FLIP ROBO

Project Report On Car Price Prediction

Submitted by

Vivek Kumar

Internship Batch 33

ACKNOWLEDGMENT

I would like to express my sincere thanks and gratitude to my SME Mr. Shwetank Mishra as well as the “FlipRobo Technologies” team for letting me work on the “Used Car Price Prediction” project. Their suggestions and directions have helped me in the completion of this project successfully. This project also helped me in doing lots of research wherein I came to know about so many new things.

TABLE OF CONTENTS:

1. INTRODUCTION

Predicting the price of used cars is an important and interesting problem. Predicting the resale value of a car is not a simple task. It is trite knowledge that the value of used cars depends on several factors. The most important ones are usually the age of the car, its make (model), the origin of the car (the original location of the manufacturer), its mileage (the number of kilometers it has run), and its horsepower (amount of power that an engine produces). Due to rising fuel prices, fuel economy is also of prime importance. Unfortunately, in practice, most people do not know exactly how much fuel their car consumes for each km driven. Other factors such as the type of fuel it uses, the interior style, the braking system, acceleration, engine displacement, the volume of its cylinders (measured in cc), its size, number of doors, paint color, the weight of the car, consumer reviews, prestigious awards won by the car manufacturer, its physical state, whether it is a sports car, whether it has cruise control, whether it is automatic or manual transmission, whether it belonged to an individual or a company and other options such as air conditioner, sound system, power steering, cosmic wheels, GPS navigator all may influence the price as well. Some special factors buyers attach importance to are the location of previous owners, and whether the car had been involved in serious accidents. The look and feel of the car certainly contribute a lot to the price. As we can see, the price depends on a large number of factors. Unfortunately, information about all these factors is not always available and the buyer must decide to purchase at a certain price based on a few factors only. In this work, we have considered only a small subset of the more important factors.

Business Problem Framing

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

Business goal: The main aim of this project is to predict the price of a used car based on various features. Machine Learning is a field of technology development with immense abilities and applications in automating tasks. So, we will deploy an ML model for car selling price prediction and analysis. This kind of system becomes handy for many people. This model will provide the approximate selling price for the car based on different features like fuel type, transmission, price, weight, running inkms, engine displacement, milage, etc, and this model will help the client to understand the price of the used cars.

Conceptual Background of the Domain Problem

Car Price Prediction is an interesting machine learning problem as many factors influence the price of a car in the second-hand market. In many developed countries, it is common to lease a car rather than buy it outright. A lease is a binding contract between a buyer and a seller (or a third party – usually a bank, insurance firm or, other financial institutions) in which the buyer must pay fixed instalments for a pre-defined number of months/years to the seller/financer. After the lease period is over, the buyer has the possibility to buy the car at its residual value, i.e., its expected resale value. Thus, it is of commercial interest to seller/financers to be able to predict the salvage value (residual value) of cars with accuracy. If the residual value is under- estimated by the seller/financer at the beginning, the instalments will be higher for the clients who will certainly then opt for another seller/financer. If the residual value is over-estimated, the instalments will be lower for the clients but then the seller/financer may have much difficulty at selling these high-priced used cars at this over-estimated residual value. Thus, we can see that estimating the price of used cars is of very high commercial importance as well.

Here we are trying to help the client works with small traders, who sell used cars to understand the price of the used cars by deploying machine learning models. These models would help the client/sellers to understand the used car market and accordingly they would be able to sell the used car in the market.

Review of Literature

Literature review covers relevant literature with the aim of gaining insight into the factors that are important to predict the price of used cars in the market. In this study, we discuss various applications and methods which inspired us to build our supervised ML techniques to predict the price of used cars in different locations. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of data information by doing web scraping from www.cardekho.com website which is a web platform where seller can sell their used car.

This project is more about data exploration, feature engineering and pre-processing that can be done on this data. Since we scrape huge amount of data that includes more car related features, we can do better data exploration and derive some interesting features using the available columns. Different techniques like ensemble techniques, k-nearest neighbours, and decision trees have been used to make the predictions.

The goal of this project is to build an application which can predict the car prices with the help of other features. In the long term, this would allow people to better explain and reviewing their purchase with each other in this increasing digital world.

Motivation for the Problem Undertaken

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, engine displacement, running, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately.

So, the main aim is to use machine learning algorithms to develop models for predicting used car prices.

- To build a supervised machine learning model for forecasting value of a vehicle based on multiple attributes.
- The system that is being built must be feature based i.e., feature wise prediction must be possible.
- Providing graphical comparisons to provide a better view

2. ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem:

We need to develop an efficient and effective Machine Learning model which predicts the price of a used cars. So, “Car_Price” is our target variable which is continuous in nature. Clearly it is a Regression problem where we need to use regression algorithms to predict the results.

This project is done on two phases:

- **Data Collection Phase:** I have done web scraping to collect the data of used cars from the well-known website www.cardekho.com where I found more features of cars compared to other websites and I fetch data for different locations. As per the requirement of our client we need to build the model to predict the prices of these used cars.

- **Model Building Phase:** After collecting the data, I built a machine learning model. Before model building, have done all data pre-processing steps. The complete life cycle of data science that I have used in this project are as follows:
  - Data Cleaning
  - Exploratory Data Analysis
  - Data Pre-processing
  - Model Building
  - Model Evaluation
  - Selecting the best model

Data Sources and their formats

We have collected the dataset from the website www.cardekho.com which is a web platform where seller can sell their used car. The data is scrapped using Web scraping technique and the framework used is Selenium. We scrapped nearly 12600 of the data and fetched the data for different locations and collected the information of different features of the car and saved the collected data in excel format. The dimension of the dataset is 12608 rows and 20 columns including target variable “Car_Price”. The particular dataset contains both categorical and numerical data type. The data description is as follows:

Variables	Definition
Car_Name	Name of the cars with manufacturing year
Fuel_type	Type of fuel used for car engine
Running_in_kms	Car running in kms till the date
Engine_disp	Engine displacement/engine CC
Gear_transmission	Type of gear transmission used in car
Milage_in_km/ltr	Overall milage of car in Km/ltr
Seating_cap	Number of seats available in the car
color	Color of the car
Max_power	Maximum power of engine used in car in bhp
front_brake_type	Type of brake system used for front-side wheels
rear_brake_type	Type of brake system used for back-side wheels
cargo_volume	Total cubic feet of space in a car's cargo area
height	Total height of car in mm
width	Width of car in mm
length	Total length of the car in mm
Weight	Gross weight of the car in kg
Insp_score	Inspection rating out of 10
top_speed	Maximum speed limit of the car in km per hours
City_url	Url of the page of cars from a particular city/location
Car_price	Price of the car

Data Pre-processing Done

Data pre-processing is the process of converting raw data into a well- readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. I have used following pre-processing steps:

- ➢ Importing necessary libraries and loading collected dataset as a data frame.
- ➢ Used pandas to set display maximum columns and rows ensuring not to find any truncated information.
- ➢ Checked some statistical information like shape, number of unique values present, info, unique() data types etc.
- ➢ From the dataset I found some numerical features having "-" sign and string value like "null" so I replaced them with NAN values and dropped

the columns having more than 50% of “-“ sign as they were of no use for prediction.

- Done feature engineering on some features as they had some irrelevant values like kms, kmpl and replaced them by empty space.
- Extracted the features Brand, Model and Manufacturing_year from the column Car_Name and created Car_age by subtracting the Manufacturing year of car from the year 2021. Replaced string values and “-“ sign by empty space in some of the columns to get the numerical data.
- The target variable "Car_price" should be continuous data but due to some string values it was showing as object data type. So, I replaced those entries with appropriate values. Then split the column into two as price_a and price_b and stored numerical values in price_a column and string values in price_b column and after that multiplied those two columns to get exact car price in numerical format.
- The column City_url contained the urls of the cities, so I created a new column as Location by replacing the urls by specific city name.
- The columns "front_brake_type" and “rear_brake_type” were had some duplicate entries that is the entries belongs to same categories so, I replaced/grouped the same categories by appropriate values.
- Converted all the numerical continuous columns from object data type into float data type after cleaning the data and saved the cleaned data in excel file format.
- Checked for null values and treated them using imputation techniques like mean, median and mode methods. Checked the statistical summary of the dataset using describe () method. Separated both numerical and categorical columns for further process.
- Performed univariate, bivariate and multivariate analysis to visualize the data. Visualized each feature using seaborn and matplotlib libraries by plotting several categorical and numerical plots like pie plot, count plot, bar plot, reg plot, strip plot, line plot, violin plot, distribution plot, box plots and pair plot.
- Identified outliers using box plots and removed outliers in continuous numerical columns using Zscore and stored the data frame after removing outliers as “new_df”.
- Checked for skewness and removed skewness in numerical columns using power transformation method (yeo-johnson).
- Encoded the columns having object data type using Label Encoder method. Used Pearson’s correlation coefficient to check the correlation between label and features. With the help of heatmap and correlation bar

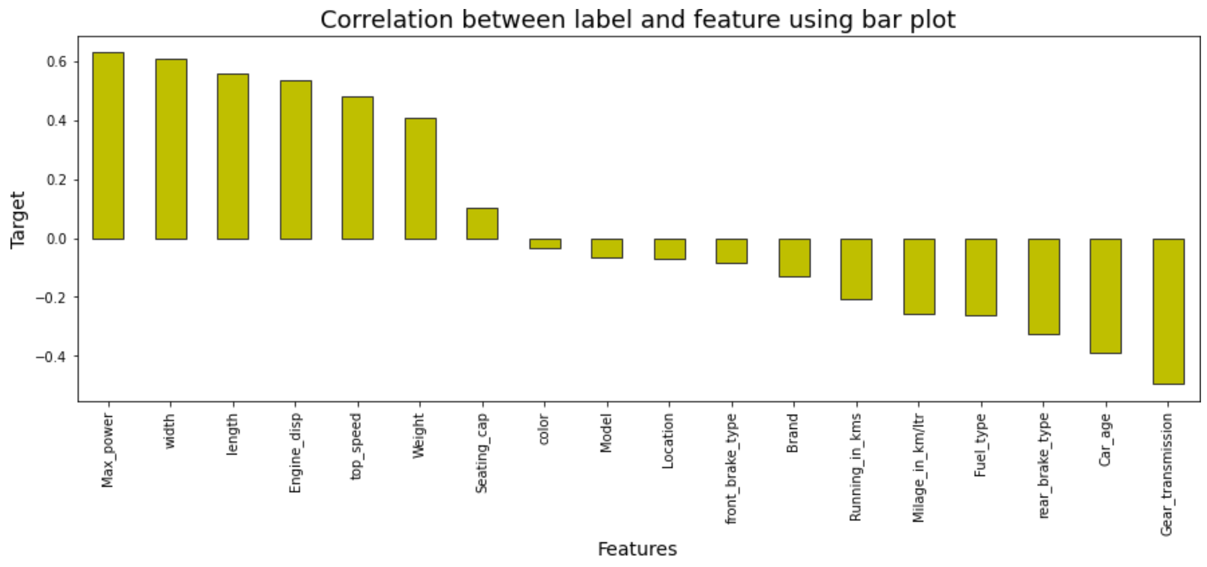
graph was able to understand the Feature vs Label relativity and insights on multicollinearity amongst the feature columns.

- Separated feature and label data and feature scaling is performed using Standard Scaler method to avoid any kind of data biasness.
- Checked Variance Inflation Factor (VIF) and dropped the column "Max_power" as it was containing VIF above 10 and got rid of multicollinearity issue.

Data Inputs- Logic- Output Relationships

The dataset consists of label and features. The features are independent and label is dependent as the values of our independent variables changes as our label varies.

- Since we had both numerical and categorical columns, I checked the distribution of skewness using dist plots for numerical features and checked the counts using count plots & pie plots for categorical features as a part of univariate analysis.
- To analyse the relation between features and label I have used many plotting techniques where I found numerical continuous variables having strong relation with label Car_Price with the help of reg plot.
- I have checked the correlation between the label and features using heat map and bar plot. Where I got both positive and negative correlation between the label and features.



Hardware & Software Requirements & Tools Used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

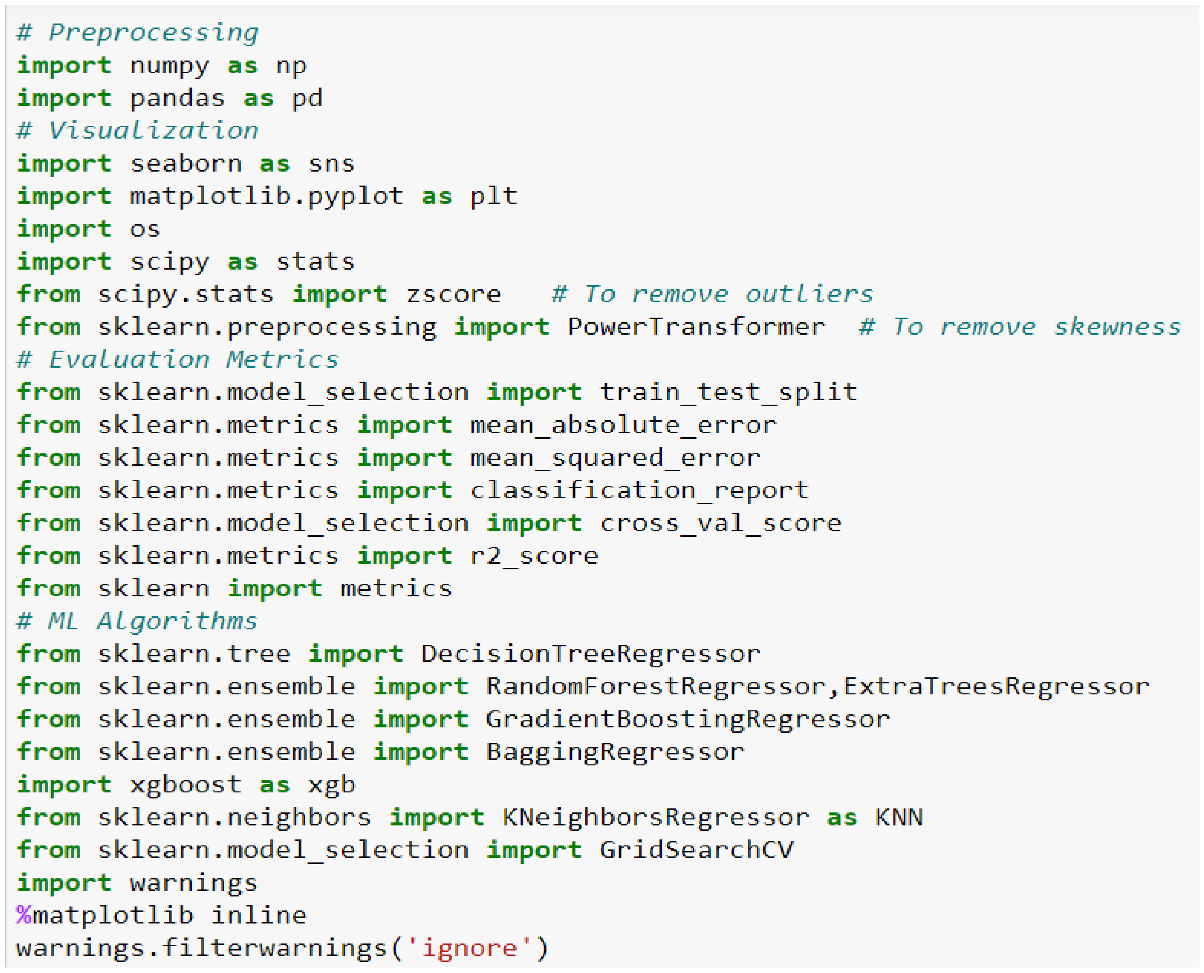
Hardware required:

- Processor: core i5 or above
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

Software required:

- Distribution: Anaconda Navigator
- Programming language: Python
- Browser based language shell: Jupyter Notebook
- Chrome: To scrape the data

Libraries required:



```python
# Preprocessing
import numpy as np
import pandas as pd
# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import os
import scipy as stats
from scipy.stats import zscore   # To remove outliers
from sklearn.preprocessing import PowerTransformer  # To remove skewness
# Evaluation Metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import r2_score
from sklearn import metrics
# ML Algorithms
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
import xgboost as xgb
from sklearn.neighbors import KNeighborsRegressor as KNN
from sklearn.model_selection import GridSearchCV
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

- ✓ **import numpy as np:** It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.
- ✓ **import pandas as pd:** Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. The data-frame term is coming from Pandas only.
- ✓ **import matplotlib.pyplot as plt:** Matplotlib and Seaborn acts as the backbone of data visualization through Python.

  Matplotlib: It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays.
- ✓ **import seaborn as sns: Seaborn** is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data.

With the above sufficient libraries, we can perform pre-processing, data cleaning and can build ML models.

MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible Problem-solving approaches (Methods):

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data also used EDA techniques and heat map to check the correlation of independent and dependent features. Treated null values using imputation methods. Removed outliers and skewness using Zscore and yeo-johnson methods respectively. Encoded data using Label Encoder. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. Checked for the best random state to be used on our Regression Machine Learning model pertaining to the feature importance details. Finally created multiple regression models along with evaluation metrics.

For this particular project we need to predict price of used cars. In this dataset, Car_Price is the target variable, which means our target column is continuous in nature so this is a regression problem. I have used many regression algorithms and predicted the car price. By doing various evaluations I have selected Extreme Gradient Boosting Regressor (XGB) as best suitable algorithm to create our final model as it is giving least difference in R2 score and cross validation score among all the algorithms used. In order to get good performance and to check whether my model getting over-fitting and under-fitting I have made use of the K-Fold cross validation by setting cv=5 and then hyper parameter tuning on best model. Then I saved my final model and loaded the same for predictions.

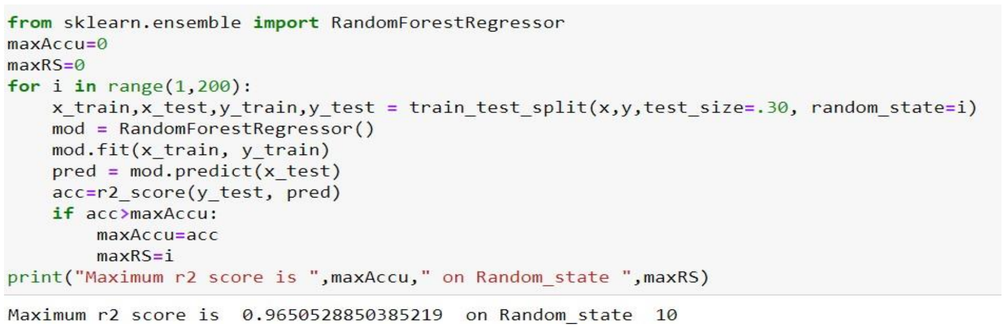
Testing of Identified Approaches (Algorithms)

Since “Car_Price” is my target variable which is continuous in nature, from this I can conclude that it is a regression type problem hence I have used following regression algorithms. After the pre-processing and data cleaning I left with 18 columns including target and with the help of feature importance bar graph I used these independent features for model building and prediction. The algorithms used on training the data are as follows:

1. Decision Tree Regressor
2. Random Forest Regressor
3. Extra Trees Regressor
4. Gradient Boosting Regressor
5. Extreme Gradient Boosting Regressor (XGB)
6. Bagging Regressor
7. KNeighbors Regressor (KNN)

Run and evaluate selected models

I have used 6 regression algorithms after choosing random state amongst 1-1000 number. I have used Random Forest Regressor to find best random state and the code is as below:



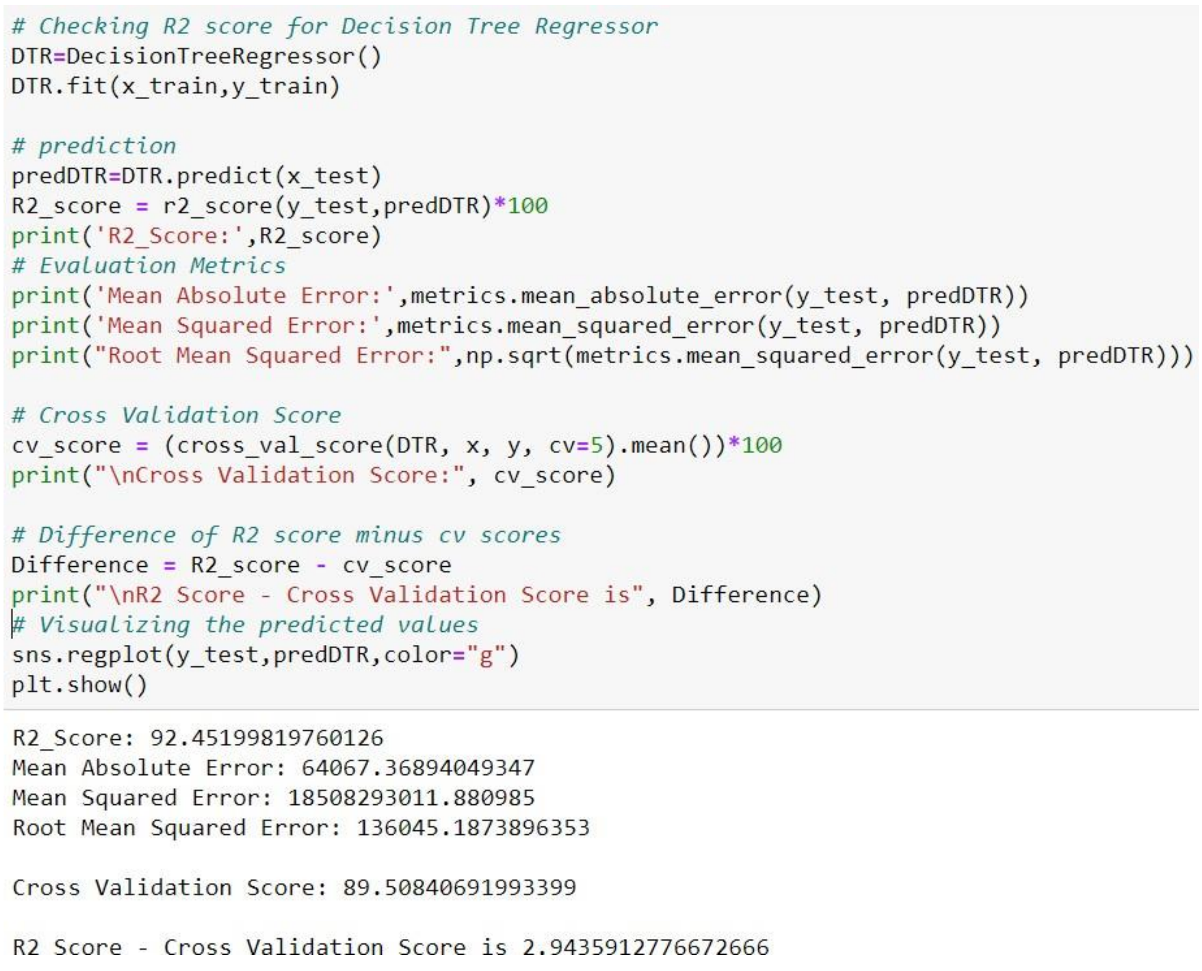
```
from sklearn.ensemble import RandomForestRegressor
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state=i)
    mod = RandomForestRegressor()
    mod.fit(x_train, y_train)
    pred = mod.predict(x_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Maximum r2 score is ",maxAccu," on Random_state ",maxRS)
```

```
Maximum r2 score is  0.9650528850385219  on Random_state  10
```

Model Building:

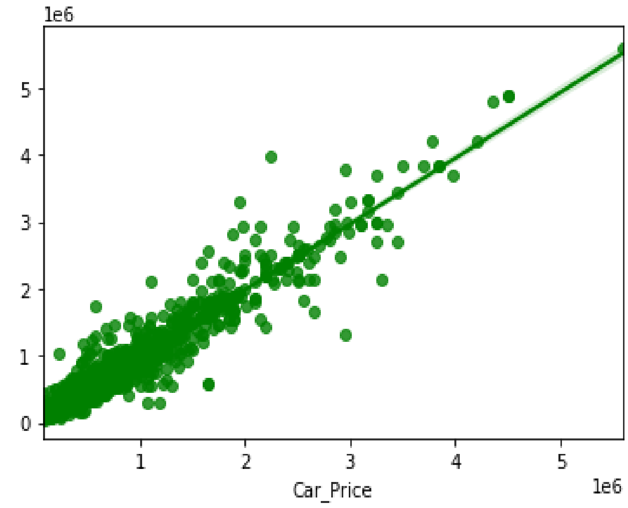
1. Decision Tree Regressor:

Decision Tree Regressor is a decision-making tool that uses a flowchart like tree structure. It observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.



```
# Checking R2 score for Decision Tree Regressor
DTR=DecisionTreeRegressor()
DTR.fit(x_train,y_train)

# prediction
predDTR=DTR.predict(x_test)
R2_score = r2_score(y_test,predDTR)*100
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predDTR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predDTR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predDTR)))

# Cross Validation Score
cv_score = (cross_val_score(DTR, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)
# Visualizing the predicted values
sns.regplot(y_test,predDTR,color="g")
plt.show()
```

```
R2_Score: 92.45199819760126
Mean Absolute Error: 64067.36894049347
Mean Squared Error: 18508293011.880985
Root Mean Squared Error: 136045.1873896353

Cross Validation Score: 89.50840691993399

R2 Score - Cross Validation Score is 2.9435912776672666
```



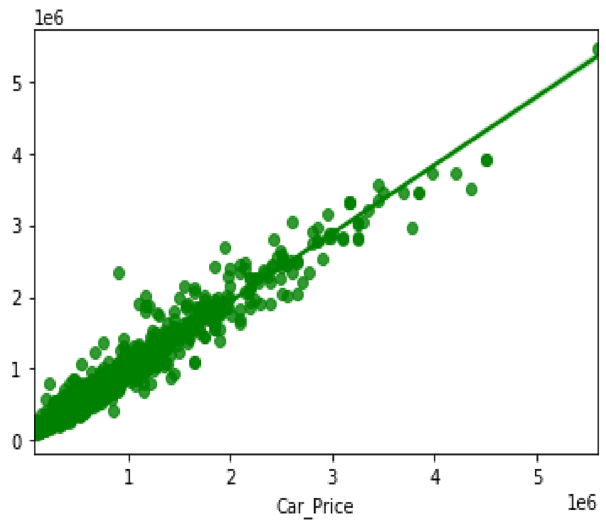
- ❖ Created Decision Tree Regressor model and checked for its evaluation metrics. The model is giving R2 score as 92.45%.
- ❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

2. Random Forest Regressor:

Random forest is an ensemble technique capable of performing both regression and classification tasks with use of multiple decision trees and a technique called Bootstrap Aggregation. It improves the predictive accuracy and control over-fitting.

```python
# Checking R2 score for Random Forest Regressor
RFR=RandomForestRegressor()
RFR.fit(x_train,y_train)

# prediction
predRFR=RFR.predict(x_test)
R2_score = r2_score(y_test,predRFR)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predRFR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predRFR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predRFR)))

# Cross Validation Score
cv_score = (cross_val_score(RFR, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predRFR,color="g")
plt.show()
```

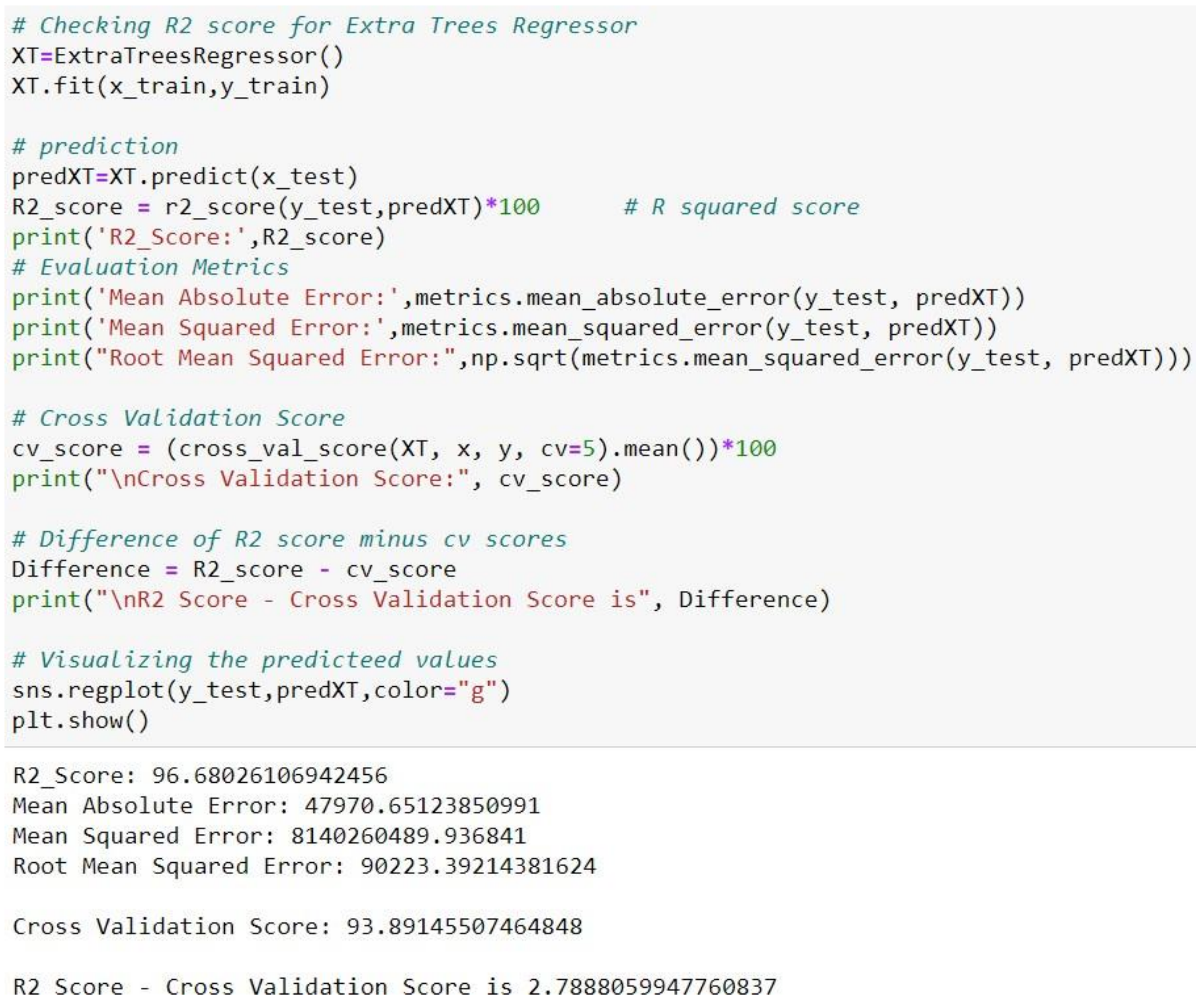
```
R2_Score: 96.05106436127106
Mean Absolute Error: 51849.249266017
Mean Squared Error: 9683100216.460907
Root Mean Squared Error: 98402.74496405528

Cross Validation Score: 93.08141777069311

R2 Score - Cross Validation Score is 2.969646590577952
```



- Created Random Forest Regressor model and checked for it's evaluation metrics. The model is giving R2 score as 96.05%.
- From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that our model has given.

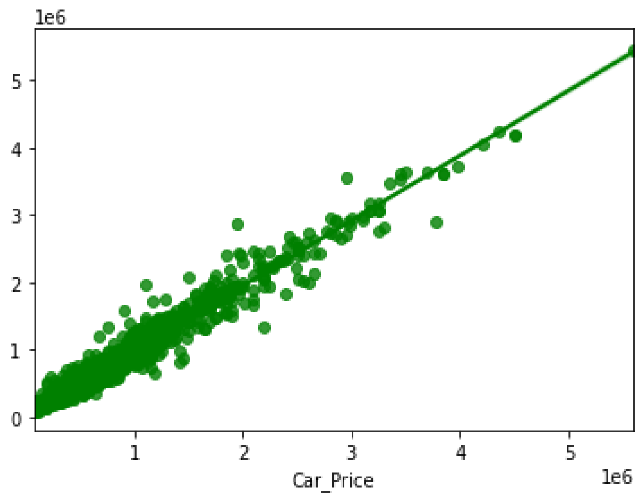
3. Extra Trees Regressor:

The **Extra Trees** implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.



```python
# Checking R2 score for Extra Trees Regressor
XT=ExtraTreesRegressor()
XT.fit(x_train,y_train)

# prediction
predXT=XT.predict(x_test)
R2_score = r2_score(y_test,predXT)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predXT))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predXT))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predXT)))

# Cross Validation Score
cv_score = (cross_val_score(XT, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predXT,color="g")
plt.show()
```

```
R2_Score: 96.68026106942456
Mean Absolute Error: 47970.65123850991
Mean Squared Error: 8140260489.936841
Root Mean Squared Error: 90223.39214381624

Cross Validation Score: 93.89145507464848

R2 Score - Cross Validation Score is 2.7888059947760837
```



❖ Created Extra Trees Regressor model and checked for its evaluation metrics. The model is giving R2 score as 96.68%.

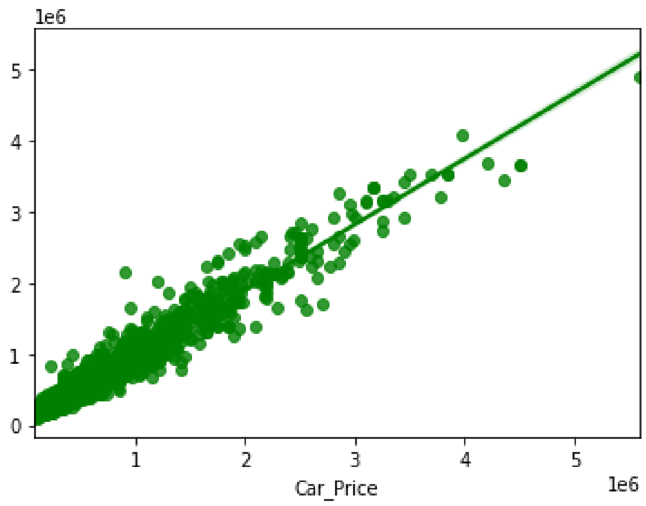
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that our model has given.

4. Gradient Boosting Regressor:

Gradient Boosting Regressor is also works for both numerical as well as categorical output variables. It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. This model was chosen to account for non-linear relationships between the features & predicted price, by splitting the data into 100 regions.

```
# Checking R2 score for GradientBoosting Regressor
GB=GradientBoostingRegressor()
GB.fit(x_train,y_train)

# prediction
predGB=GB.predict(x_test)
R2_score = r2_score(y_test,predGB)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predGB))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predGB))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predGB)))

# Cross Validation Score
cv_score = (cross_val_score(GB, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)
# Visualizing the predicteed values
sns.regplot(y_test,predGB,color="g")
plt.show()
```

```
R2_Score: 94.14109744052004
Mean Absolute Error: 74211.37262764617
Mean Squared Error: 14366489057.336084
Root Mean Squared Error: 119860.28974325102

Cross Validation Score: 91.06697781192152

R2 Score - Cross Validation Score is 3.074119628598524
```

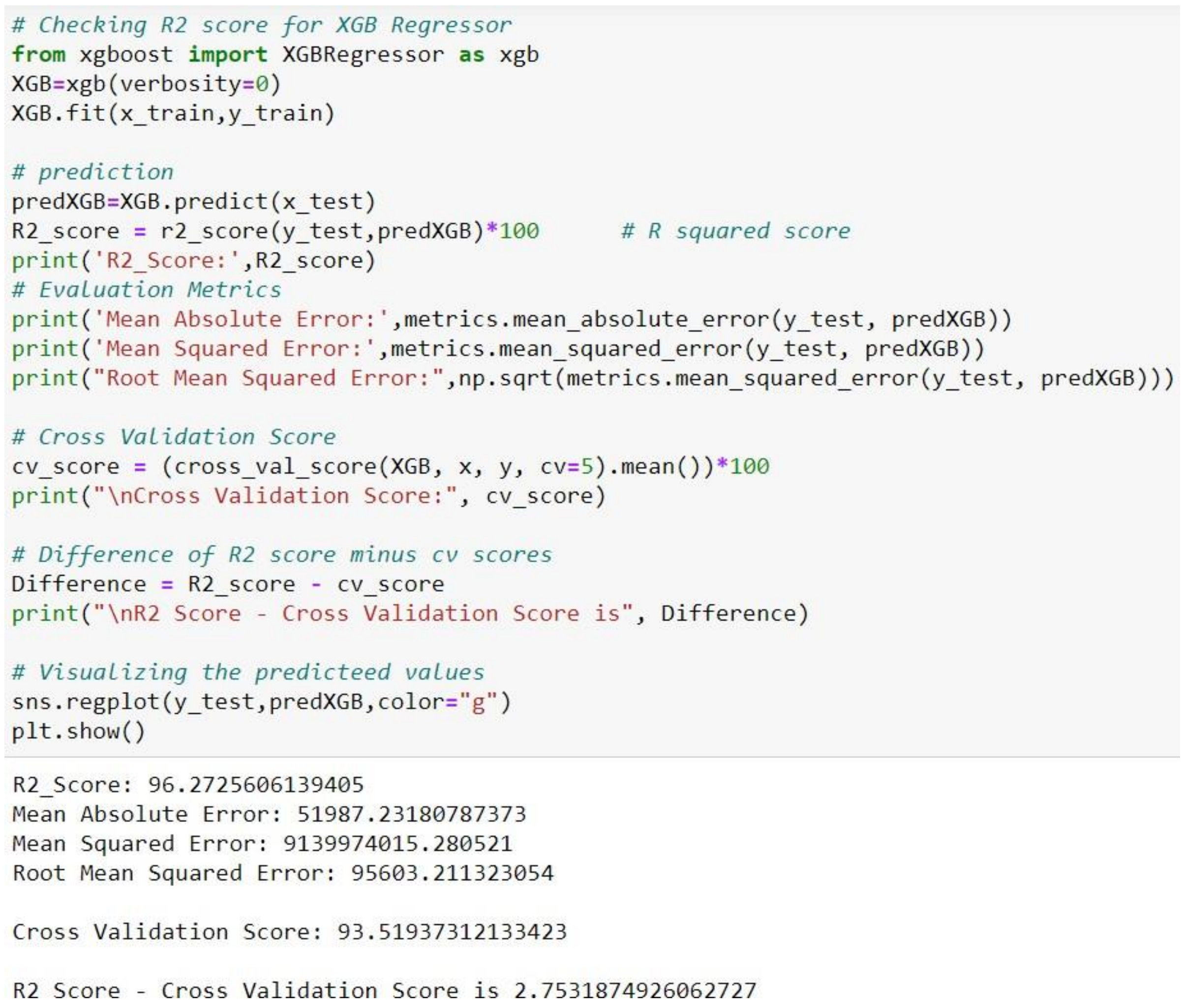


❖ Created Gradient Boosting Regressor model and checked for its evaluation metrics. The model is giving R2 score as 94.14%.

❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

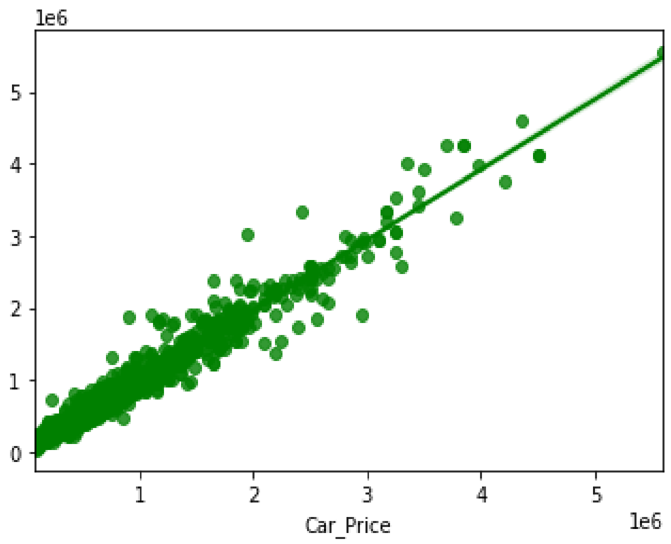
5. Extreme Gradient Boosting Regressor (XGB Regressor):

XGB Regressor is a popular supervised machine learning model and it is an implementation of Gradient Boosting trees algorithm. It is best known to provide better solutions than other machine learning algorithms.



```
# Checking R2 score for XGB Regressor
from xgboost import XGBRegressor as xgb
XGB=xgb(verbosity=0)
XGB.fit(x_train,y_train)

# prediction
predXGB=XGB.predict(x_test)
R2_score = r2_score(y_test,predXGB)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predXGB))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predXGB))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predXGB)))

# Cross Validation Score
cv_score = (cross_val_score(XGB, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predXGB,color="g")
plt.show()
```

```
R2_Score: 96.2725606139405
Mean Absolute Error: 51987.23180787373
Mean Squared Error: 9139974015.280521
Root Mean Squared Error: 95603.211323054

Cross Validation Score: 93.51937312133423

R2 Score - Cross Validation Score is 2.7531874926062727
```



❖ Created XGB Regressor model and checked for its evaluation metrics. The model is giving R2 score as 96.27%.

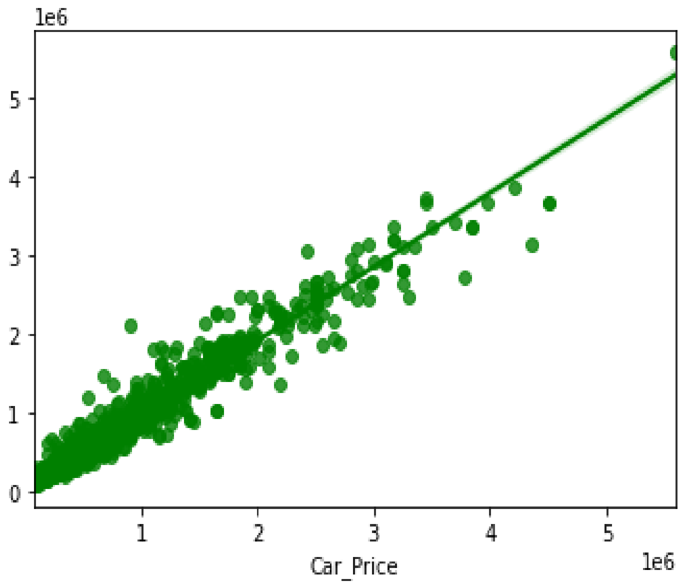
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

6. Bagging Regressor:

A Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions to form a final prediction.

```
# Checking R2 score for BaggingRegressor
BR=BaggingRegressor()
BR.fit(x_train,y_train)

# prediction
predBR=BR.predict(x_test)
R2_score = r2_score(y_test,predBR)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predBR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predBR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predBR)))
# Cross Validation Score
cv_score = (cross_val_score(BR, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predBR,color="g")
plt.show()
```

```
R2_Score: 95.30298308762171
Mean Absolute Error: 55890.32746008708
Mean Squared Error: 11517454231.188765
Root Mean Squared Error: 107319.40286448096

Cross Validation Score: 92.48114680639328

R2 Score - Cross Validation Score is 2.8218362812284283
```

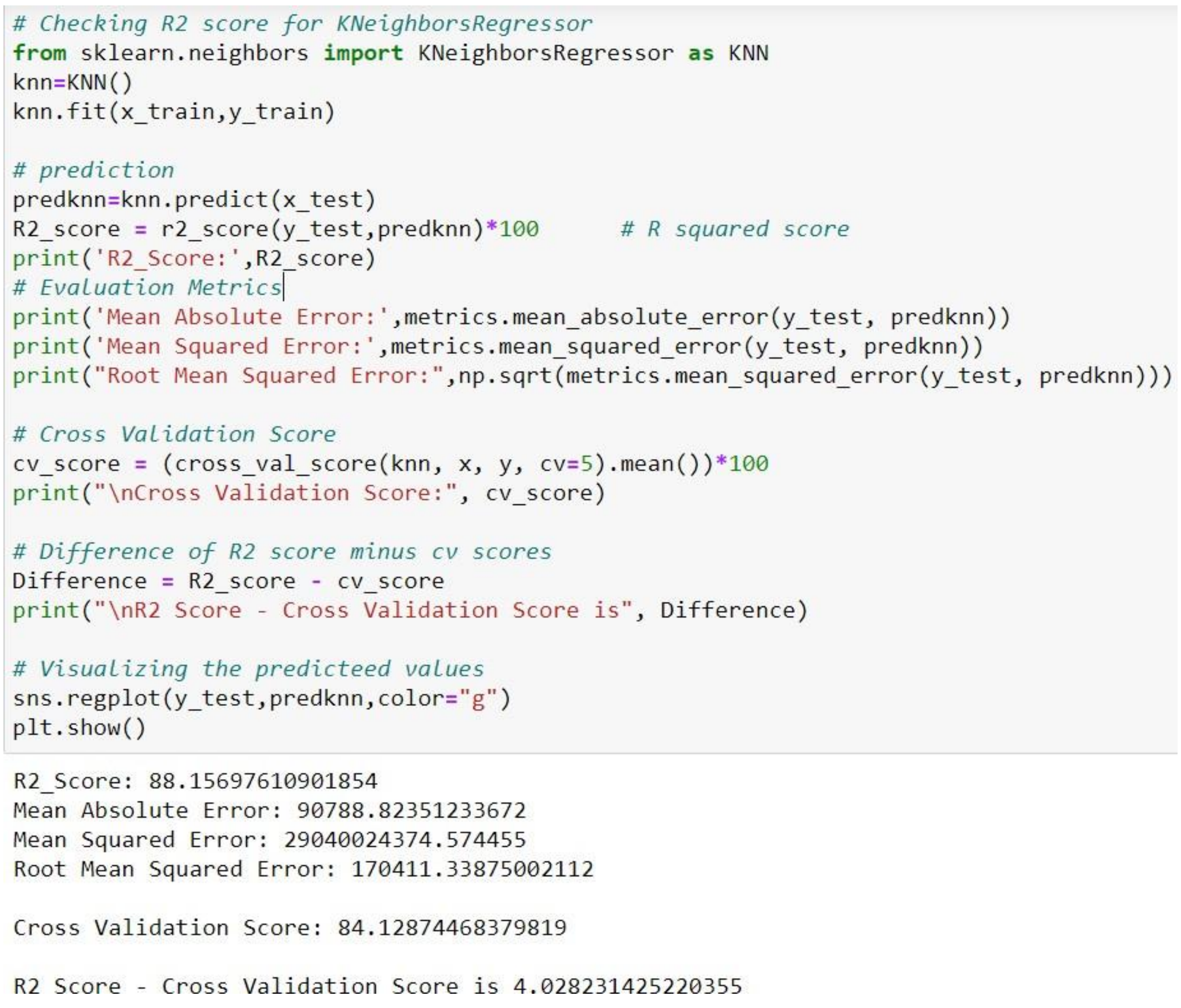


❖ Created Bagging Regressor model and checked for its evaluation metrics. The model is giving R2 score as 95.30%.

❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

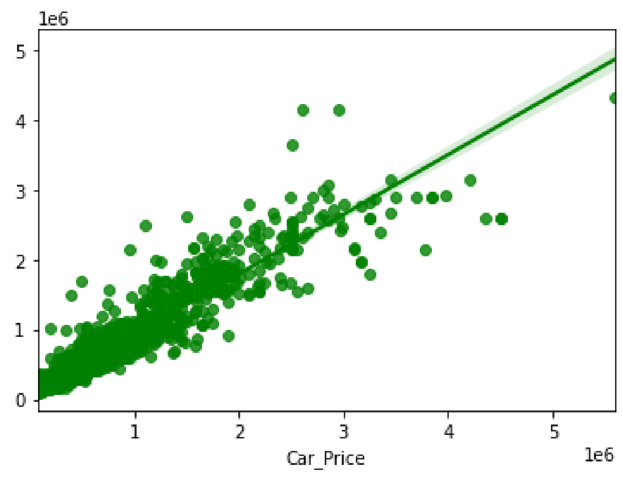
7. K Nearest Neighbors Regressor:

The **K nearest neighbors** is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure and it 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.



```
# Checking R2 score for KNeighborsRegressor
from sklearn.neighbors import KNeighborsRegressor as KNN
knn=KNN()
knn.fit(x_train,y_train)

# prediction
predknn=knn.predict(x_test)
R2_score = r2_score(y_test,predknn)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predknn))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predknn))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predknn)))

# Cross Validation Score
cv_score = (cross_val_score(knn, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predknn,color="g")
plt.show()
```

```
R2_Score: 88.15697610901854
Mean Absolute Error: 90788.82351233672
Mean Squared Error: 29040024374.574455
Root Mean Squared Error: 170411.33875002112

Cross Validation Score: 84.12874468379819

R2 Score - Cross Validation Score is 4.028231425220355
```



❖ Created KNN Regressor model and checked for its evaluation metrics. The model is giving R2 score as 88.15%.

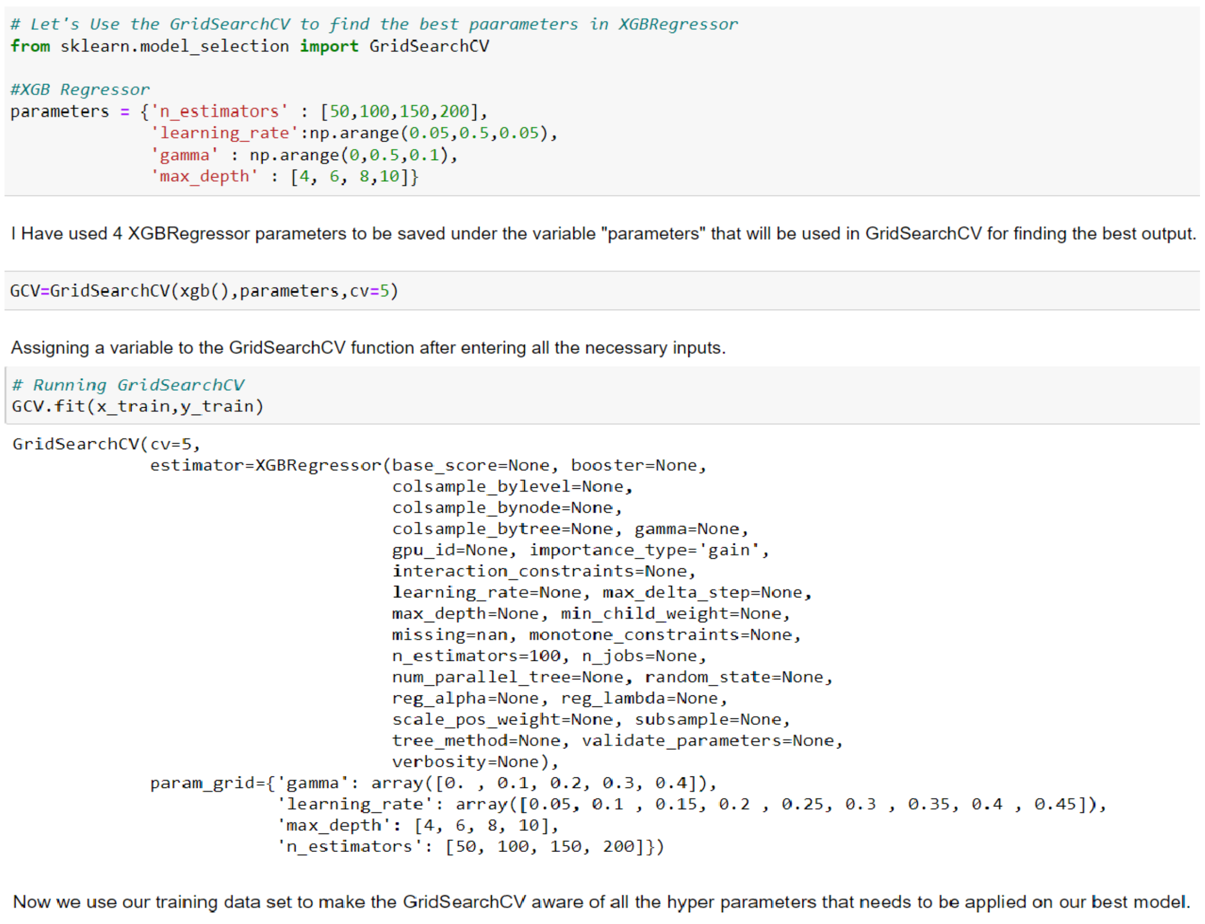
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

Model Selection:

Model	R2 Score	Cross_Validation_Score	Difference
Decision Tree Regressor	92.451	89.508	2.943
Random Forest Regressor	96.051	93.081	2.969
Extra Trees Regressor	96.680	93.891	2.788
Gradient Boosting Regressor	94.141	91.066	3.074
XGB Regressor	96.272	93.519	2.753
Bagging Regressor	95.302	92.481	2.821
K Neighbors Regressor	88.156	84.128	4.028

From the difference between R2 score and Cross Validation score, it can be seen that the XGB Regressor has least difference and low evaluation metrics compared to other models. That is XGBoosting as a regression gave the best R2 score, MAE, MSE and RMSE values. So, we can conclude that XGB Regressor as our best fitting model. Let's try to increase our model score by tuning the best model using different types of hyper parameters.

Hyper Parameter Tuning:



```
# Let's Use the GridSearchCV to find the best paarameters in XGBRegressor
from sklearn.model_selection import GridSearchCV

#XGB Regressor
parameters = {'n_estimators' : [50,100,150,200],
              'learning_rate':np.arange(0.05,0.5,0.05),
              'gamma' : np.arange(0,0.5,0.1),
              'max_depth' : [4, 6, 8,10]}
```

I Have used 4 XGBRegressor parameters to be saved under the variable "parameters" that will be used in GridSearchCV for finding the best output.

```
GCV=GridSearchCV(xgb(),parameters,cv=5)
```

Assigning a variable to the GridSearchCV function after entering all the necessary inputs.

```
# Running GridSearchCV
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5,
             estimator=XGBRegressor(base_score=None, booster=None,
                                    colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None, gamma=None,
                                    gpu_id=None, importance_type='gain',
                                    interaction_constraints=None,
                                    learning_rate=None, max_delta_step=None,
                                    max_depth=None, min_child_weight=None,
                                    missing=nan, monotone_constraints=None,
                                    n_estimators=100, n_jobs=None,
                                    num_parallel_tree=None, random_state=None,
                                    reg_alpha=None, reg_lambda=None,
                                    scale_pos_weight=None, subsample=None,
                                    tree_method=None, validate_parameters=None,
                                    verbosity=None),
             param_grid={'gamma': array([0. , 0.1, 0.2, 0.3, 0.4]),
                         'learning_rate': array([0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45]),
                         'max_depth': [4, 6, 8, 10],
                         'n_estimators': [50, 100, 150, 200]})
```

Now we use our training data set to make the GridSearchCV aware of all the hyper parameters that needs to be applied on our best model.

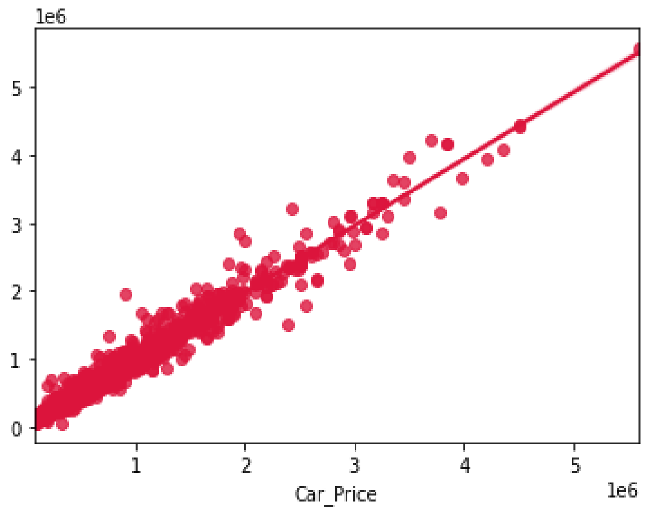
```
# Finding best parameters
GCV.best_params_
```

```
{'gamma': 0.0, 'learning_rate': 0.2, 'max_depth': 6, 'n_estimators': 200}
```

This gives us the list of parameters which will be used further in our final model creation.

```
# Creating final model
Car_price_model = xgb(gamma=0.0, learning_rate=0.2, max_depth=6, n_estimators=200)

# Prediction
Car_price_model.fit(x_train, y_train)
pred = Car_price_model.predict(x_test)
print('R2_Score:',r2_score(y_test,pred)*100)

# Metric Evaluation
print('Mean absolute error:',metrics.mean_absolute_error(y_test, pred))
print('Mean squared error:',metrics.mean_squared_error(y_test, pred))
print('Root Mean Squared error:',np.sqrt(metrics.mean_squared_error(y_test, pred)))

# Visualizing the predicted values
sns.regplot(y_test,pred,color="crimson")
plt.show()
```

```
R2_Score: 96.90005372063244
Mean absolute error: 48866.11765466256
Mean squared error: 7601311653.289676
Root Mean Squared error: 87185.50139380788
```



- ♣ I have successfully incorporated the hyper parameter tuning using best parameters of XGB Regressor by using GridSearchCV and the R2 score of the model has been increased after hyperparameter tuning and received the R2 score as 96.90% which is very good.
- ♣ From the graph we can observe how our final model is mapping. In the graph we can observe the best fit line which is our actual dataset and the dots are the predictions that our best final model has given.

Saving the final model and predicting the price of used car

```
# Saving the model using joblib library
import joblib
joblib.dump(Car_price_model,"Used_Car_Price_Prediction.pkl")
```

```
['Used_Car_Price_Prediction.pkl']
```

I am using the joblib option to save the final regression model in the form of .pkl.

Loading the saved model and predicting Used Car Price

```
# Loading the saved model
Model=joblib.load("Used_Car_Price_Prediction.pkl")

#Prediction
prediction = Model.predict(x_test)
prediction
```

```
array([1325754.  , 1542465.  ,  598151.4 , ...,  589439.3 ,  298487.53,
        338071.3 ], dtype=float32)
```

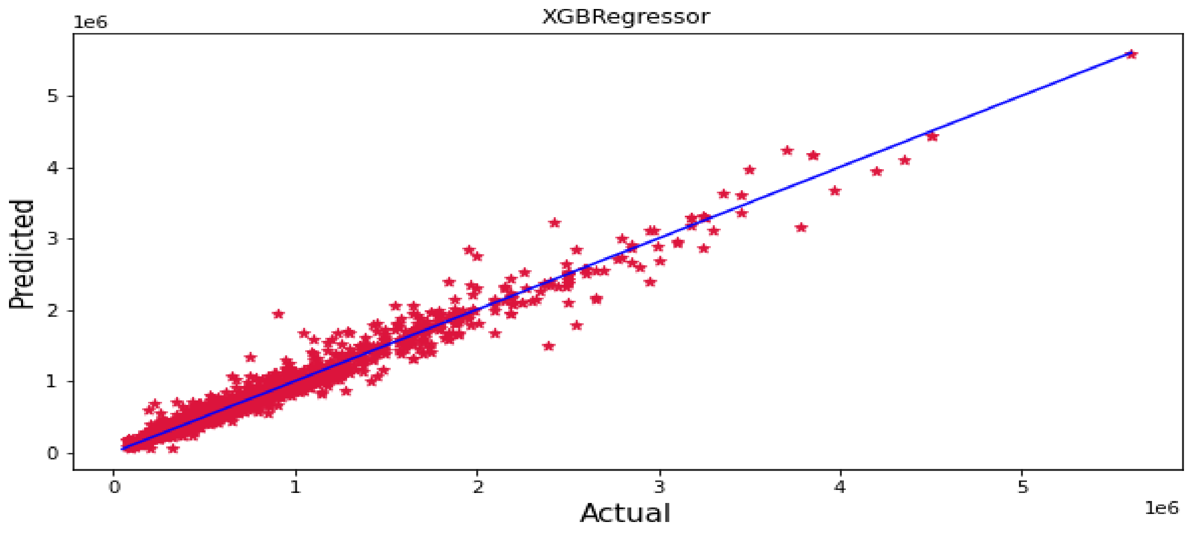
These are the predicted price of the used cars.

Creating DataFrame for the predicted values

```
Predicted_Used_Car_Price = pd.DataFrame([Model.predict(x_test)[:],y_test[:]],index=["Predicted","Original"])
Predicted_Used_Car_Price
```

	0	1	2	3	4	5	6	7	8	9	10	11	
Predicted	1325754.0	1542465.0	598151.375	577281.875	1746357.875	1110489.5	1613285.375	642901.5	678358.875	728778.125	553981.0	567225.8125	184260
Original	1651000.0	1575000.0	651000.000	715000.000	1675000.000	1099000.0	1500000.000	625000.0	660000.000	735000.000	485000.0	541000.0000	172500

Using regression model, we have got the predicted price of the used cars. From the above output we can observe that predicted values are almost near to the actual values.

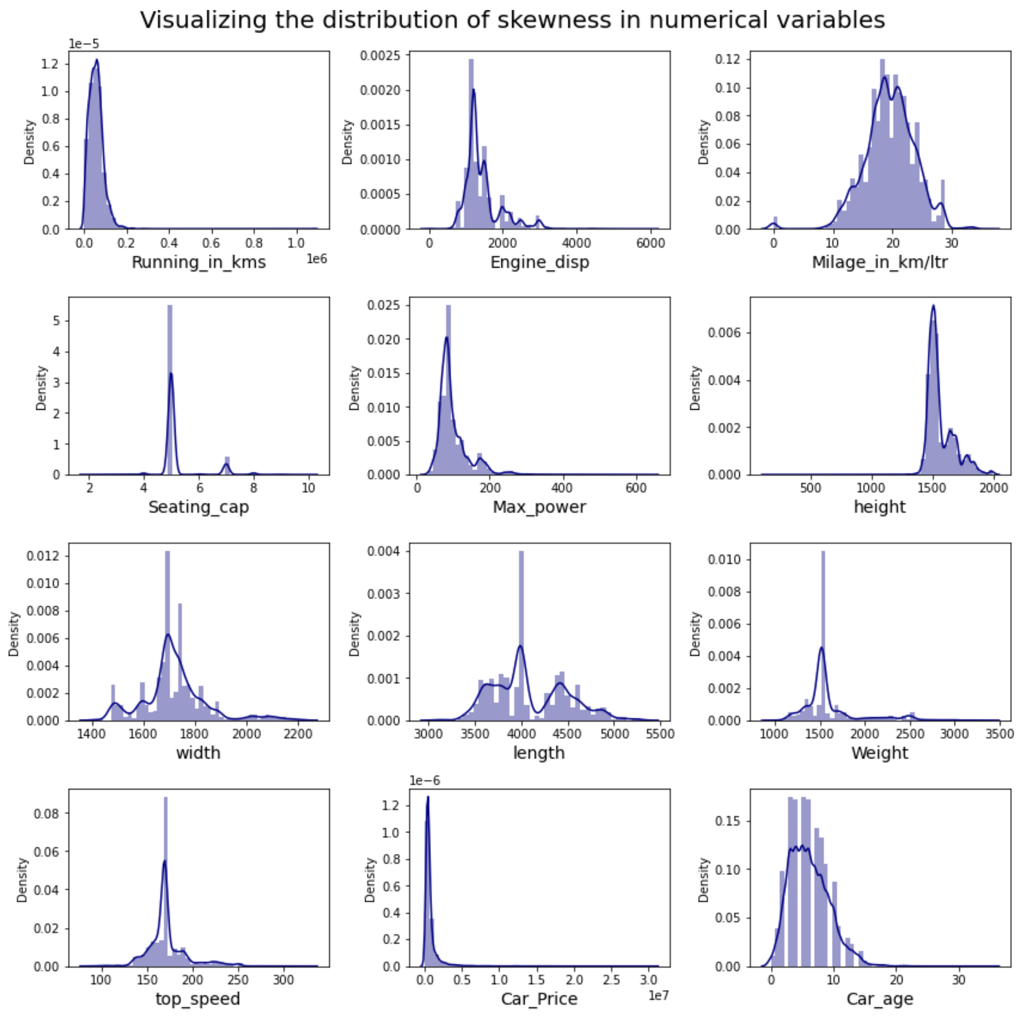


The graph shows how our final model is mapping. The plot gives the linear relation between predicted and actual price of the used cars. The blue line is the best fitting line which gives the actual values/data and red dots gives the predicted values/data.

Visualizations

I used pandas profiling to get the over viewed visualization on the pre-processed data. Pandas is an open-source Python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable. I have analysed the data using univariate, bivariate and multivariate analysis. In univariate analysis I have used distribution plot, pie plot and count plot and in bivariate analysis I have used bar plots and strip plots to get the relation between categorical variable and target column Car price and used line plot, reg plots to understand the relation between continuous numerical variables and target variable. Apart from these plots I have used violin plot, pair plot (multivariate analysis) and box plots to get the insight from the features.

Univariate Analysis: Univariate analysis is the simplest way to analyse data. "Uni" means one and this means that the data has only one kind of variable. The major reason for univariate analysis is to use the data to describe. The analysis will take data, summarise it, and then find some pattern in the data. Mainly we will get the counts of the values present in the features.



Distribution of Skewness in Numerical Columns:

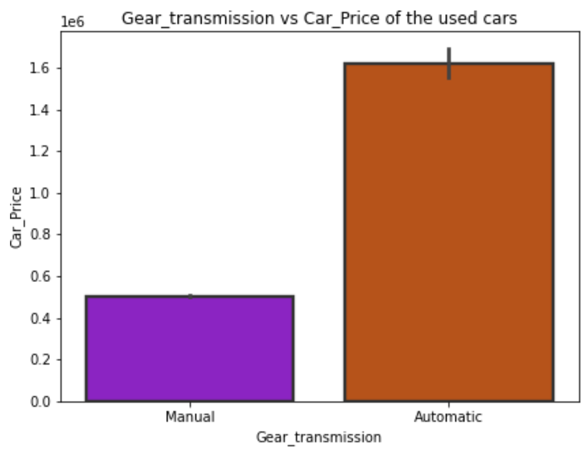
Observations:

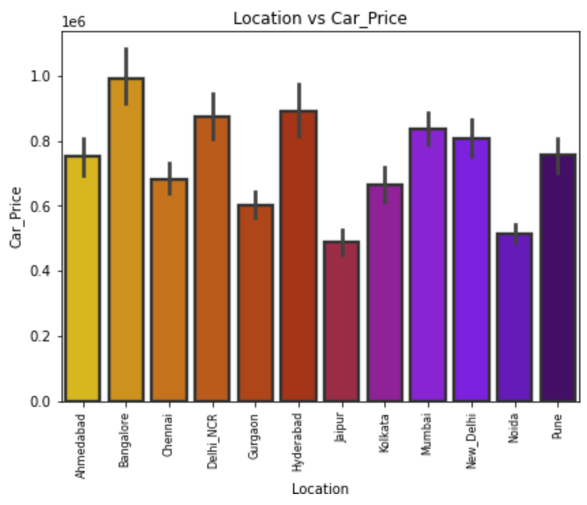
Above plot shows how the data has been distributed in each of the columns.

- ✓ From the distribution plots we can observe most of the columns are not normally distributed, only the columns "Milage_in_km/ltr" looks somewhat normal.
- ✓ Also, we can notice the columns like "Running_in_kms", "Engine_disp", "Max_power", "Weight", "Car_age" etc are skewed to right as the mean value in these columns are much greater than the median (50%).
- ✓ The data in the column "height" skewed to left since the mean values is less than the median.

Bivariate Analysis: Bivariate analysis is one of the statistical analysis where two variables are observed, Bi means two variables. One variable here is dependent while the other is independent. These variables are usually denoted by X and Y. We can also analyse the data using both independent variables. Bivariate analysis is finding some kind of empirical relationship between two variables. In this study I will be showing the relation between dependent (target) and independent variables (features) using different plotting techniques.

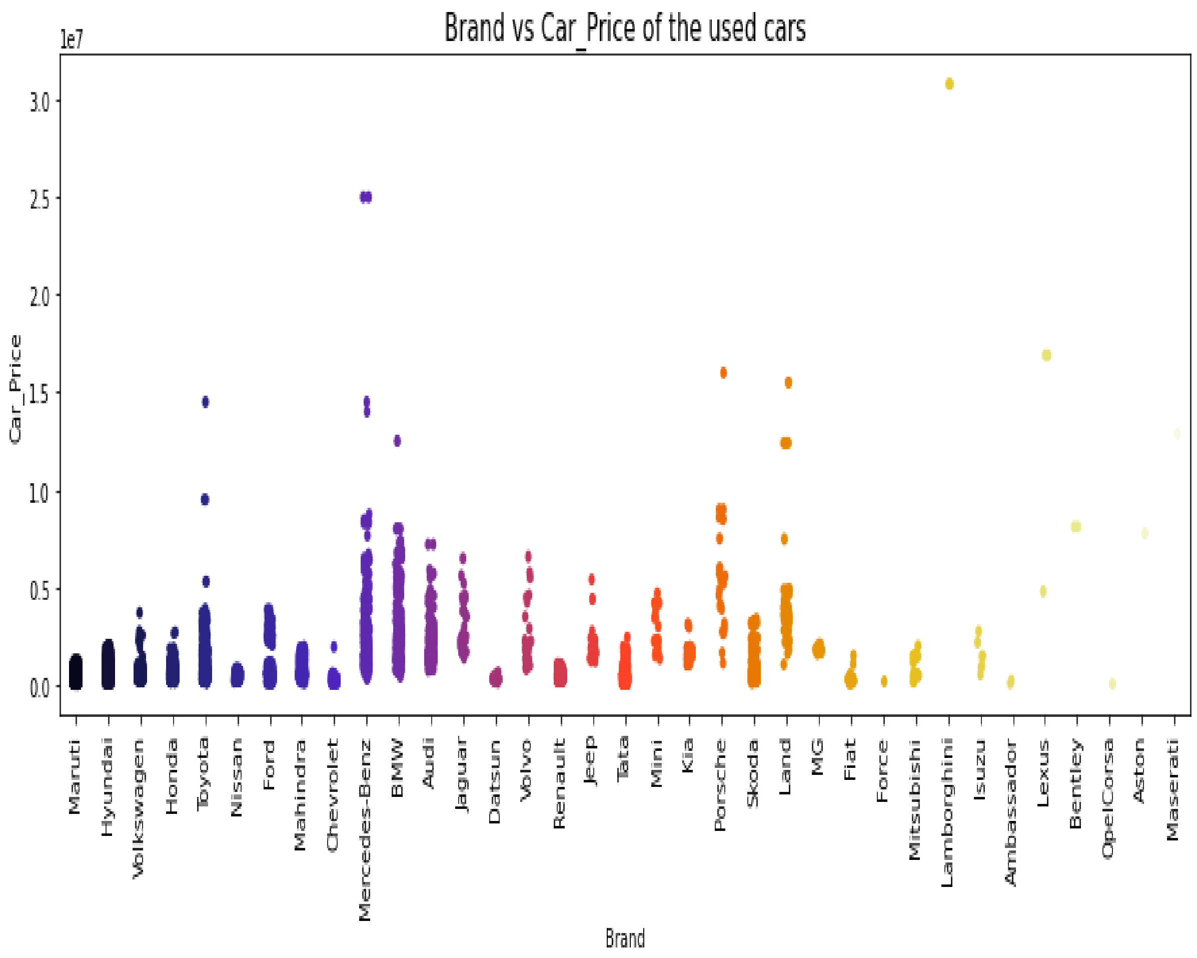
Visualizing Categorical Variables vs Car_Price:





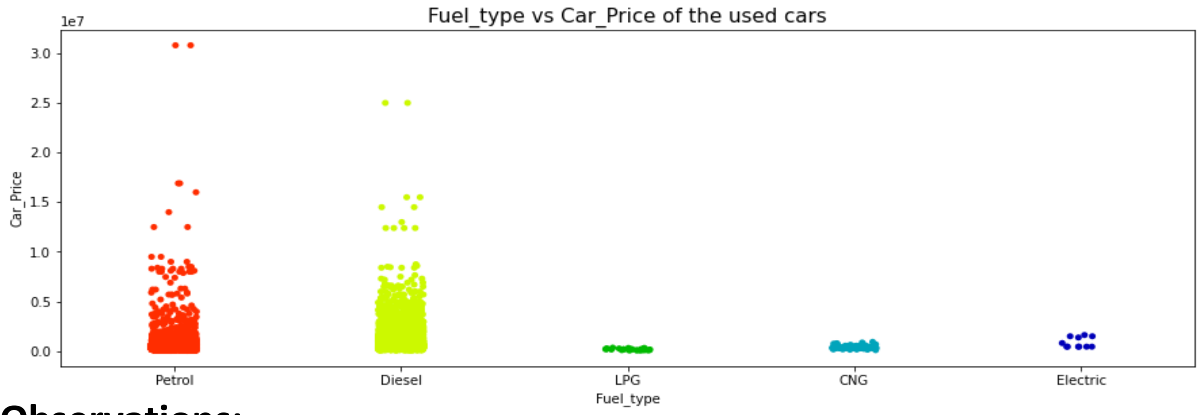
Observations:

- ✓ **Car_Price vs Gear_transmission:** From the bar plot we can observe that the cars which have Automatic gear transmission system are having high price compared to the cars which have Manual gear transmission system.
- ✓ **Car_Price vs Location:** From the second plot we came to know that the old cars from the city Bangalore have higher price followed by Hyderabad and Delhi_NCR. And the cars from the cities Jaipur, Noida, Gurgaon etc have very less price.



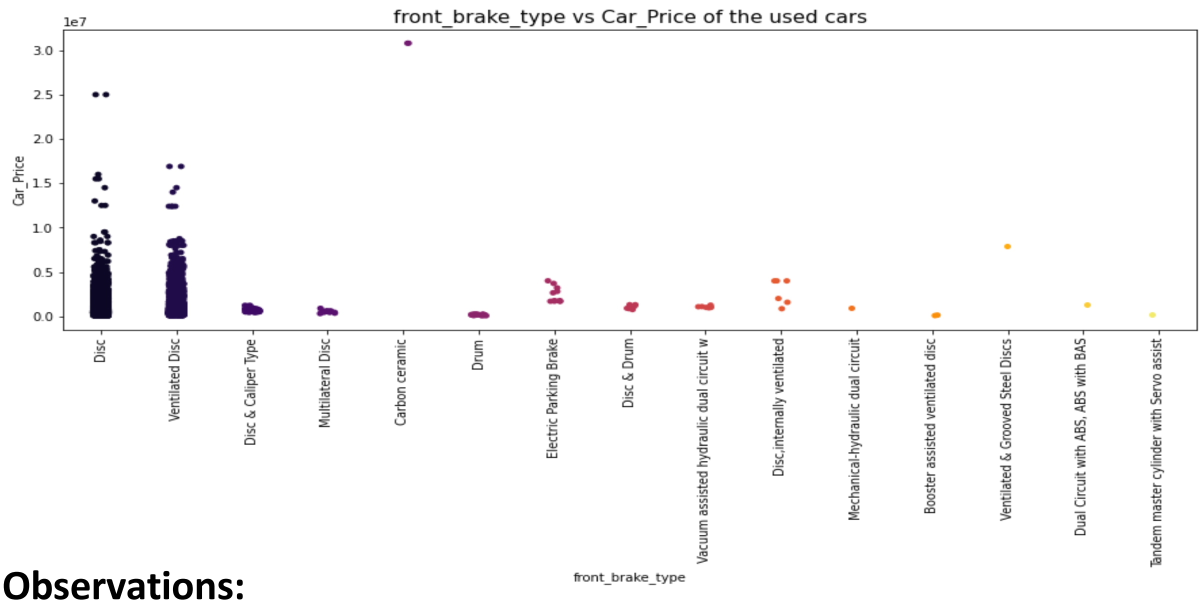
Observations:

- ✓ **Car_Price vs Brand:** The above strip plot shows how the used car prices changes depending on Brands. Here the cars from Mercedes_Benz and BMW brand have high price compared to other brands.



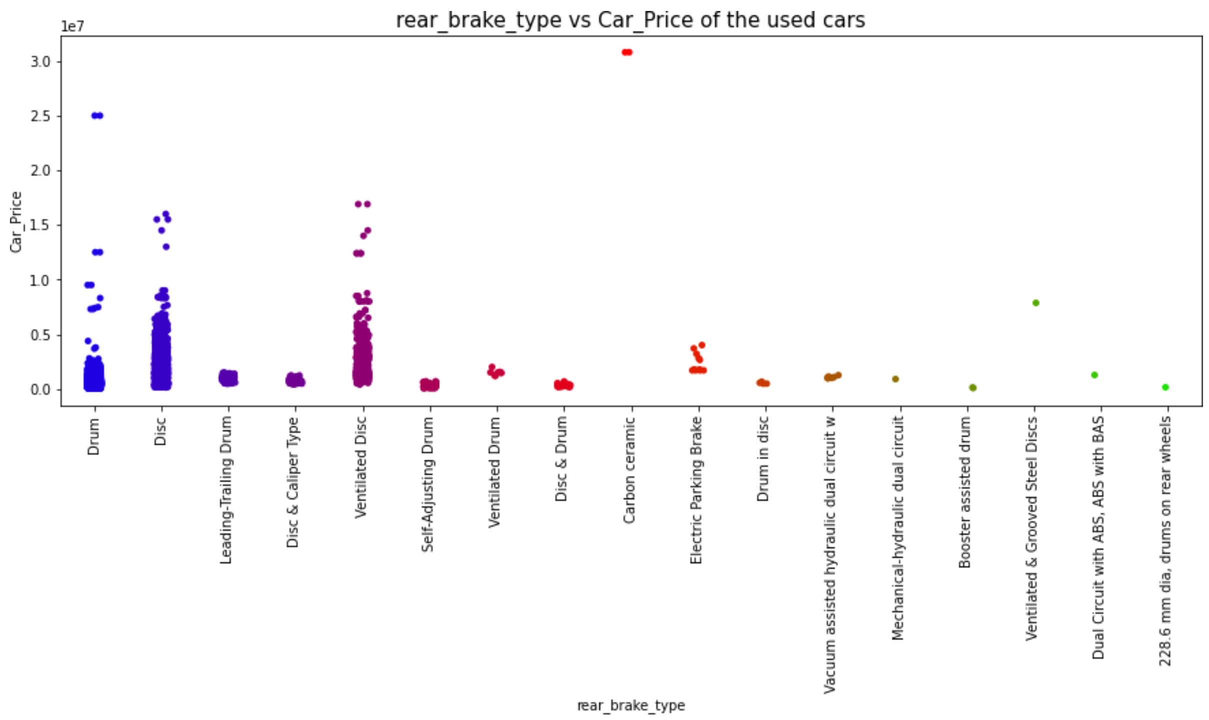
Observations:

- ✓ **Car_Price vs Fuel_type:** From the graph we can conclude that a greater number of cars are using Petrol and Diesel fuels and these cars have wide range of price from minimum to maximum. And very few of the cars uses CNG, LPG, and Electricity as fuel type which are not much expensive when compared to that of the diesel and petrol cars.



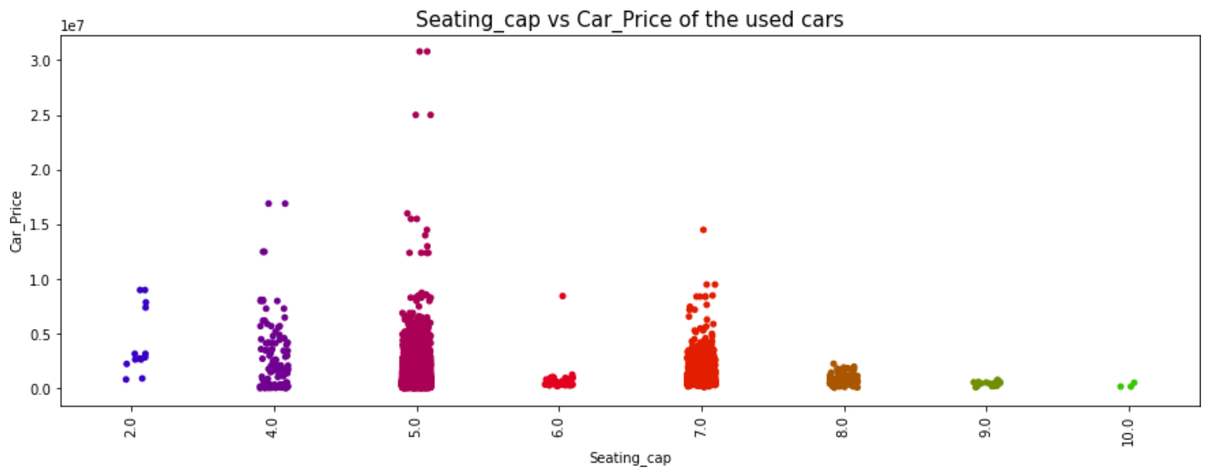
Observations:

- ✓ **Car_Price vs front_brake_type:** Looking at the above bar plot for front_brake_type vs Car_Price we can say that the cars with Disc and Ventilated Disc system for front wheels are having higher prices than other type of braking systems.



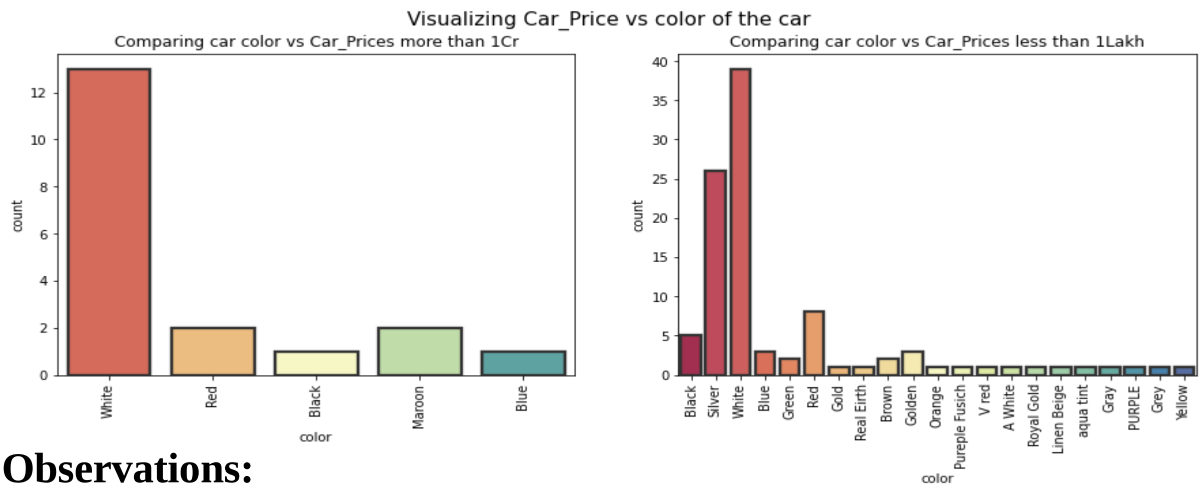
Observations:

- ✓ **Car_Price vs rear_brake_type:** The above graph is representing a bar plot for rear_brake_type vs Car_Price which tells us that the cars having Ventilated Disc or Disc or Drum brake system are having higher prices than the cars with other type of braking system at rear side.



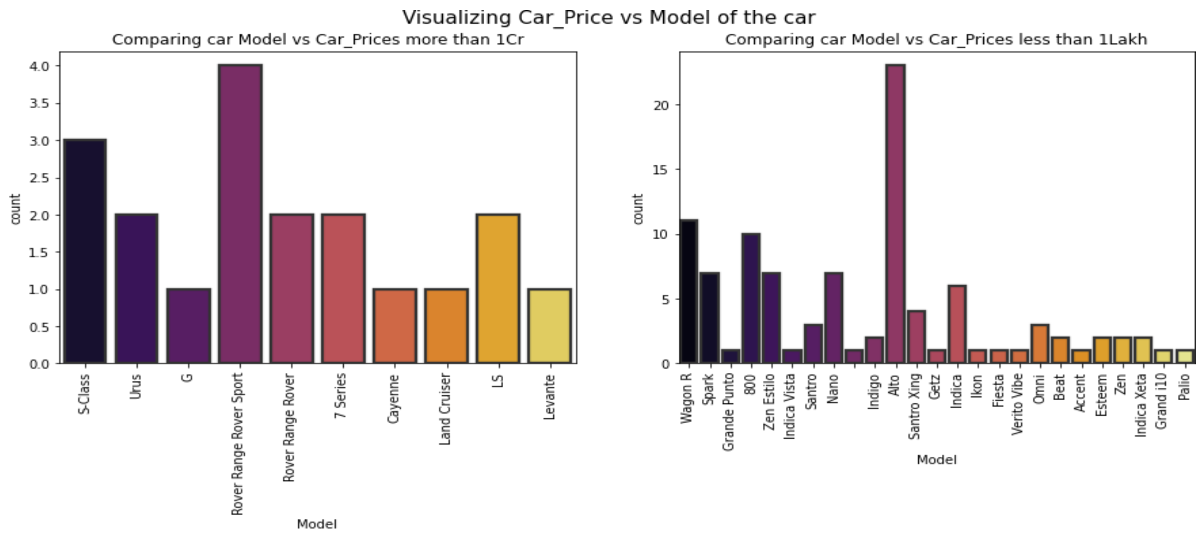
Observations:

- ✓ **Car_Price vs Seating_cap:** Most of the cars have seating capacity of 5, 7 and 4 and these cars having higher prices than other cars. And only 3 cars are observed with the seating capacity of 10.



Observations:

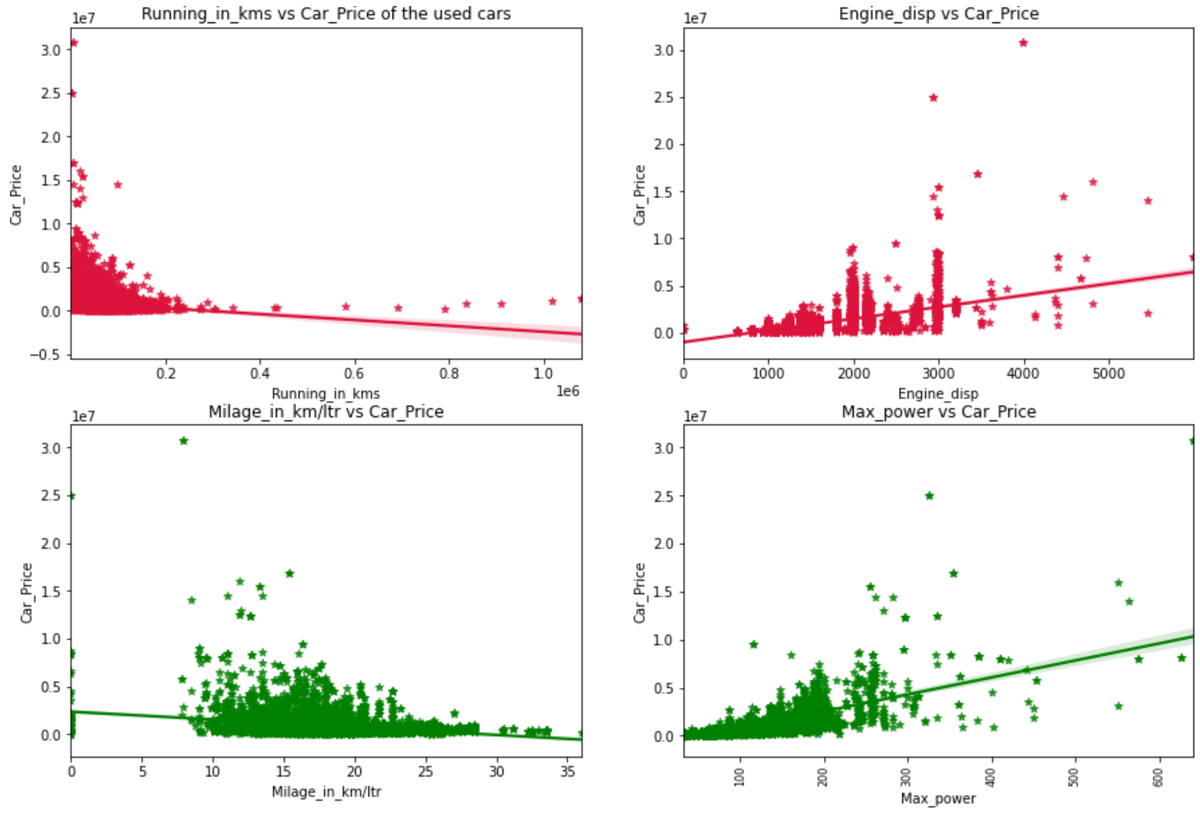
- ✓ **Car_Price vs color:** The first count plot is for the car color vs Car_Prices more than 1 Cr. The plot shows the colors of expensive cars. The white color cars are more expensive compared to the cars with other colors.
- ✓ The second graph is for the car color vs car prices below 1 Lakh and it shows the colors of cars which are cheap. From the plot we can say the cars with Silver color and white color have less price.



Observations:

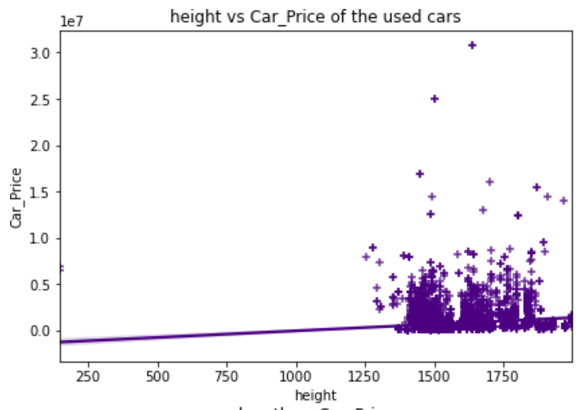
- ✓ **Car_Price vs Model:** The first plot is for car model vs car price more than 1Cr. This plot showing the models of expensive cars. The Rover Range Sport model are expensive compared to other models.
- ✓ The second plot is for car model vs car price less than 1 Lakh. This plot showing the models which are very cheap. So, from the graph we can say that the car model Alto have very less price.

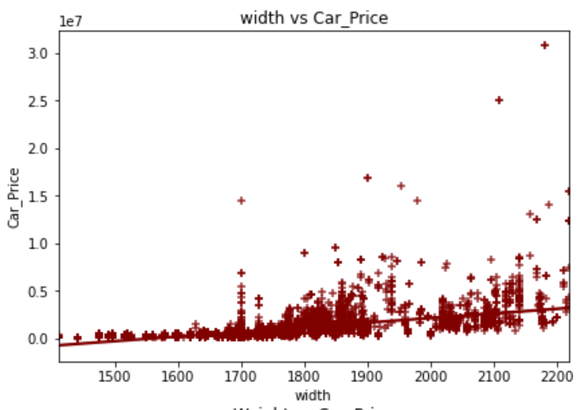
Visualizing Numerical Variables vs Car_Price:

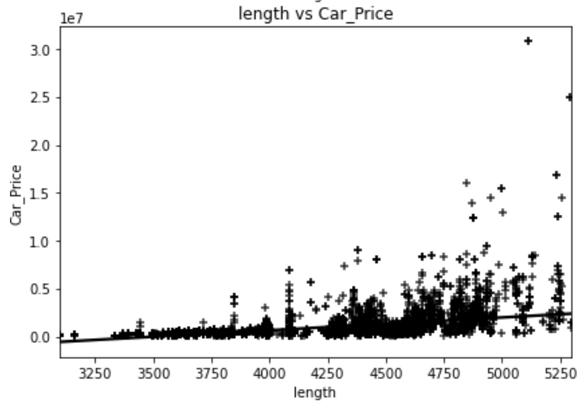


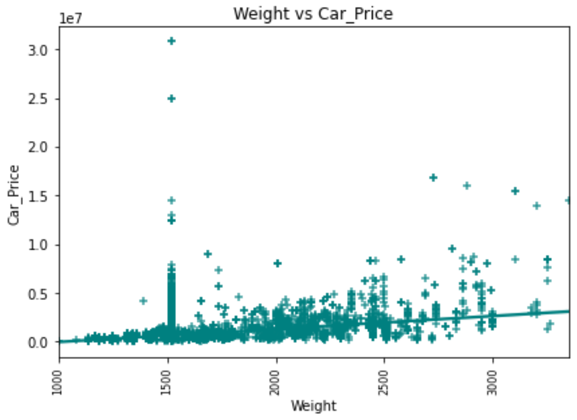
Observations:

- ✓ **Car_Price vs Running_in_kms:** From the plot we can say that the prices of cars are higher for the cars which have less running in kms. We can also notice there is negative linear relation between the price and running of cars.
- ✓ **Car_Price vs Engine_disp:** There is a positive correlation between car price and engine displacement. So, we can say as the engine disp or engine cc increases, the price of car also increases.
- ✓ **Car_Price vs Milage_in_km/ltr:** The cars having the milage in the range of 10 to 20 km/ltr are having high sale price. From the graph we can also notice there is negative linear/correlation between the price nad milage also some used cars have 0 milage which is unrealistic.
- ✓ **Car_Price vs Max_power:** Looking at the graph we can say there is positive correlation between car price and maximum engine power so, we can say as maximum power engine increases, the car prices also go on increasing.



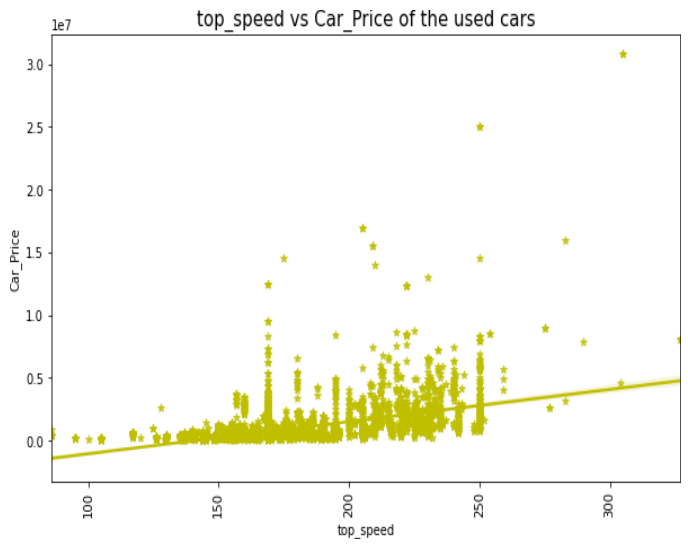




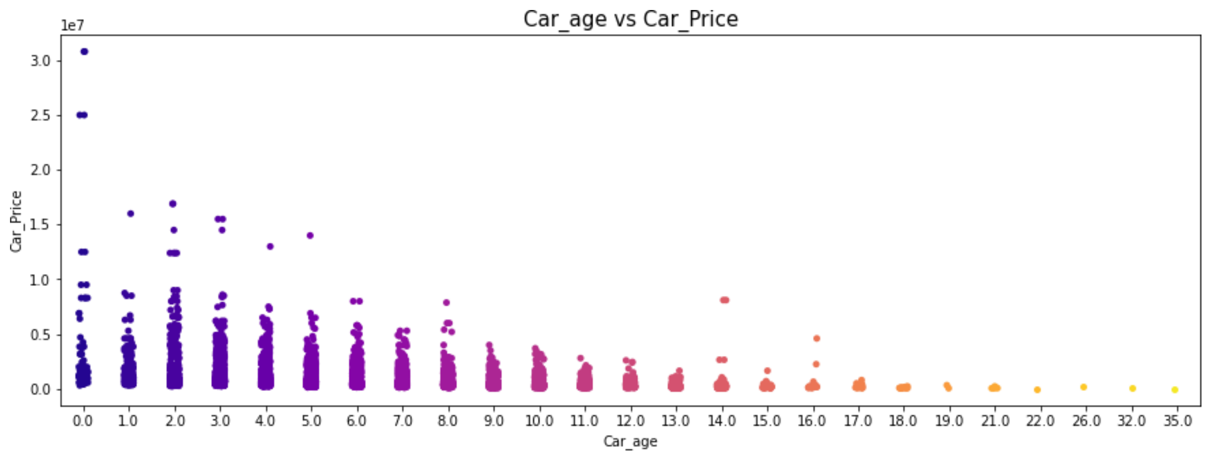


Observations:

- ✓ **Car_Price vs height:** From the graph it is clear that the car price is not strongly related with the height of the car, we can say the cars having height in the range of 1300 mm to 1800 mm have somewhat high price.
- ✓ **Car_Price vs width:** The graph shows there is some positive linear relation between car price and width of the car, so the cars having width in the range of 1700mm to 2200mm have high price. So, we can conclude as the width of the car increases, the price of the car also goes on increasing.
- ✓ **Car_Price vs length:** There is some positive linear relation between car price and length of the cars. As the length of the cars increases, the price of the cars also increases. The cars that are having the length above 4250mm have high price.
- ✓ **Car_Price vs Weight:** There is some positive linear relation between price of the car and weight. The cars with weight 1500kg have high price.

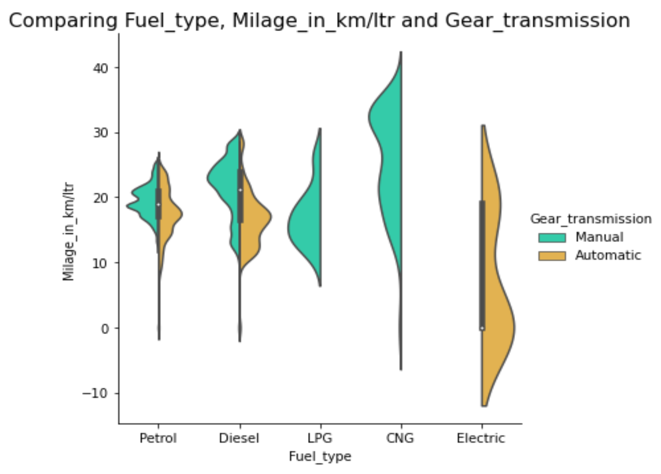


Car_Price vs top_speed: From the graph we can notice there is positive linear relation between car price and maximum speed limit of the car. The cars having top speed in the range of 120 km/hr to 250 km/hr having higher price and there are very less number of cars which have top speed below 100km/hr. So, we can conclude that as the maximum speed limit of the car (top_speed) increases, the car price also increases.

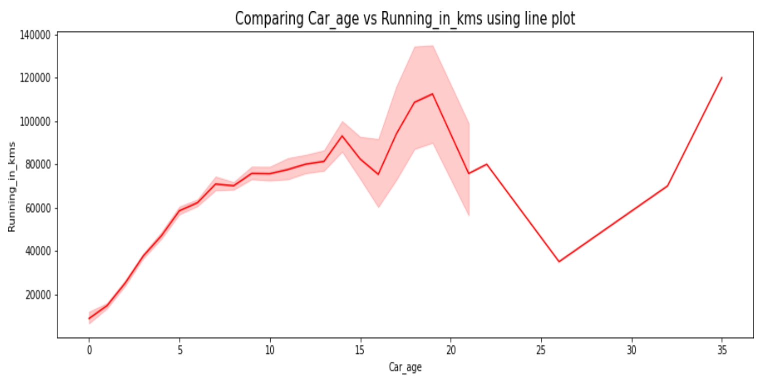


Observations:

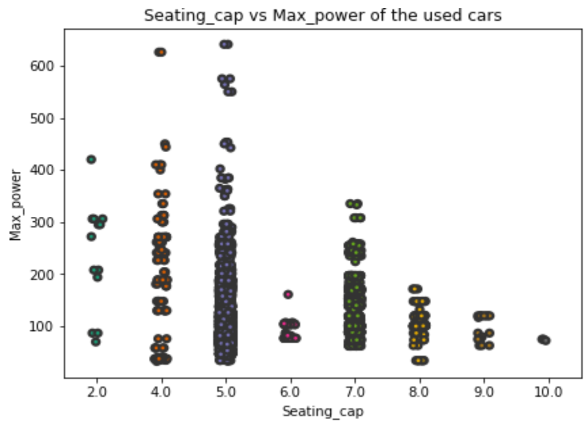
- ✓ **Car_Price vs Car_age:** From the above strip plot we can say that the older cars are having very lower prices when compared to the new cars that is the cars having very less age. So, there is negative relation between car price and age of the cars and we can conclude as the age decreases, the car prices increase.

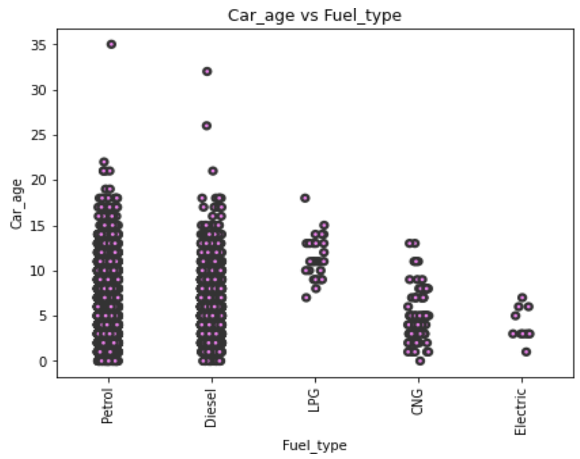


Fuel_type vs Milage_in_km/ltr: The violin plot gives the relation between Milage in km/ltr and Fuel type on the basis of gear transmission. As we can observe the cars with Manual gear transmission which are using CNG as a fuel are having good milage compared to other fuel types.



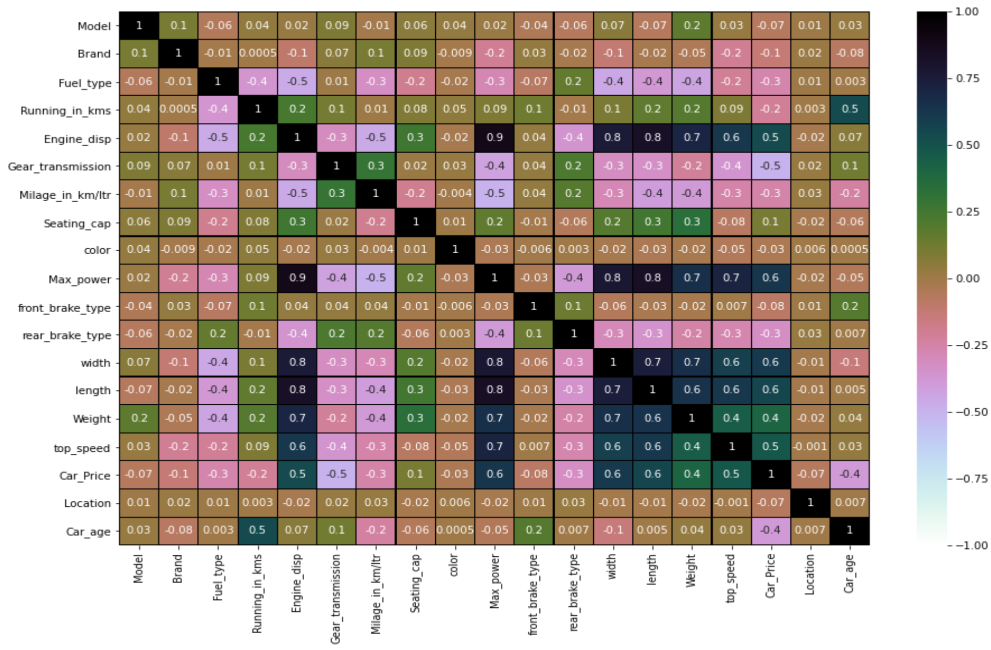
Car_age vs Running_in_kms: The above graph represents car_age vs Running in kms. The cars which have their age from 2 years to 16 years have highly used. That is the running kms for these cars are around 1 lakh kms.





- ✓ **Seating_cap vs Max_power:** The cars with seating capacity 5 have high maximum power of engine used in cars and the cars with 10 seating capacity have very less maximum engine power.
- ✓ **Fuel_type vs Car_age:** The cars which are using Patrol and Diesel as fuel they have high age and the cars with low age are using electricity as the fuel.

Correlation Between Label and Features Using Heat Map:



• Interpretation of the Results

Visualizations: In univariate analysis I have used count plots and pie plots to visualize the counts in categorical variables and distribution plot to visualize the numerical variables. In bivariate analysis I have used reg plots, bar plots, strip plots, line plots and violin plot to check the relation between label and the features. Used pair plot to check the pairwise relation between the features. The heat map and bar plot helped me to understand the correlation between dependent and independent features. Also, heat map helped to detect the multicollinearity problem and feature importance. Detected outliers and skewness with the help of box plots and distribution plots respectively. And I found some of the features skewed to right as well as to left. I got to know the count of each column using bar plots.

Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed few processing steps which I have already mentioned in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model building: After cleaning and processing data, I performed train test split to build the model. I have built multiple regression models to get the accurate R2 score, and evaluation metrics like MAE, MSE and RMSE. I got Extreme Gradient Boosting Regressor (XGB) as the best model which gives 96.27% R2 score. I checked the cross-validation score ensuring there will be no overfitting and underfitting. After tuning the best model, the R2 score of Extreme Gradient Boosting Regressor has been increased to 96.90% and also got low evaluation metrics. Finally, I saved my final model and got the good predictions results for price of used cars.

CONCLUSION

Key Findings and Conclusions of the Study

The case study aims to give an idea of applying Machine Learning algorithms to predict the sale price of the used cars. After the completion of this project, we got an insight of how to collect data, pre-processing the data, analysing the data, cleaning the data and building a model.

In this study, we have used multiple machine learning models to predict the sale price of the used cars. We have gone through the data analysis by performing feature engineering, finding the relation between features and label through visualizations. And got the important feature and we used these features to predict the car price by building ML models. After training the model we checked CV score to overcome with the overfitting issue. Performed hyper parameter tuning on the best model and the best model's R2 score increased and was giving R2 score as 96.90%. We have also got good prediction results of car price.

Findings: From the whole study we got to know that the continuous numerical variables having some strong positive linear relation with the label "Car_Price". By comparing car price and categorical variables we got to know that the cars having automatic gear transmission, cars from the city Bangalore, cars using petrol and deisel as fuels, cars having the brands Benz and BMW and cars with 5-7 seating capacity have high sale price. While comparing continuous numerical variables and Car_Price we found that cars which are having good milage, engine displacement, less running in kms have good linear relation with the price that is the cars with this kind of qualities have high selling prices. We found outliers and removed them and further removed skewness. Looking at the heat map, I could see there were some features which were correlated with each other, so I used VIF method to remove the feature causing multicollinearity and scaled the data to overcome with the data biasness.

Learning Outcomes of the Study in respect of Data Science

While working on this project I learned many things about the features of cars and about the car selling web platforms and got the idea that how the machine learning models have helped to predict the price of used cars which provides greater understanding into the many causes and benefits of selling old cars. I found that the project was quite interesting as the dataset contains several types of data. I used several types of plotting to visualize the relation between target and features. This graphical representation helped me to understand which features are important and how these features describe selling price of the old cars. Data cleaning was one of the important and crucial things in this project where I dealt with features having string values, features extraction and selection. Finally got XGBoosting Regressor as best model.

The challenges I faced while working on this project was when I was scrapping the real time data from cardekho website, it took so much time to gather data. The data was quite difficult to handle and cleaning part was challenging for me but fixed it well and it was unable to remove skewness in Seating_cap column so moved further keeping it as it is.

Finally, our aim was achieved by predicting the sale price of used cars and built car price evaluation model that could help the clients to understand the future price of used cars.

Limitations of this work and scope for future work

Limitations: The main limitation of this study is the low number of records that have been used. In the dataset our data is not properly distributed in some of the columns many of the values in the columns are “-“ and some values which are not realistic. Because I have seen the column Running in kms showing 0 kms and some of the cars having age as 0 years which are not possible in case of used cars. So, because of that data our models may not make the right patterns and the performance of the model also reduces. So that issues need to be taken care.

Future work: As future work, we intend to collect more data and to use more advanced techniques like artificial neural networks and genetic algorithms to predict car prices. In future this machine learning model may bind with various website which can provide real time data for price prediction. Also, we may add large historical data of car price which can help to improve accuracy of the machine learning model.

References:

I have also used few external resources that helped me to complete this project successfully. Below are the external resources that were used to create this project.

1. https://www.google.com/
2. https://scikit-learn.org/stable/index.html
3. www.ripublication.com
4. www.irjet.net
5. www.ijcseonline.org
6. https://towardsdatascience.com/
7. https://www.analyticsvidhya.com/