , pandas, and Matplotlib

The functionality that scikit-learn provides include:

i. Regression, including Linear and Logistic Regression
ii. Classification , including K-Means and K-Means++
iii. Model selection
iv. Preprocessing including Min-Max Normalization

Like using sklearn.model_selection we used train_test_split, cross_val_score, GridSearchCV,

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Describe the approaches you followed, both statistical and analytical, for solving of this problem.

- **Testing of Identified Approaches (Algorithms)**

  Listing down all the algorithms used for the training and testing.

1) **For Splitting the columns in dependent and independent features**

   i)train_test_split

   train_test_split function was used which is imported from sklearn.model_selection

2) For Training the Model

i) LinearRegression

   I LinearRegression Model was used , which is imported from sklearn.linear_model

ii) DecisionTreeClassifier

   DecisionTreeClassifier Model was used, which is imported from sklearn.tree

3) For Cross Validation

i) Cross_Val_Score

cross_val_score is used for cross_validation which is imported from sklearn.model_selection

4) **For Ensembling**

   i) RandomForestRegressor

      RandomForestRegressor is used for ensembling which is imported from sklearn.ensemble

5) **For Regularization**

   i) Lasso

      From sklearn.linear_model Lasso is imported to regularize

   ii) Ridge

      From sklearn.linear_model Ridge is imported to regularize

6) **For Saving the model**

   i) Pickle

      Pickle library is imported to save the model in .pkl file

- **Run and Evaluate selected models**

  Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

Linear Regression- it is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables).

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Traing the Linear Regression Model and finding the best randomstate an r2score ¶

In [88]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [89]:
```
maxr2score=0
maxRandomState=0
for i in range(1,1000):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.22,random_state=i)
    lr=LinearRegression()
    lr.fit(x_train,y_train)
    pred=lr.predict(x_test)
    r2score=r2_score(y_test,pred)
    if r2score > maxr2score:
        maxr2score= r2score
        maxRandomState=i
print("Best Accuracy is ",maxr2score," on Random State",maxRandomState)
```

```
Best Accuracy is  0.1117835653759447  on Random State 808
```

In [90]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.22,random_state=808)
lr=LinearRegression()
lr.fit(x_train,y_train)
predlr=lr.predict(x_test)
r2score=r2_score(y_test,predlr)
print(r2score)
```

```
0.1117835653759447
```

DecisionTreeRegressor :-

Decision tree is the basic building block of the random forest algorithm. It is considered as one of the most popular algorithms in machine learning and is used for classification purposes. The decision given out by a decision tree can be used to explain why a certain prediction was made. This means the in and out of the process would be clear to the user. They are also known as CART, i.e Classification And Regression Trees. It can be visualized as a binary tree (the one studied in data structures and algorithms).

Every node in the tree represents a single input variable, and the leaf nodes (which are also known as terminal nodes) contain output variable. These leaf nodes are used to make the prediction on the node. When a decision tree is being created, the basic idea is that the given space is being divided into multiple sections. All the values are put up and different splits are tried so as to attain less cost and best prediction values. These values are chosen in a greedy manner.

Splitting up of these nodes goes on until the maximum depth of the tree is reached. The idea behind using decision tree is to divide the input dataset into smaller dataset based on specific feature value until every target variable falls under one single category. This split is made so as to get the maximum information gain for every step.

Every decision tree begins with a root, and this is the place where the first split is made. An efficient way should be devised to ensure that the nodes are defined.

This is where Gini value comes into picture. Gini is considered to be one of the most commonly used measurement to measure inequality. Inequality refers to the target class (output) which every subset in a node may belong to.

Traing the DecisionTree Model and finding the best randomstate and r2_score

93]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```

94]:
```
maxr2score=0
maxRandomState=0
for i in range(1,1000):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.22,random_state=i)
    dtr=DecisionTreeRegressor()
    dtr.fit(x_train,y_train)
    preddtr=dtr.predict(x_test)
    r2score=r2_score(y_test,preddtr)
    if r2score > maxr2score:
        maxr2score= r2score
        maxRandomState=i
print("Best Accuracy is ",maxr2score," on Random State",maxRandomState)
```

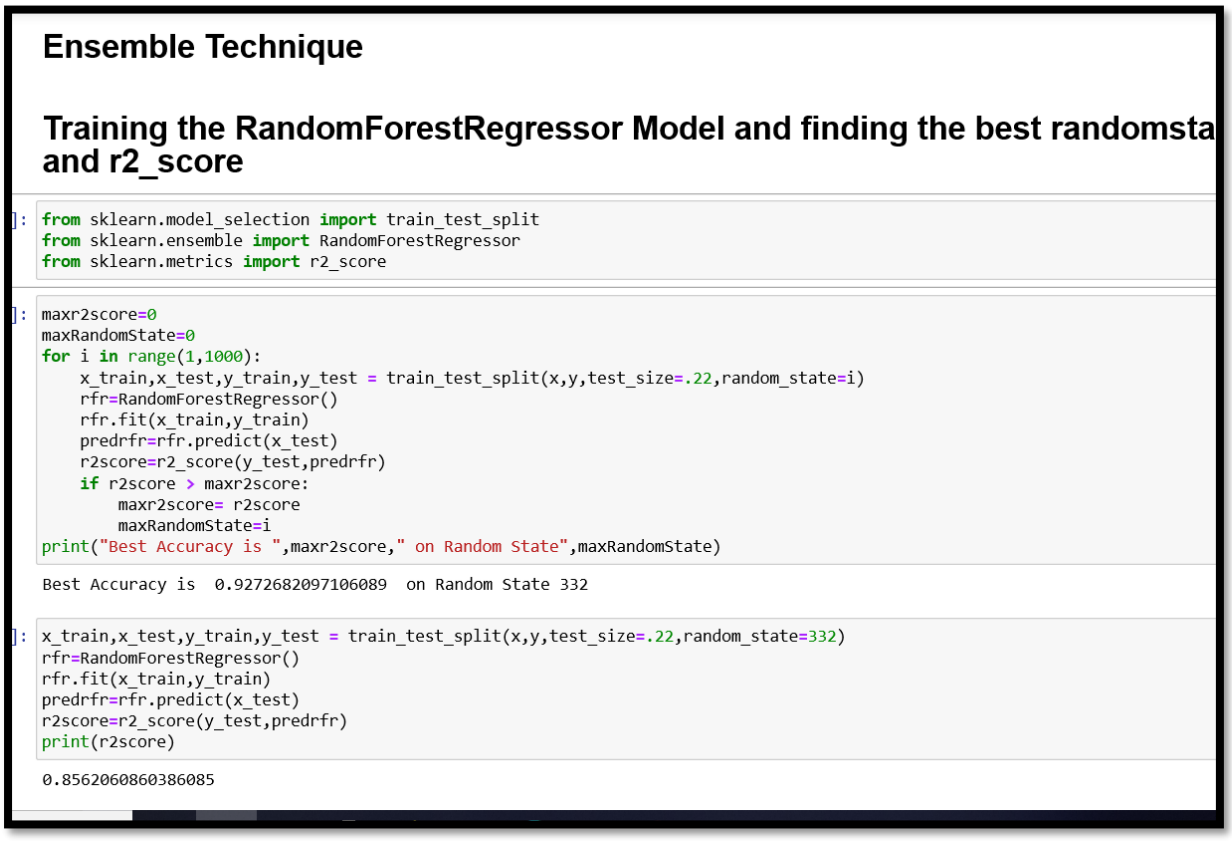
```
Best Accuracy is  0.9832492109830121  on Random State 942
```

95]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.22,random_state=595)
dtr=DecisionTreeRegressor()
dtr.fit(x_train,y_train)
preddtr=dtr.predict(x_test)
r2score=r2_score(y_test,preddtr)
print(r2score)
```

```
0.9850506474141556
```

RandomForestRegressor : - Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration



Ensemble Technique

Training the RandomForestRegressor Model and finding the best randomsta and r2_score

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
```

```
maxr2score=0
maxRandomState=0
for i in range(1,1000):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.22,random_state=i)
    rfr=RandomForestRegressor()
    rfr.fit(x_train,y_train)
    predrfr=rfr.predict(x_test)
    r2score=r2_score(y_test,predrfr)
    if r2score > maxr2score:
        maxr2score= r2score
        maxRandomState=i
print("Best Accuracy is ",maxr2score," on Random State",maxRandomState)
```

```
Best Accuracy is  0.9272682097106089  on Random State 332
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.22,random_state=332)
rfr=RandomForestRegressor()
rfr.fit(x_train,y_train)
predrfr=rfr.predict(x_test)
r2score=r2_score(y_test,predrfr)
print(r2score)
```

```
0.8562060860386085
```

Regularization

Lasso and Ridge Regression

LASSO stands for *Least Absolute Shrinkage and Selection Operator*. I know it doesn't give much of an idea but there are 2 key words here – '*absolute*' and '*selection*'.

Lets consider the former first and worry about the latter later.

Lasso regression performs **L1 regularization**, i.e. it adds a factor of sum of absolute value of coefficients in the optimization objective. Thus, lasso regression optimizes the following:

Objective = RSS + α * (sum of absolute value of coefficients)

As mentioned before, ridge regression performs '**L2 regularization**', i.e. it adds a factor of sum of squares of coefficients in the optimization objective. Thus, ridge regression optimizes the following:

Objective = RSS + α * (sum of square of coefficients)

Here, α (alpha) is the parameter which balances the amount of emphasis given to minimizing RSS vs minimizing sum of square of coefficients. α can take various values:

α = 0:

The objective becomes same as simple linear regression.

We'll get the same coefficients as simple linear regression.

α = ∞:

The coefficients will be zero. Why? Because of infinite weightage on square of coefficients, anything less than zero will make the objective infinite.

0 < α < ∞:

The magnitude of α will decide the weightage given to different parts of objective.

The coefficients will be somewhere between 0 and ones for simple linear regression.

Lasso

In [103]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
import warnings
warnings.filterwarnings('ignore')
```

In [104]:
```
from sklearn.linear_model import Lasso
parameters={'alpha':[.0001,.001,.01,.1,1,10], 'random_state':list(range(0,10))}
ls=Lasso()
clf=GridSearchCV(ls,parameters,cv=8)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'alpha': 10, 'random_state': 0}
```

In [106]:
```
ls=Lasso(alpha=10,random_state=0)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)

pred_ls = ls.predict(x_test)

lss=r2_score(y_test,pred_ls)
lss
```

Out[106]:
```
0.03520420201989827
```

Ridge

Ridge

In [108]:
```
from sklearn.linear_model import Ridge
parameters={'alpha':[.0001,.001,.01,.1,1,10], 'random_state':list(range(0,10))}
rd=Ridge()
clf=GridSearchCV(rd,parameters,cv=8)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'alpha': 10, 'random_state': 0}
```

In [109]:
```
rd=Ridge(alpha=10,random_state=0)
rd.fit(x_train,y_train)
rd.score(x_train,y_train)
pred_rd = rd.predict(x_test)

rds=r2_score(y_test,pred_rd)
rds
```

Out[109]:
```
0.03521350958402414
```

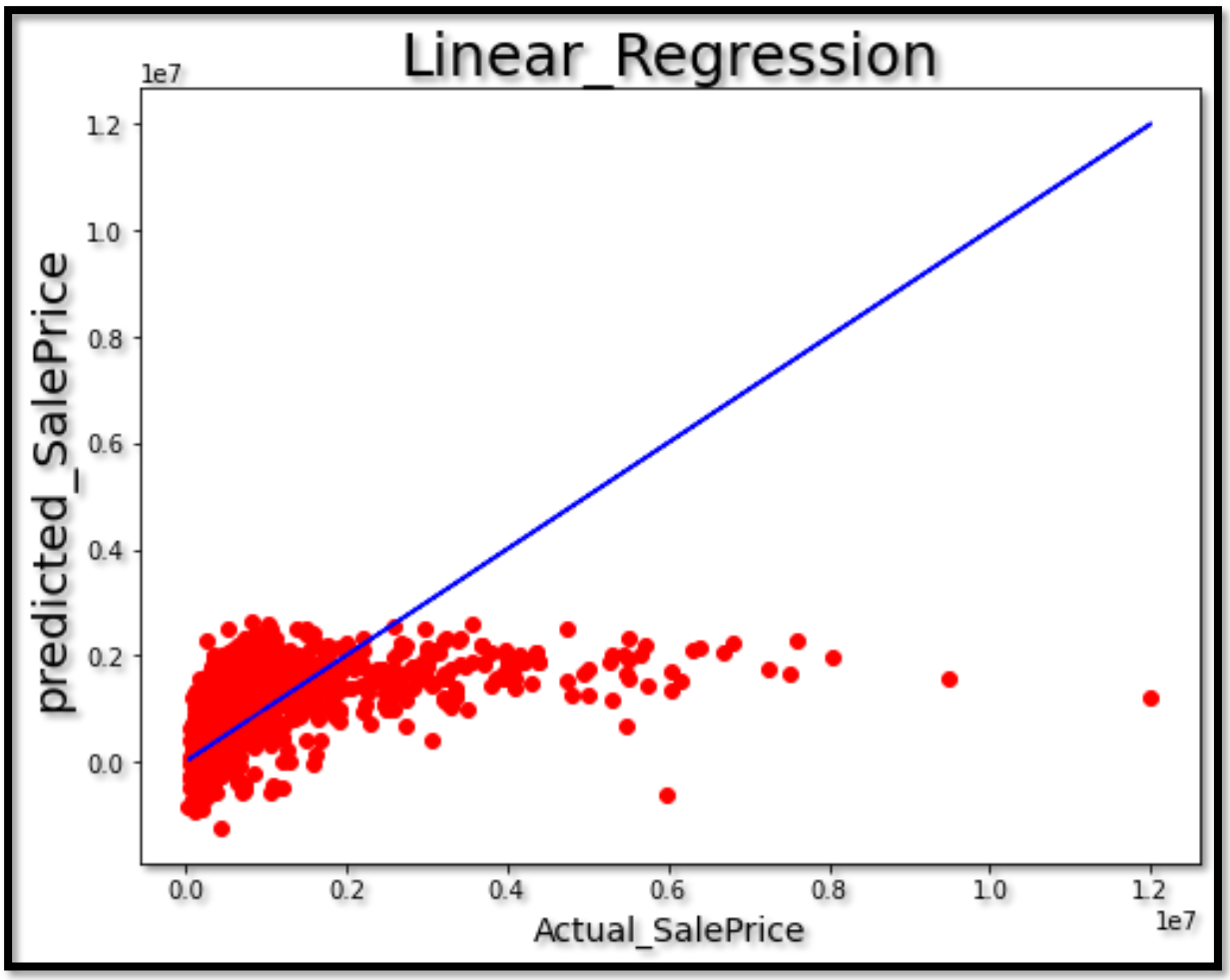
In [110]:
```

import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_rd,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel("Actual_SalePrice",fontsize=14)
plt.ylabel("predicted_SalePrice",fontsize=20)
plt.title("Ridge",fontsize=25)
plt.show()
```

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

  What were the key metrics used along with justification for using it? You may also include statistical metrics used if any.

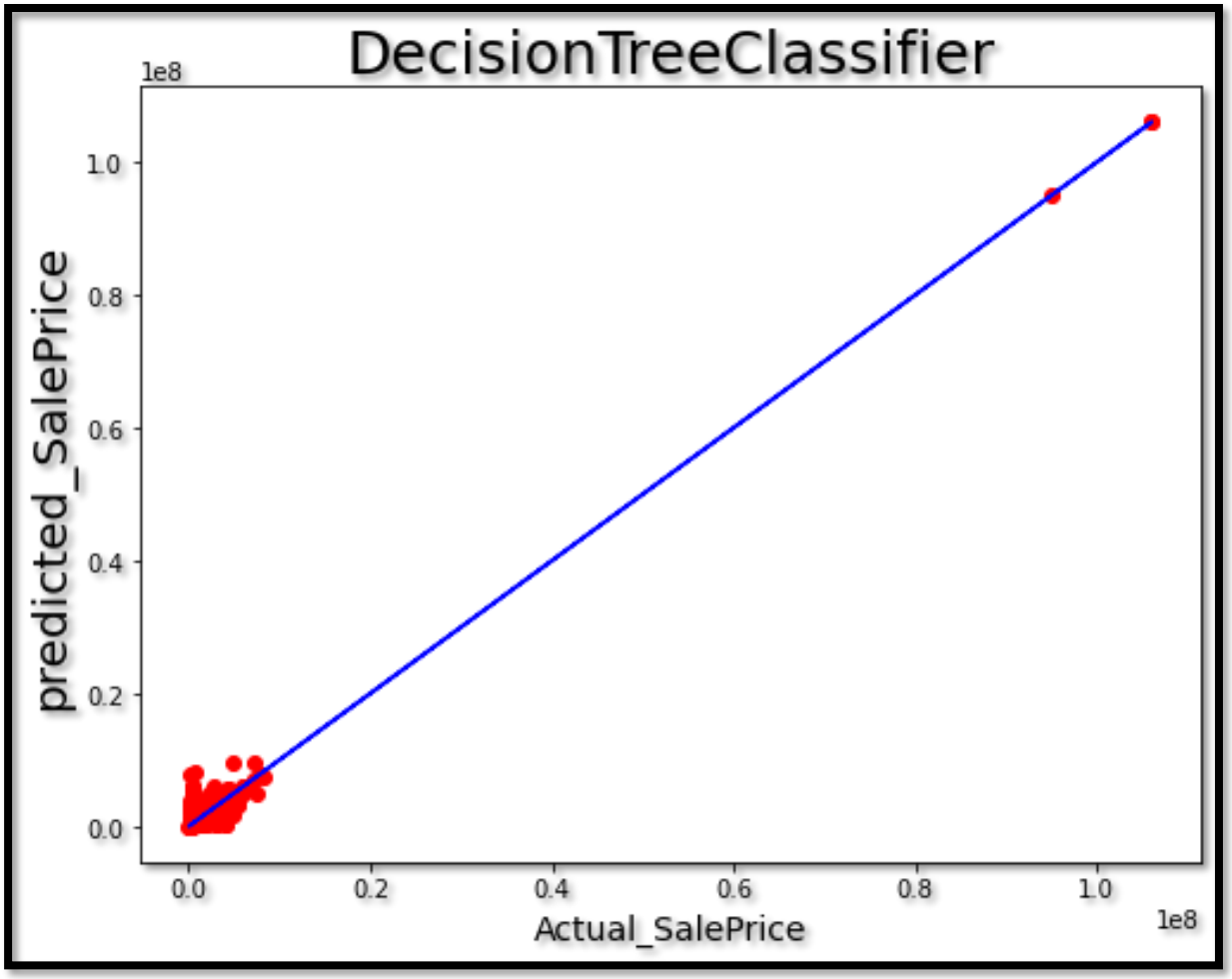
For the Justification of the model r2_score and a scatter plot was used.

Linear Regression :- In Linear Regression the r2_score was 11 % but the result in scatter plot was not so desirable because of Mean Squared Error or Mean Absolute Error. The following is the scatter plot of Linear Regression



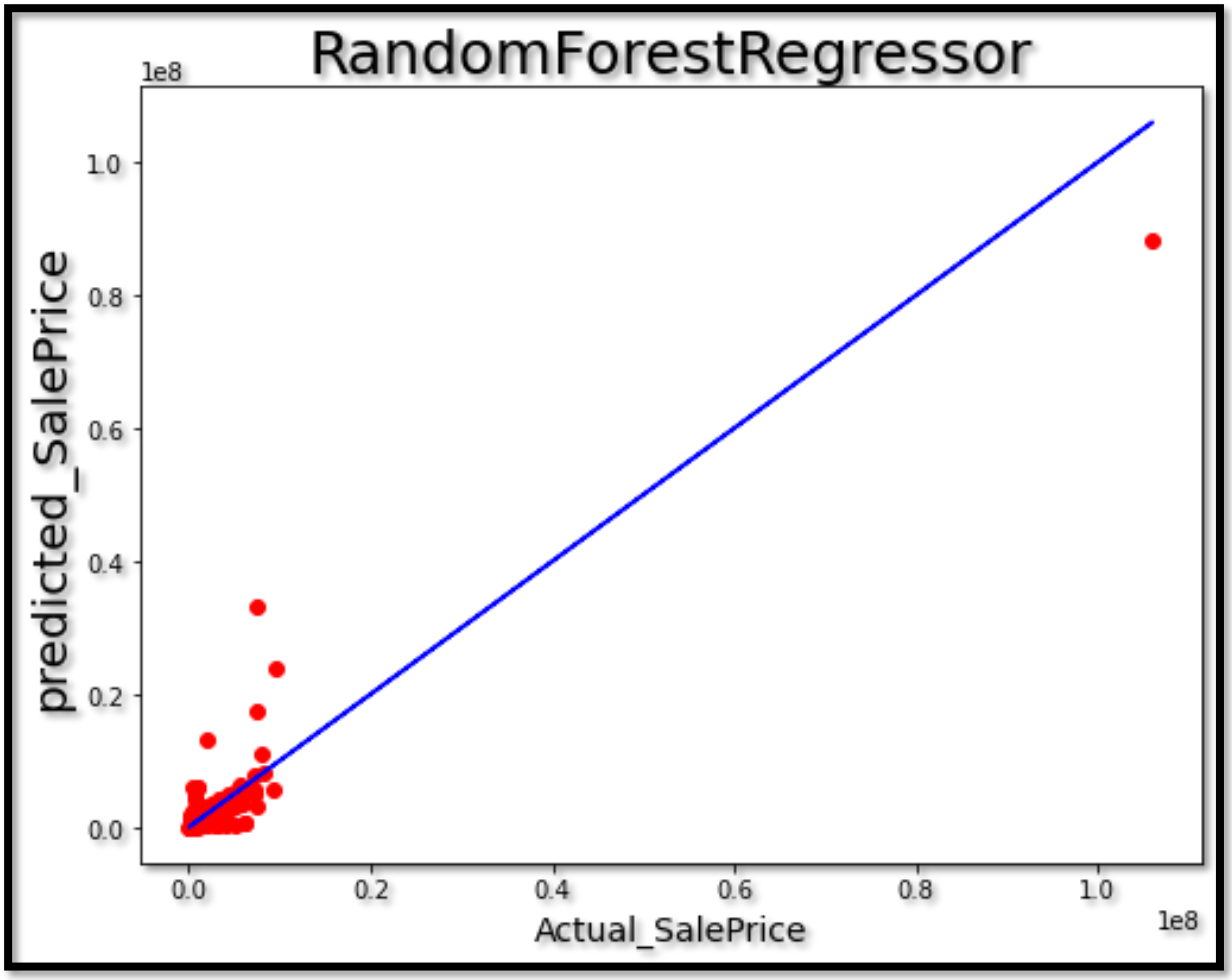
The red points are the predicted price and the blue line is the actual sale pric

DecisionTreeRegressor In DecisionTreeRegressor the r2_score was 98% and the scatter plot of predicted and actual sale price was



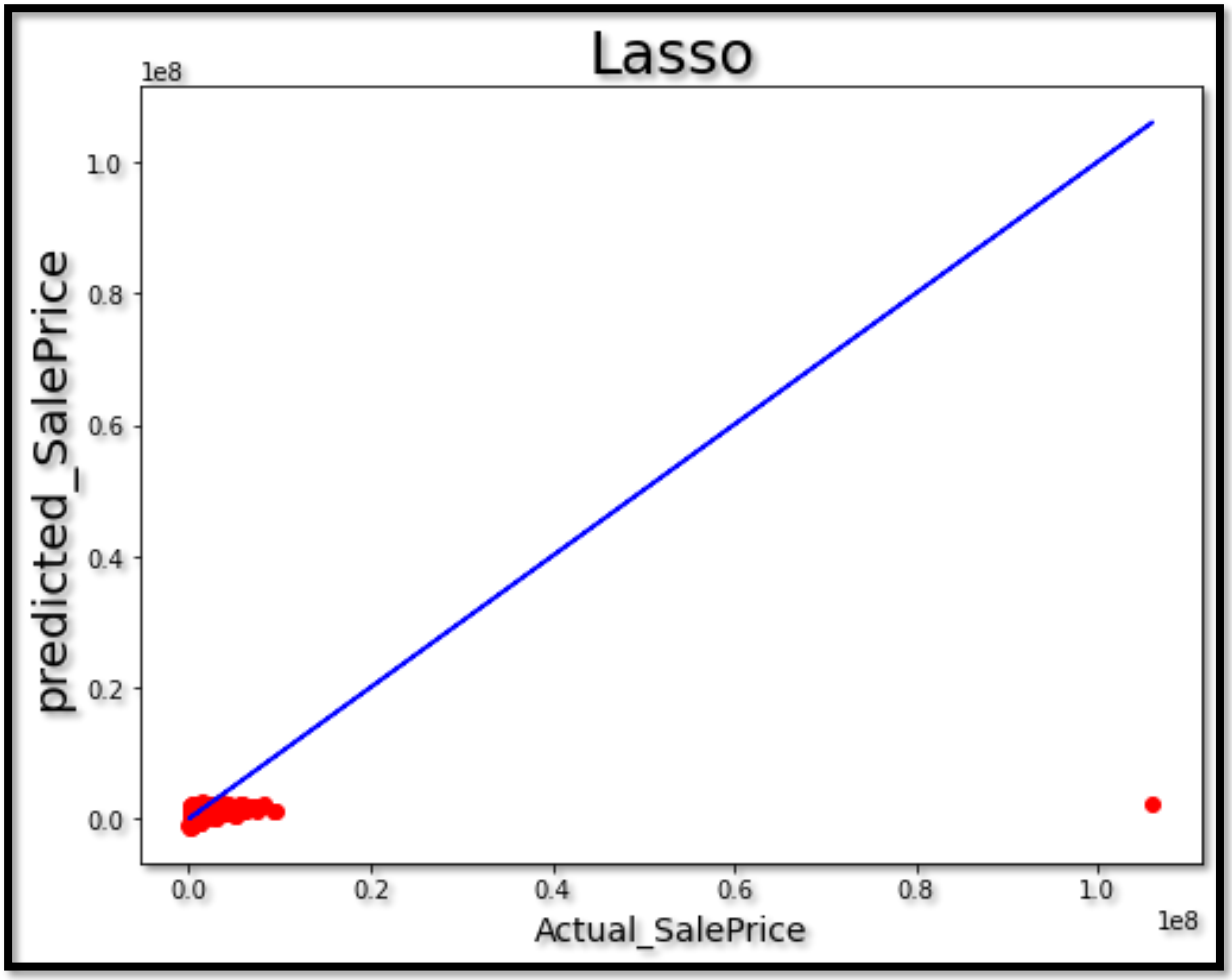
The red points are the predicted price and the blue line is the actual sale price

RandomForestRegressor In RandomForestRegressor the r2_score was 92% and the scatter plot of actual points and predited point was



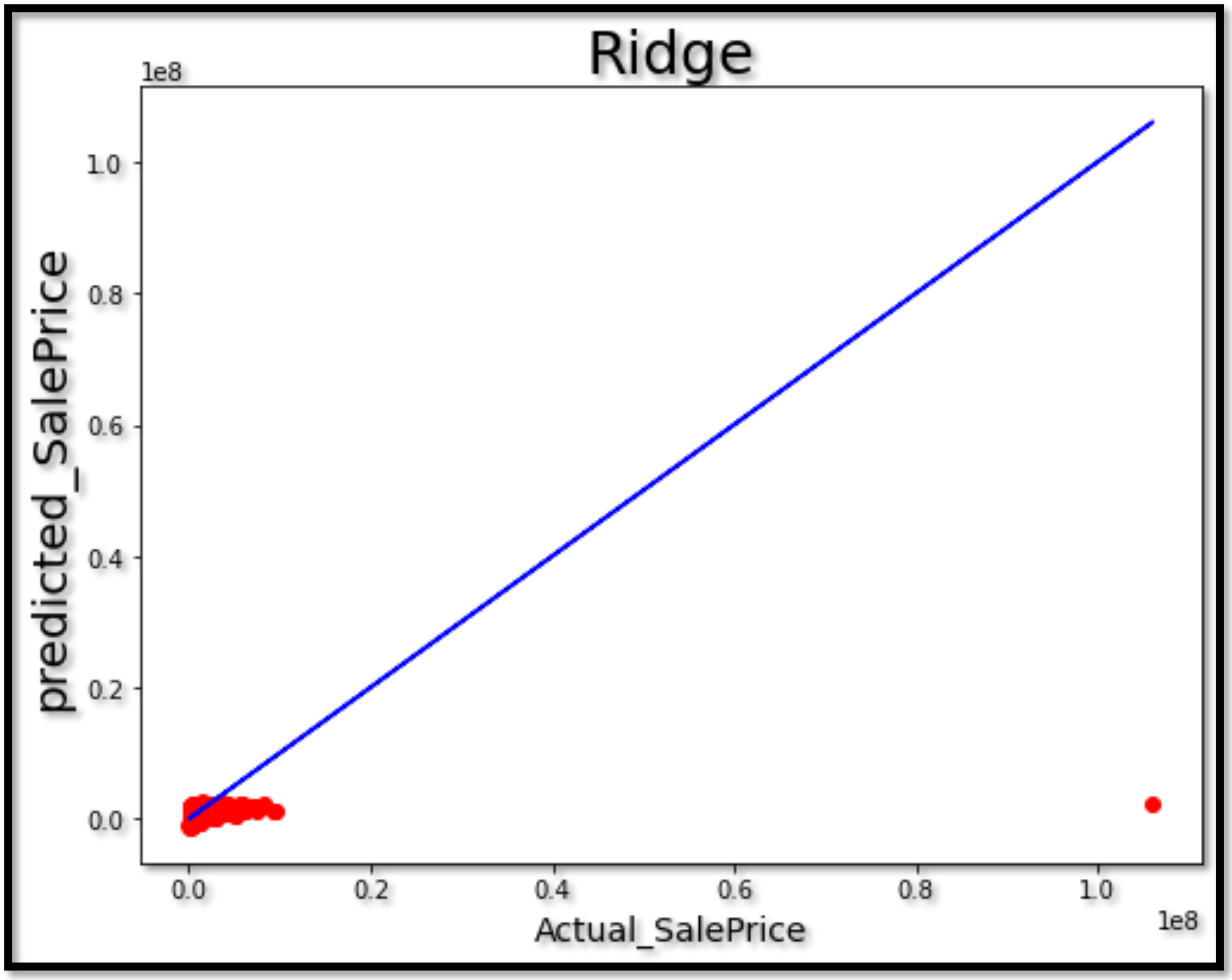
The red points are the predicted price and the blue line is the actual sale price

Lasso In Regularization method Lasso the r2_score was 03.5% and the scatter plot was



The red points are the predicted price and the blue line is the actual sale price

Ridge In Ridge the r2_score was 03.5 % and scatter plot was like



The red points are the predicted price and the blue line is the actual sale price

- **Visualizations**

Mention all the plots made along with their pictures and what were the inferences and observations obtained from those. Describe them in detail. If different platforms were used, mention that as well.

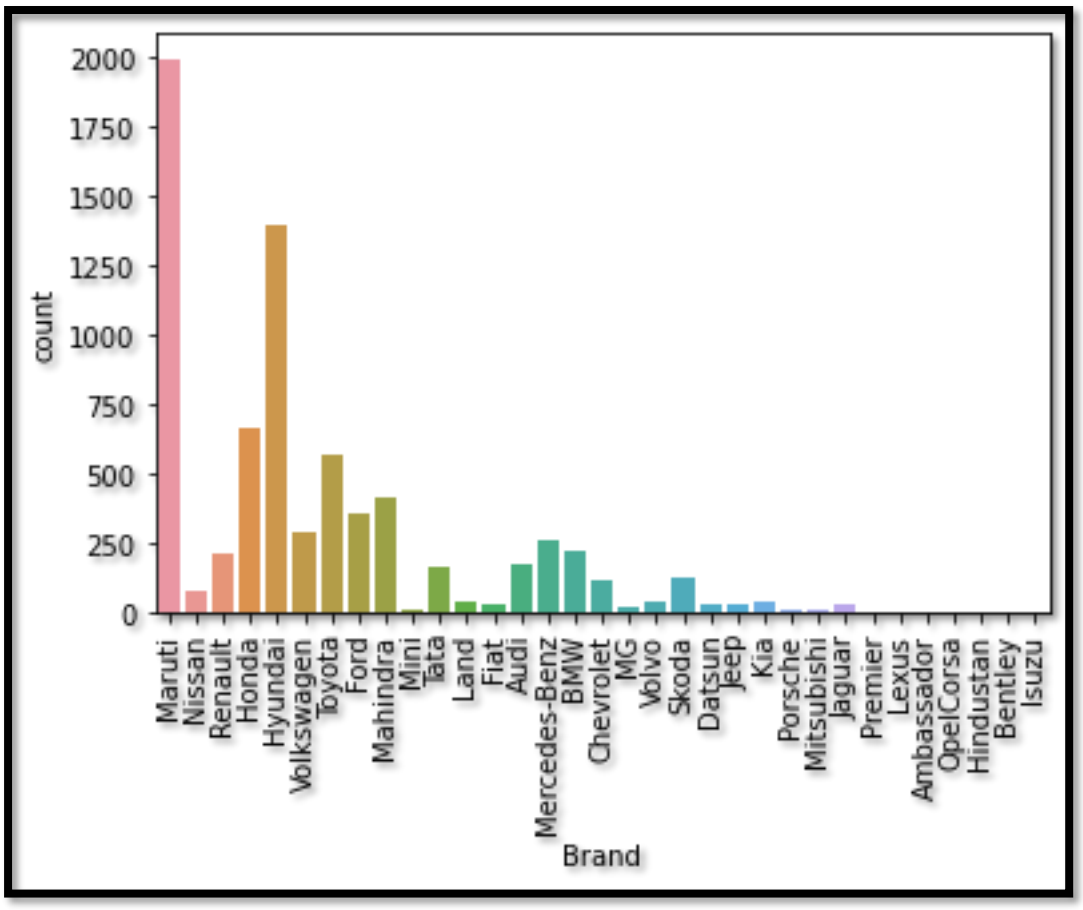
&

- **Interpretation of the Results**

Give a summary of what results were interpreted from the visualizations, preprocessing and modelling.

1) Countplot

i) Brand

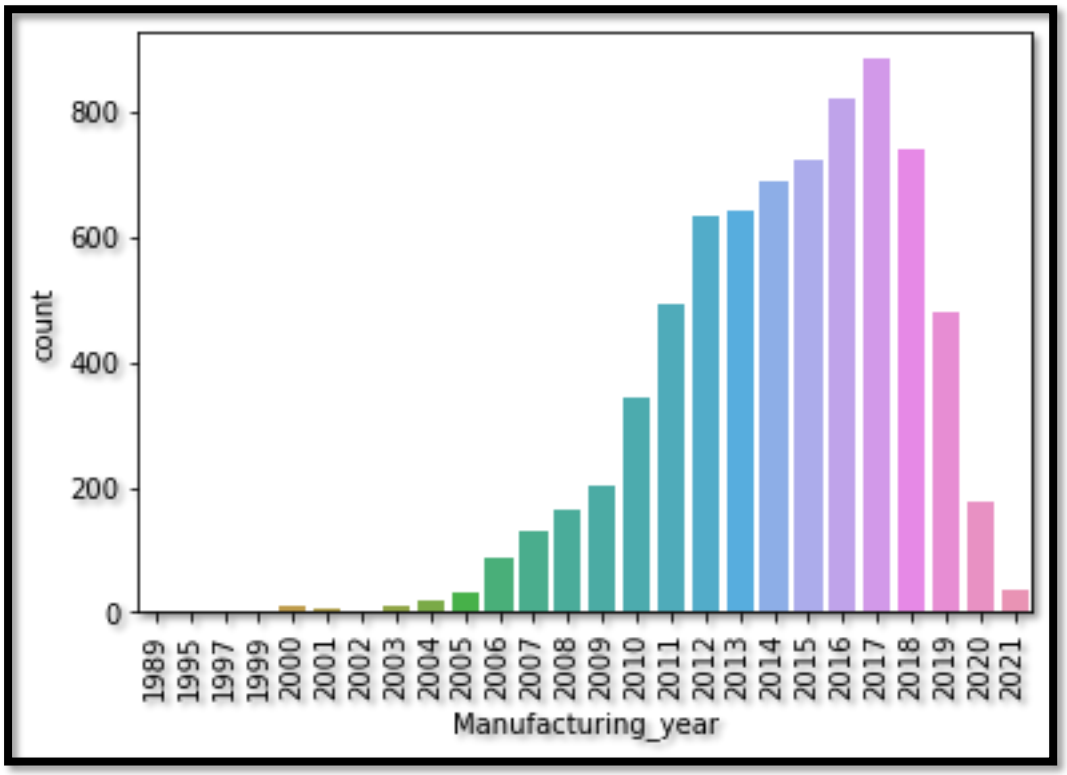


Maruti and Hyundai company have higghest number of cars for reselling and Brands like Hindustan ,Premier ,OpelCorsa ,Ambassador have only 1 car for reselling And other brands and counts were as follows

Brands – Counts

Maruti - 1995
Hyundai -1401
Honda -670
Toyota -567
Mahindra- 419
Ford -356
Volkswagen -285
Mercedes-Benz -264
BMW - 218
Renault - 216
Audi - 171
Tata -168
Skoda -123
Chevrolet -112
Nissan -78
Land -40
Kia -35
Volvo -35
Jaguar - 34
Datsun - 33
Fiat - 29
Jeep - 25
MG - 22
Mitsubishi - 14
Mini -13
Porsche - 13
Bentley - 4
Lexus - 3
Isuzu - 2
Hindustan - 1
Ambassador - 1
Premier - 1
OpelCorsa - 1

ii) Manufacturing Year



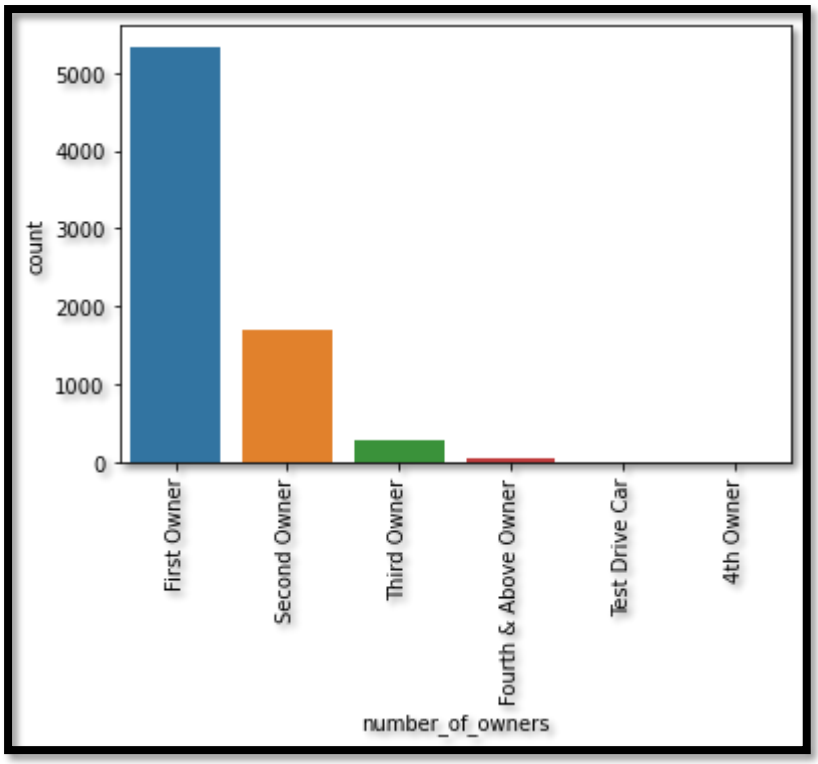
recently built cars of year like 2017, 2016 have highest numbers of car for reselling and old models or old cars have less number of cars for reselling
and other counts were as follows
year- count
2017 - 885
2016 - 824
2018 - 742
2015 - 726
2014 - 691
2013 - 643
2012 - 634
2011 - 493
2019 - 482
2010 - 344
2009 - 204

2020 - 178
2008 - 163
2007 - 129
2006 - 86
2021 - 35
2005 - 33
2004 - 20
2003 - 12
2000 - 10
2001 - 7
1997 - 3
1999 - 2

1989 - 1
1995 - 1
2002 - 1

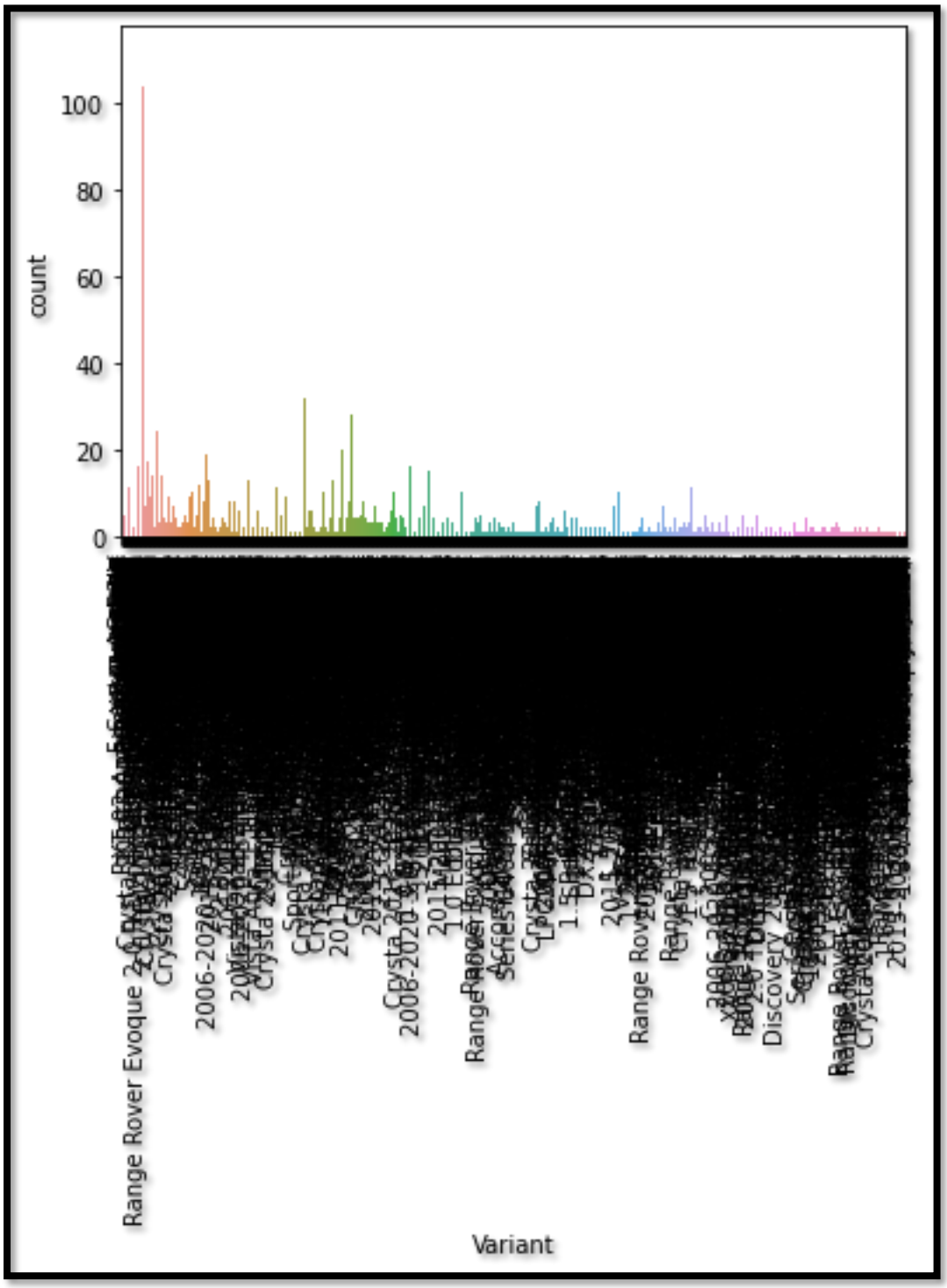
iii) No_of_owners

5335 cars that are offered for reselling are used by only one owner, 1685 cars are of second owner, 272 cars are offered by third owner, 50 cars are offered by fourth and above owner 6 cars are of test Drive and only one car is shared by 4th owner



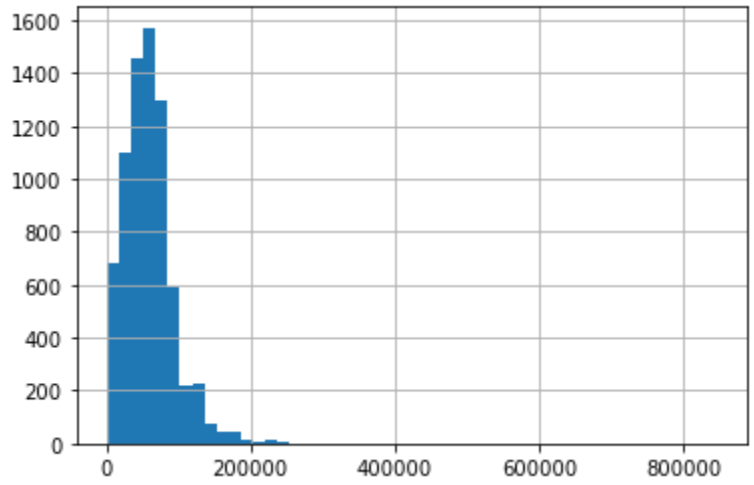
iv) Variant:

There are huge number of variant so we are getting the figure as follow



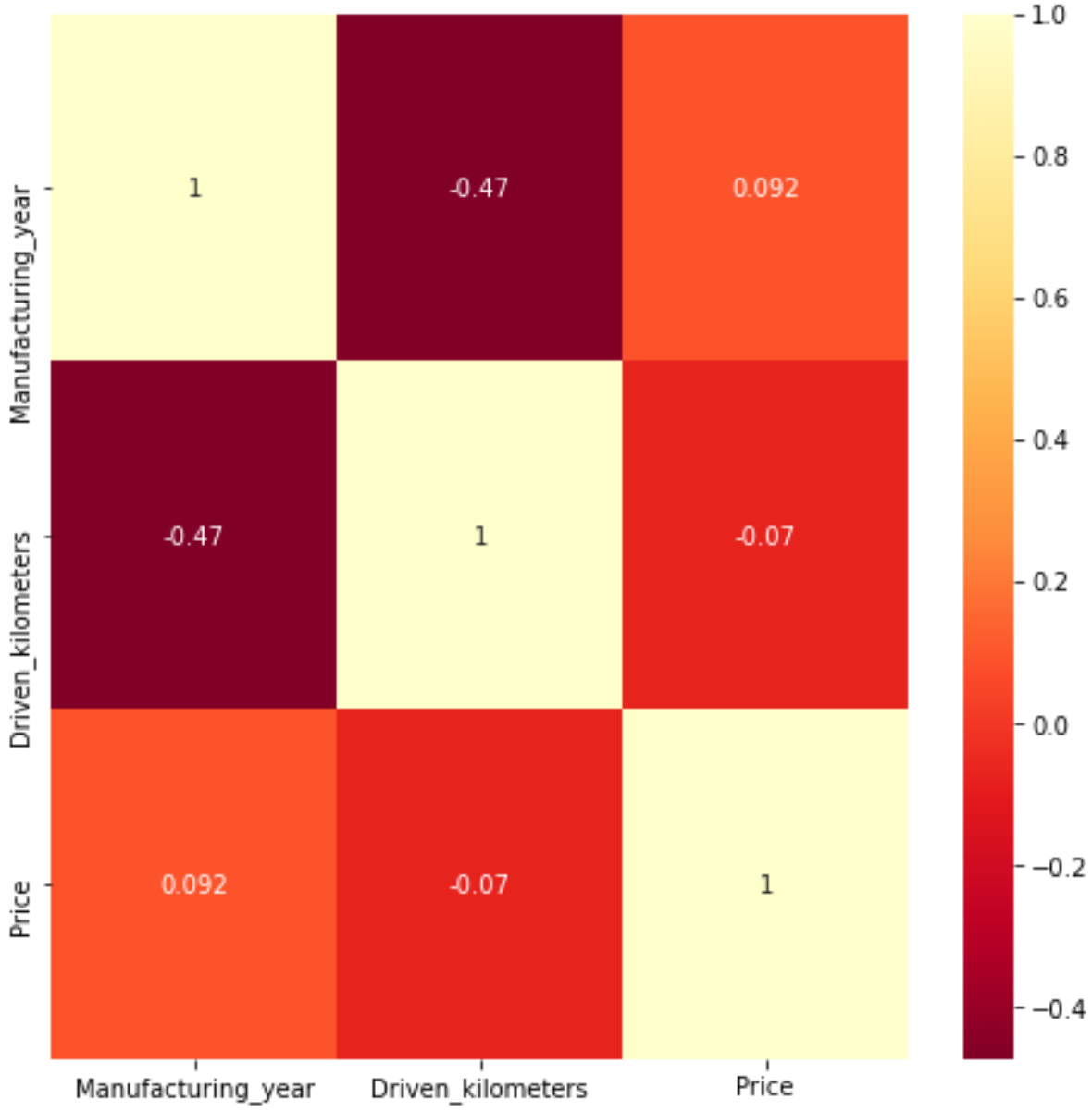
2) Histogram

i)Driven_kilometers



3) Heatmap

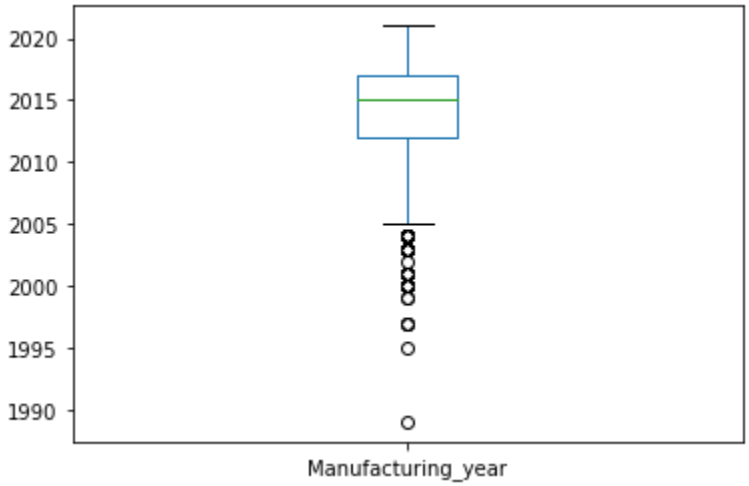
i)correlation



- Manufacturing_year has positive correlation with price
- driven_kilometers have negative correlation with price

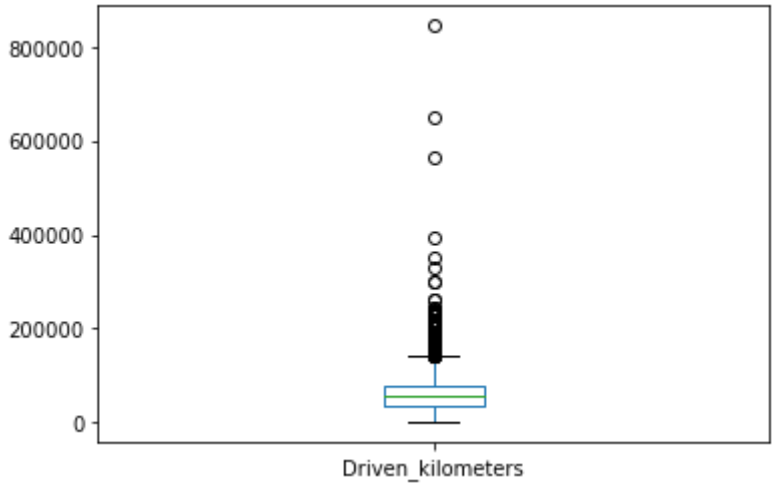
4) Boxplot

i) Manufacturing_year



There may be possibility that the old cars are offered for sale and some models of cars can be old so we will not remove this outlier, as we aim to make dynamic and generalized model so for that our model should know about the old cars their sale value so not removing the outliers

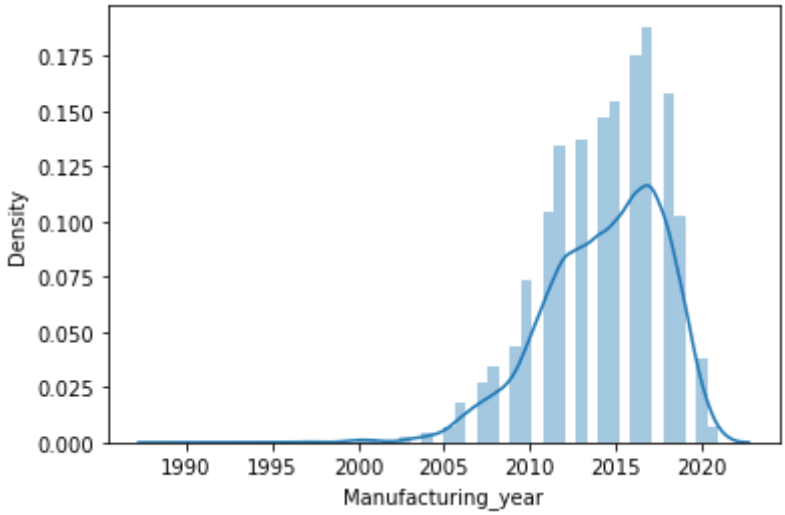
ii) Driven_kilometer



As driven_kilometer is showing some outlier we need to cure it to some extent. we will take iqr * 3 for curing outlier becausae there may be possibility that old model cars have drived alot so all the numbers that are greater than IQR * 3 will be considered as outlier and replaced with IQR * 3

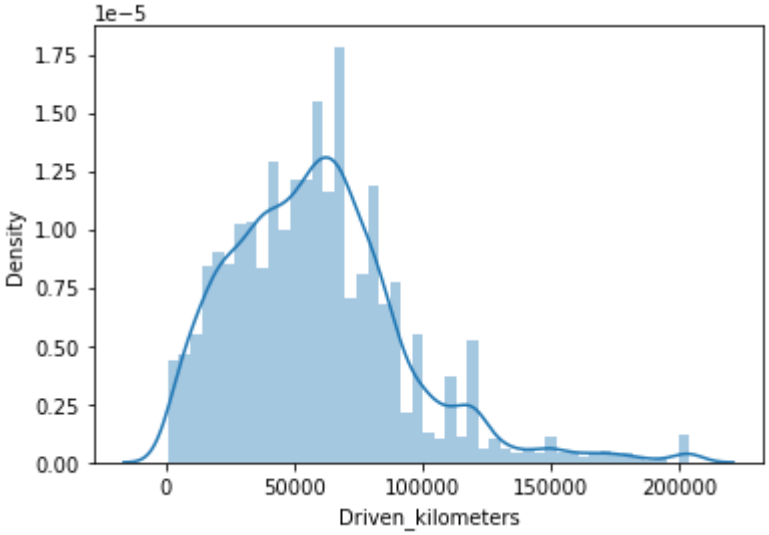
5) Distplot

i) Manufacturing_year



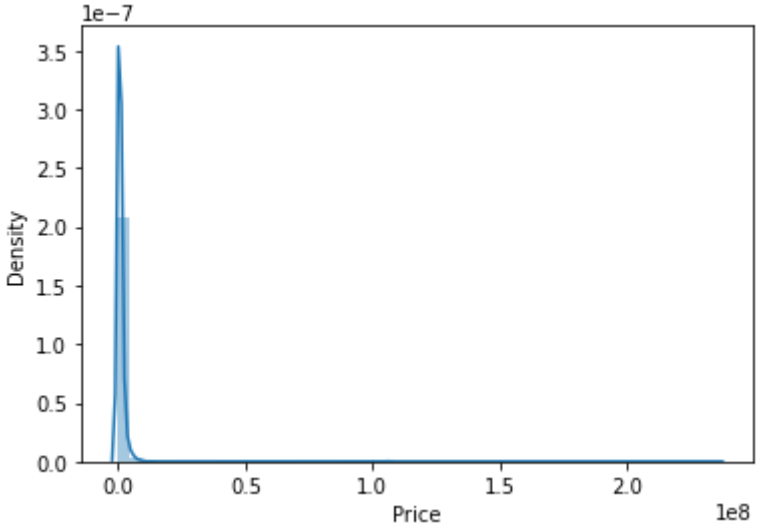
Here the data is showing little left skewness because of the manufacturing year varying from 1989 to 2021,

ii) Driven_kilometers



In Driven_kilometers column the column is showing little skewness at right side and it is possible because as the cars are pretty old too they might have run so far kilometres.

iii) Price



As price column is target column ,its skewness cannot be cured.

CONCLUSION

- **Key Findings and Conclusions of the Study**

  Describe the key findings, inferences, observations from the whole problem.

  &

- **Learning Outcomes of the Study in respect of Data Science**

  List down your learnings obtained about the power of visualization, data cleaning and various algorithms used. You can describe which algorithm works best in which situation and what challenges you faced while working on this project and how did you overcome that.

From the whole problem I have learned many things like

The things that I learned in this method was

About Visualization

1) Jointplot gives the best visualization and relation between the independent and dependent feature , it perfectly shows the regression is present or not , outliers and also the distplot at the side. Using jointplot we performed both Univariate and Bivariate analysis simultaneously, like we got complete information about the dependent variable and relation with independent variable
2) Catplot gives a result better than countplot because countplot only shows the count (frequency) of a categorical variable, but catplot shows that to what extent the particular category of a column has touched the dependent variable

Data Cleaning

Data Cleaning involves many process like Handling Null Values, Removing Outliers ,Removing Skewness, Handling Categorical Variables, Normalization of Data in data cleaning I learned

1) when there are many outliers zscore removes almost all the columns so treating only the columns that have outliers works best as we can bring the outlier to a certain upper bridge
2) MinMaxScaler works well when the data is positive
3) Applying np.log to remove skewness on the columns where mean is far greater gives the NAN as an output.

Algorithms used

1) The things that were observed was when we finally built the model , the Linear Regression Model gave the r2_Score of 88% but the model was overfitted to the some extent and it came to know when cross validation of the Linear Algorithm was done. Because the cv-score of Linear Regression was 88 %.
2) The Model RandomForestRegressor worked better as compared to the Linear Regression it gave the r2_score of 91% and the model was not so overfitted , because the cross_val_score was 88%.
3) When we performed the Regularization then we found that Lasso and Ridge were giving the same result , MSE and MAE were there so the red points that is predicted points were somewhat away from the actual line. So to cover that Hyperparameter tuning was done on RandomForestRegression. And as a result the model showed less error and more perfect model.

- **Limitations of this work and Scope for Future Work**

What are the limitations of this solution provided, the future scope? What all steps/techniques can be followed to further extend this study and improve the results.

Limitations :-

Well there were no such **limitations** found but the only thing that affected the model was that we cannot remove the outliers or skewness of the dependent column.

Future Scope:-

The model can be used by all the companies that deal in the second hand vehicles and the model can be used by various websites who deal in second-hand products or C2C (Customer to Customer) business.