A

Major Project Report

On

USED CAR PRICE PREDICTION USING MACHINE LEARNING

Submitted for the partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

Submitted to:

Prof. Sanjeev Khambra
Dr. Jai Bhagwan
Deptt. of CSE
GJUS&T, Hisar

Submitted by:

Chirag
180010130033
Deepanshu
180010130035
B.Tech (CSE) – 8th Sem

Department of Computer Science & Engineering
Guru Jambheshwar University of Science & Technology, Hisar
'A' Grade NAAC Accredited
2018-2022

CANDIDATE'S DECLARATION

I/we, hereby declare that the project work entitled "*Used Car Price Prediction Using Machine Learning*" is an authentic work carried out by me/us under the guidance of Prof. Sanjeev Khambra and Dr. Jai Bhagwan, Department of Computer Science & Engineering in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering and this has not been submitted anywhere else for any other degree.

Date: Signature

Chirag
180010130033

Deepanshu
180010130033

CERTIFICATE

This is to certify that *Chirag(180010130033)* and *Deepanshu(180010130035)* are students of B.Tech (CSE), Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar have completed the project entitled "*Used Car Price Prediction Using Machine Learning*".

Prof. Sanjeev Khambra
Deptt. of CSE
GJUS&T, Hisar

Dr. Jai Bhagwan
Deptt. of CSE
GJUS&T, Hisar

PLAGIARISM CERTIFICATE

This is to certify that *Chirag(180010130033) and Deepanshu(180010130035)* are students of B.Tech (CSE), Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar have completed the project entitled "*Used Car Price Prediction Using Machine Learning*".

Their complete project report has been checked by Turntin Software and the similarity index is _____ i.e. the accepted norms of the university. The project report may be considered for the award of the degree.

Supervisor Signature

Chirag
180010130033

Deepanshu
180010130035

Contents

USED CAR PRICE PREDICTION USING MACHINE LEARNING

Chapter 1

Introduction

The pre-owned and used vehicle market is a developing business with a market esteem that has almost multiplied itself in recent years. The ascent of online sites and other instruments like it have made it more straightforward for the second purchasers and merchants to improve comprehension and understanding of the variables that decide the market worth of a pre- owned/ used vehicle. In light of a set of variables in consideration, Machine Learning calculations might be used to conjecture the cost of any vehicle to the point of almost accuracy. The informational collection will remember data obtained from various sources for an assortment of vehicles. There will be data concerning the vehicle's specialized components, for example, the motor kind, fuel type, total and driven kilometers, and that's only the tip of the iceberg, for each vehicle.

There is currently no all-inclusive instrument for building up the retail cost of utilized vehicles in light of the fact that unique sites utilize various techniques to make it. By utilizing measurable models to expect to value, it is conceivable to acquire a fundamental value gauge without entering everyone of the subtleties into the ideal site. The fundamental motivation behind this study is to think about the precision of two distinct expectation models for assessing a pre-owned vehicle's retail cost. Subsequently, we offer a Machine Learning-based philosophy of anticipating the costs of secondhand vehicles in light of their attributes.

This philosophy can help purchasers hoping to buy a pre-owned vehicle inmaking more informed decisions. Clients can now search for all vehicles in a district without actual endeavors, whenever and from any area.

Several transformational and important changes were observed in the vehicle market as the consequences of the Coronavirus pandemic.
Currently some vehicles are in demand so they are overpriced and some are not popular and therefore cheaper in price than other vehicles. As the market adjusts due to the impact of the coronavirus 19, sellers/dealers are encountering problems with their previous car pricing AI / machine learning models that were used before the pandemic hit. Along these lines, both the sellers and the buyers are actively looking out for new and improved Artificial Intelligence based models based on the latest information and more variables. This is where the car price evaluation model that we built comes into play.

Creation of a dataset with the help of web scraping and the anticipation of the vehicles given considering the various factors is the primary objective of this venture.

The costs of new vehicles in business areas are taken care of by the producer for certain extra

expenses brought about by the Government as assessments. Along these lines, clients purchasing another vehicle can be guaranteed of the money/investment they contribute to be commendable. Be that as it may, because of the expanded cost of new vehicles and the ineptitude of clients to purchase new vehicles because of the absence of vehicles deals are on a worldwide increment. There is a requirement at a pre-owned vehicle cost expectation framework to successfully decide the value of the vehicle utilizing an assortment of highlights.

In the commercial areas, the cost of new vehicles increased by the manufacturer because of certain additional costs generated by the government, such as appraisals or even assessments. In this way, customers who buy the other vehicle can be sure that the investments invested is creditworthy. Be that as it may, the price increase of new vehicles and the inability of buyers to buy new vehicles due to the scarcity of vehicle offers are increasing worldwide. The car costs used in the expectation framework stipulate that the value of the vehicle must be successfully determined on the basis of various highlights.

Despite the fact that there are sites that offers this assistance,their expectation technique may not be awesome. Also, different frames and models can help predict the true market value of a used vehicle. During both operations, it is essential to recognize its true market value.

Having the option to anticipate utilized/used vehicles market worth can help the two purchasers and merchants. Utilized Vehicle merchants are one of the greatest objective gathering that can be keen on consequences of this review. On the off chance that pre-owned vehicle merchants better get what makes a vehicle attractive, what the significant highlights are for a pre-owned vehicle, then, at that point, they might think about this information and proposition a superior assistance.

Problem Statement

A model to anticipate the price of a pre-owned vehicle should be developed in order to assess its value based on a variety of characteristics. Several factors affect the price of a used car, such as company, model, year, transmission, distance driven, fuel type, seller type, and owner type. As a result, it is crucial to know the car's actual market value before purchasing or selling it.

Data Sources and their formats

There has been a continuous paradigm of commodity exchanges in existence for a long time. Previously, these transactions were conductedthrough a barter(exchange of goods) system, that ultimately was converted into a system based on money. And, as a result of these considerations, any changes in the pattern of re-selling things were also affected. The resale of an object can be accomplished in two ways.

The first is offline, while the second is online. In offline transactions, there is a middleman who is extremely susceptible to corruption and making excessively lucrative deals. The second alternative is to sell it online, where there are websites and platform that allows the costumers to find out whatprice they may earn if they sell it.

We scrape the data for 5000+ cars from websites like Olx, cars24, Cardekho and AutoPortal. And

save it for Machine Learning Model.

We have extracted attributes like Brand, model name along with its manufacturing year; Variant; Fuel type; number of owners; location;Date of posting ad online; transmission; driven kilometers and lastly,price. Price is an integer type column and is our target variable. Rest all attributes are of object data type.

Mathematical/ Analytical Modeling of the Problem

We find out that attributes like brand, model, manufacturing year, variant, total driven kilometers, and date of posting ad online and location have very wide range of variables and it's not very for us to study their plots. We certainly do not need to perform bivariate analysis on these.

Chapter 2

Existing System

The primary existing system is manual and requires the buyer to contact the seller/dealer by himself and then trust the seller enough to ask for a considerate amount for the vehicle. The seller then would ask for a certain amount which might be reasonable or unreasonable, also the market value has to be checked manually by consulting various dealers/ marketplaces which is tedious and time consuming. The number of vehicles in this survey exercise is usually low due to intense travelling and other hassles and inconveniences such as travelling costs and time taken to achieve so. Sometimes the brand or model of the care on sale might be so rare and uncommon that it might become impossible to calculate its actual worth despite its conditions.

The key factors such as model, fuel type, mileage, miles driven etc. become secondary in this interaction and thus the price of the vehicle is usually not accurate or up to the market rate.

Secondly, though not to a very large extent, Artificial intelligence or machine learning based systems had also been come up with pre-pandemic, and had even found for themselves a considerable user base, but even then they were not found to be accurate and consistent with the vehicle prediction results and pandemic even hit them worse in terms of their accuracy and reliability since there were categorical changes in the factors that were taken into consideration to determine the price of the vehicle.

Chapter 3

Problems in Existing System

Following are the several issues with the existing system that the proposed system intends to address;

1. The existing system is manual and hence is inefficient in terms of efforts, money spent and the outcomes.
2. The existing system is time consuming and inconsistent with results.
3. The existing system is more prone to human errors.
4. The number of data entries taken into consideration are limited to the certain geographical area such as the town or the city.
5. There is an obvious limitation when an entry of car with unpopular brand or model is encountered and the system fails to attend to it.

Chapter 4

Proposed System

Determining the list price of a pre-owned vehicle is a big challenge because there might be several determining factors that affect the market price of a used car. The primary goal of the project is to develop Machine Learning models that can anticipate the price of a pre-owned vehicle accurately taking in consideration its characteristics, so that informed deals of purchases can be made.

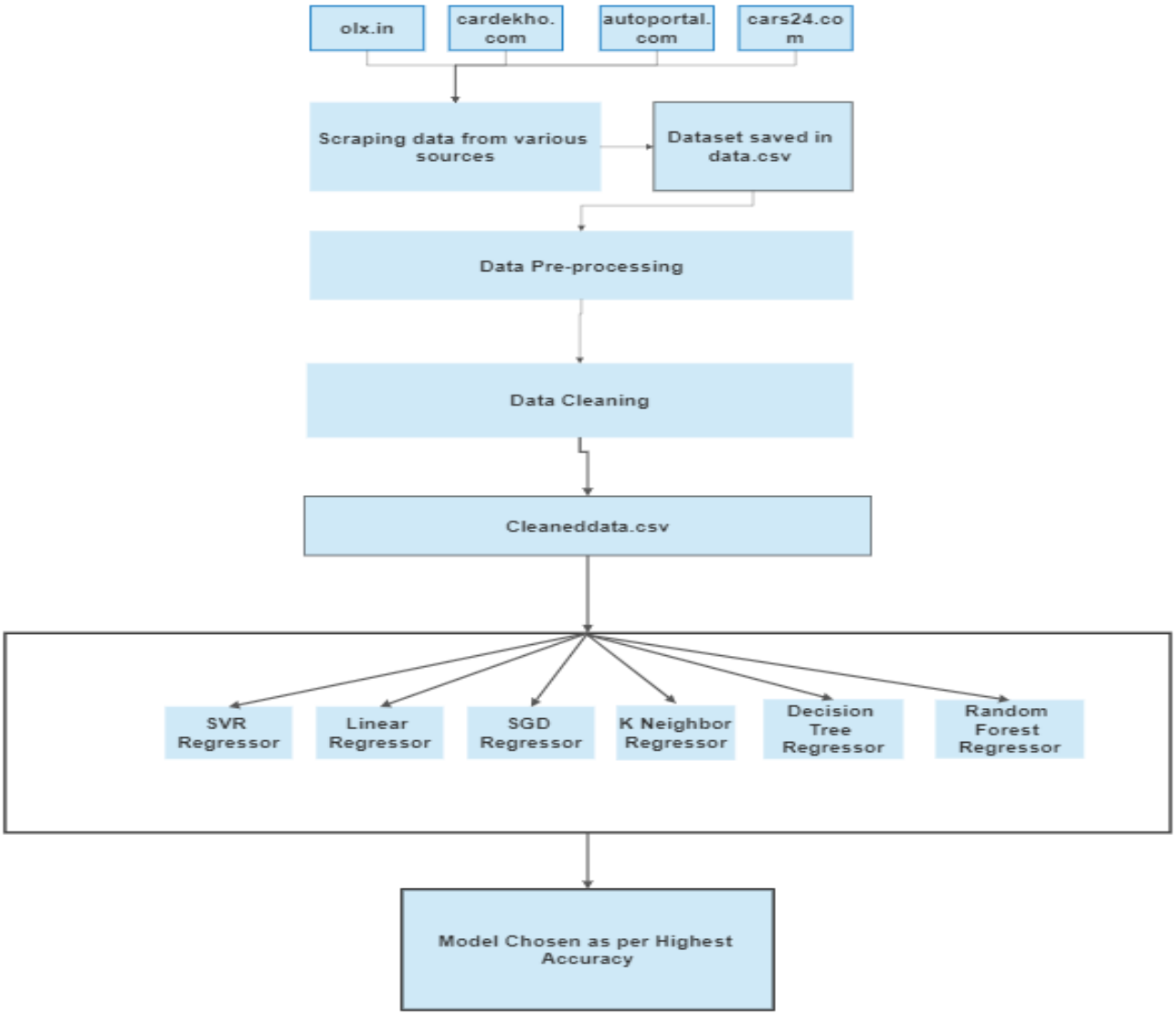
Forecasting of vehicle prices with accuracy requires specialized knowledge, as the price may often find dependency on many different elements , factors and features. Generally, the most significant ones are make and model, brand, miles driven, condition, age, mileage and horsepower of the vehicle.

Another major factor that affect the cost of the pre-owned vehicle is the type of fuel that the vehicle runs on and the mileage(kilometers per liter) of the vehicle because the petroleum prices are subject to changes and daily fluctuations, thus the diesel, petrol and gas prices.

Various characteristics of the vehicle such as the color of the exteriors, number of doors, transmission type, dimensions of the vehicle, passenger safety, air conditioning, interior space and conditions, with or without navigation, also affect the price of the car. Through the same project, we use different techniques and methods to obtain a more accurate forecast of used vehicle prices.

Instead of predicting the range of the price of the vehicle, we were to predict the absolute price of the vehicle, so we chose to use regression algorithms due to the fact that a continuous value is produced as the outcome of this algorithm rather than some classified categorical values.

Both the processes of gathering of the data through web scraping and then the actual analysis of the gathered data turned out to be complex because the size of the data collected and the number of entries made for used and pre-owned vehicles was very large, a number in thousands, also a single entry of a vehicle consists of values of several characteristics of the vehicle.



Used cars data scrape from various websites, and popular platforms for putting up on sale pre-owned and new vehicles in India. Features like the brand and model of the vehicle, make, seating capacity, color, fuel efficiency, engine capability, brakes, Torque, Transmission type, Maximum power, Gearbox type, type of steering(power or not), type of engine, turbocharger, supercharger and obviously the price were considered

Chapter 5

Advantages of Proposed System

1. While imagining a situation where you have an old car and want to sell it. You may of course approach an agent for this and find the market price, but later may have to pay pocket money for his service in selling your car. But what if you can know your car selling price without the intervention of an agent. Or if you are an agent, definitely this will make your work easier. Yes, this system has already learned about previous selling prices over years of various cars.
2. Since this system is not manual, the process is fast, efficient and obviously pocket friendly.
3. So, to be clear, this deployed project provides its users the approximate selling price for the pre-owned vehicle based on the fuel type, years of service, showroom price, the number of previous owners, kilometres driven, if dealer/individual, and finally if the transmission type is manual/automatic. And that's a brownie point.
4. Any kind of modifications can also be later inbuilt in this application. It is only possible to later make a facility to find out buyers. The applications of Machine Learning don't end here

Chapter 6

System Requirements and Tools Used

Hardware requirements:

Processor	:	clock speed 1.50GHz and above
RAM	:	4 GB and above
Hard disk	:	50 GB and above

Software requirements:

OS	:	Windows 7 or above
Technology	:	Python 3.8
IDE	:	Jupyter Notebook
Web Browser	:	Mozilla Firefox/ Google Chrome/ Microsoft Edge

Libraries

Following python libraries were used in this project:

- Selenium
- NumPy
- Pandas
- Matplotlib
- SciPy
- Scikit-Learn

Python was the most popular technology for implementing machine learning ideas, owing to the fact that it has a large number of built-in algorithms in the form of bundled libraries. The following are some ofthe most important libraries and tools that have been used in this project:

1. **Numpy:**
   NumPy is a Python module for array processing. It includes a high-performance multidimensional array object as well as utilities for manipulating them. It is the most important Python module for scientific computing. NumPy may be used as a multi-dimensional container of general data in addition to its apparent scientific applications.
   Numpy allows any data types to be created, allowing NumPy to connect with a broad range of databases cleanly and quickly.

2. **Pandas:**

Pandas is a Python library generally employed for working with data sets. It facilitates its users with functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" traces its origin r reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Pandas allows the users to analyze big data and make conclusions and assortments based on statistical theories. Pandas can be used to clean messy data sets, and make them readable, understandable and relevant. Relevant data is very important in data science just like in this project itself.

3. **Selenium:**

Selenium is considered as one the several popular and commonly used open source Web User Interface based automation testing suites.

Selenium was developed in 2004 as a tool for the purpose of internal use while working at thought works by Jason Huggins.

Selenium extends its support to be used for automation across almost all web browsers, programming technologies and OS platforms. Deployment of selenium can be made with on different operating systems including but not limited to Microsoft Windows, Linux based OS, Solaris and even Apple Macintosh OS. Support for mobile based operating systems has also been provided(for Android , Windows Mobile and Apple IOS).

4. **Matplotlib:**

Matplotlib is an amazingly capable and powerful visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library for python built on NumPy arrays and designed to work with the broader and versatile SciPy stack. It was introduced to public by John Hunter in the year 2002.

One of the greatest advantages of visualization is that it allows the users visual access to huge amounts of data in easily digestible/ consumable visuals. Matplotlib is capable of being used to obtain several plots like line, bar, scatter, histogram etc.

5. **SciPy:**

SciPy is a Python library for scientific and technical computing that is free and open-source. Optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other activities used in research and engineering areall covered by SciPy modules.

SciPy is based on the NumPy array object, and it is a part of the NumPy stack, which also contains Matplotlib, pandas, and SymPy, as well as a growing number of scientific computing libraries. Other apps with comparable users to NumPy include MATLAB, GNU Octave, and Scilab.

The SciPy stack is occasionally used interchangeably with the NumPystack. The SciPy library is now available under the BSD license, with an open community of developers sponsoring and supporting itsdevelopment.

6. Scikit-Learn

Scikit-learn offers a standard Python interface for a variety of supervised and unsupervised learning techniques. It is provided under several Linux distributions and is licensed under a liberal simplified BSD license, promoting academic and commercial use. The library is being constructed.

7. Jupyter Notebook

Jupyter Notebook is an open-source online software that lets you create and share documents with live code, equations, visualizations,and narrative prose. Data cleansing and transformation, numerical statistical modelling, data visualization, machine learning,and more are all included.

Jupyter Notebook is an open-source online software that lets you create and share documents with live code, equations, visualizations, and narrative text. Data cleansing and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and more are all included.

Chapter 7

Proposed System Design

39 missing/null values were observed in column ‘variant’ (for the then obtained and considered dataset) and since it is a column with data type categorical data type , we will replace the null values with mode.

Data Cleaning

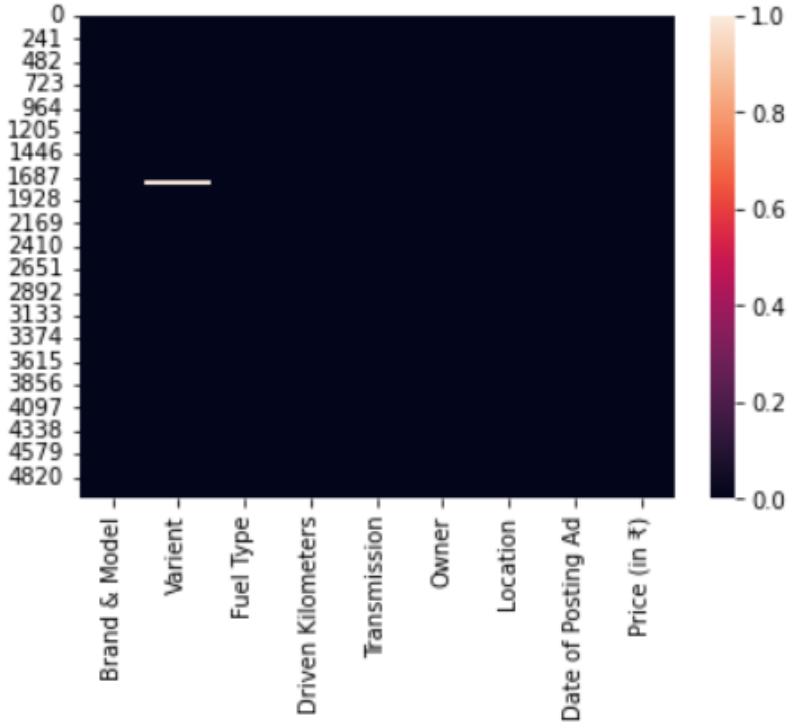
In [8]:
```
#Checking for Null values
df.isnull().sum()
```

Out[8]:
```
Brand & Model          0
Varient               39
Fuel Type              0
Driven Kilometers      0
Transmission           0
Owner                  0
Location               0
Date of Posting Ad     0
Price (in ₹)           0
dtype: int64
```

There are 39 null values in column 'Varient' and since it is of object type, we will use Mode to fill the rows with null value.

```
In [9]: sn.heatmap(df.isnull())
```

```
Out[9]: <AxesSubplot:>
```



This is the visualization of having just a small number null values in the dataset.

Model/s Development and Evaluation

1. Identification of possible problem-solvingapproaches (methods)

We go over the many techniques and datasets that were used to createthis module.

The model will be trained using a dataset comprising over 5000 tuples. The value of a car is determined by factors such as the number of kilometers driven, the year of registration, the kind of gasoline used, and the financial strength of the owner. We created regressor methods and compared the two on different car models because this is a regression problem.

Anaconda seeks to address Python's dependency hell, where distinct projects have various dependency versions, so that project dependencies do not require separate versions, which might conflict.

2. Testing of Identified Approaches (Algorithms)

For the purpose of training and then testing the datasets, the following models were used:

1. SVR(Support Vector Regression)
2. Linear Regression
3. SGD Regressor(Stochastic gradient descent)
4. K Neighbors Regressor
5. Decision Tree Regressor
6. Random Forest Regressor

3. Run and Evaluate selected models

1. **SVR:**
   SVR is based on the same principles as SVM, with a few small exceptions. It tries to determine the curve given a set of data points. However, because it is a regression technique, rather of utilizing the curve as a decision boundary, the curve is used to identify a match between the vector and the curve's location. Support Vectors aid in establishing the most accurate match between data points and the function used to represent them.

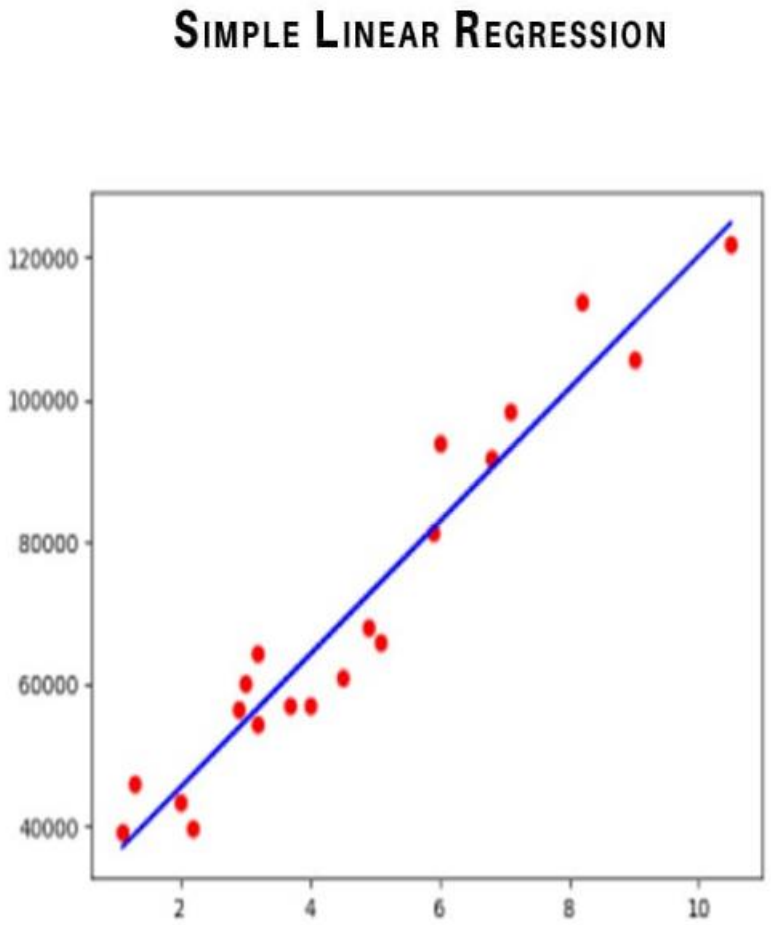
```
: xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_state = 45)
  svr = SVR()
  svr.fit(xtrain,ytrain)
  pred_train_svr=svr.predict(xtrain)
  pred_test_svr=svr.predict(xtest)
  print('SVR Regressor Score:',svr.score(xtrain,ytrain))
  print('SVR Regressor r2_score:',r2_score(ytest,pred_test_svr))
  print("Mean squared error of SVR Regressor:",mean_squared_error(ytest,pred_test_svr))
  print("Root Mean Square error of SVR Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_svr)))
```

```
SVR Regressor Score: -0.06325370730174851
SVR Regressor r2_score: -0.06820672247987791
Mean squared error of SVR Regressor: 297284446193.9865
Root Mean Square error of SVR Regressor: 545237.9720764012
```

The Accuracy of SVR is in negative which stats that this is not the correct model to apply here.

2. Linear Regression

Regression is a method for predicting a dependent component with the help of independent variables.



The method is commonly used to predict and calculate correlations between independent and

dependent variables. Theregression model establishes a linear or exponential connection between independent and dependent variables.

Linear regression is a type of regression analysis in which the independent(x) and dependent(y) variables can be constrained in a linear relationship. The red line in the graph above is knownas the best fit straight line. We want to draw a line that best predicts the data points given the data points we have. The line may be represented using the linear equation below.

$$\underline{y = a0 + a1 * x} \qquad \text{\# Linear Equation}$$

```
lr= LinearRegression()
lr.fit(xtrain,ytrain)
lr.coef_
pred_train=lr.predict(xtrain)
pred_test=lr.predict(xtest)
print('Linear Regression Score:',lr.score(xtrain,ytrain))
print('Linear Regression r2_score:',r2_score(ytest,pred_test))
print("Mean squared error of Linear Regression:",mean_squared_error(ytest,pred_test))
print("Root Mean Square error of Linear Regression:",np.sqrt(mean_squared_error(ytest,pred_test)))
```

```
Linear Regression Score: 0.055020586349048384
Linear Regression r2_score: 0.061364713139148486
Mean squared error of Linear Regression: 261224410556.7481
Root Mean Square error of Linear Regression: 511101.1744818712
```

The accuracy of Linear Regression is only 6%

3. SGD Regressor

The loss gradient is calculated each sample at a time, and the model is updated along the way using a decreasing strength schedule. SGD stands for Stochastic Gradient Descent (aka learning rate).

The regularizer is a penalty applied to the loss function that decreases model parameters towards zero using either the squared Euclidean norm L2 or the absolute norm L1 or a mix of the two (Elastic Net). The update is trimmed to 0.0 whenever the parameter update passes the 0.0 value due to the regularizer, allowing for the learning of sparse models and online feature selection.

```
sgd=SGDRegressor()
sgd.fit(xtrain,ytrain)
pred_train_sgd=sgd.predict(xtrain)
pred_test_sgd=sgd.predict(xtest)
print('SGD Regressor Score:',sgd.score(xtrain,ytrain))
print('SGD Regressor r2_score:',r2_score(ytest,pred_test_sgd))
print("Mean squared error of SGD Regressor:",mean_squared_error(ytest,pred_test_sgd))
print("Root Mean Square error of SGD Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_sgd)))
```

```
SGD Regressor Score: 0.05307367896974513
SGD Regressor r2_score: 0.06076093246004877
Mean squared error of SGD Regressor: 261392444141.47607
Root Mean Square error of SGD Regressor: 511265.5319317703
```

The accuracy of SGD Regressor is also very poor, it's only 6%

4. KNeighbors Regressor

Algorithm calculating the average of the numerical goal of the K nearest neighbors is a straightforward implementation of KNNregression. An inverse distance weighted average of the K closest neighbors is another method. The distance functions used in KNN regression are the same as those used in KNN classification.

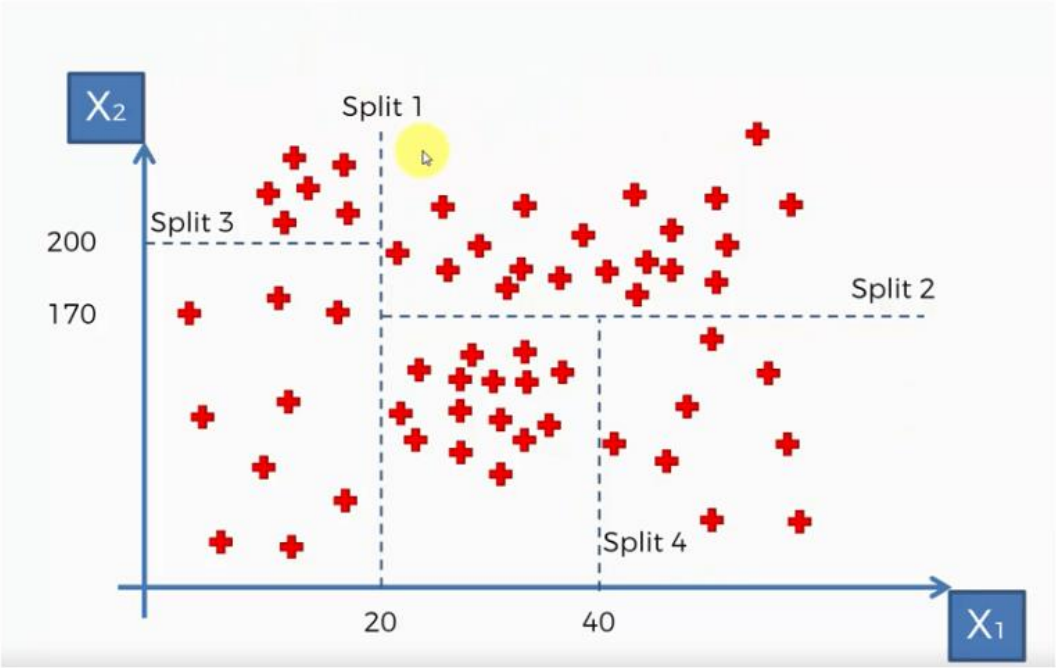
```
knr = KNeighborsRegressor()
knr.fit(xtrain,ytrain)
pred_train_knr=knr.predict(xtrain)
pred_test_knr=knr.predict(xtest)
print('K Neighbors Regressor Score:',knr.score(xtrain,ytrain))
print('K Neighbors Regressor r2_score:',r2_score(ytest,pred_test_knr))
print("Mean squared error of K Neighbors Regressor:",mean_squared_error(ytest,pred_test_knr))
print("Root Mean Square error of K Neighbors Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_knr)))
```

```
K Neighbors Regressor Score: 0.748266387098428
K Neighbors Regressor r2_score: 0.6042476227806619
Mean squared error of K Neighbors Regressor: 110138818466.21764
Root Mean Square error of K Neighbors Regressor: 331871.68976310355
```

The accuracy of K Neighbors Regressor is 60% which is okay.

5. Decision Tree Regressor

To get from observations about an item (represented in the branches) to inferences about the item's goal value, decision tree learning employs a decision tree (as a predictive model) (represented in the leaves). In statistics, data mining, and machine learning, it is modeling predictive modelling methodologies.



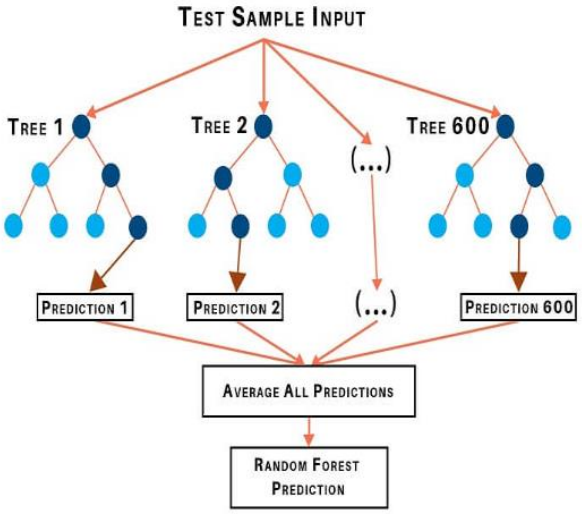
Classification trees are tree models in which thegoal variable can take a discrete set of values; in these tree structures, leaves indicate class labels and branches represent feature combinations that lead to those class labels. Regression trees are decision trees in which the target variable can take continuous values (usually real numbers). The objective is to build a model that predicts the value of a target variable from a set of input variables.

```
dtr=DecisionTreeRegressor(criterion='mse')
dtr.fit(xtrain,ytrain)
pred_train_dtr=dtr.predict(xtrain)
pred_test_dtr=dtr.predict(xtest)
print('Decision Tree Regressor Score:',dtr.score(xtrain,ytrain))
print('Decision Tree Regressor r2_score:',r2_score(ytest,pred_test_dtr))
print("Mean squared error of Decision Tree Regressor:",mean_squared_error(ytest,pred_test_dtr))
print("Root Mean Square error of Decision Tree Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_dtr)))
```

```
Decision Tree Regressor Score: 0.9979397374843745
Decision Tree Regressor r2_score: 0.8024097840748301
Mean squared error of Decision Tree Regressor: 54989822361.62706
Root Mean Square error of Decision Tree Regressor: 234499.08818932978
```

6. Random Forest Regressor

A regressor with a random forest. A random forest is a Meta estimator that employs averaging to increase predicted accuracy and control over-fitting a number of classification decision trees on various sub-samples of the dataset.



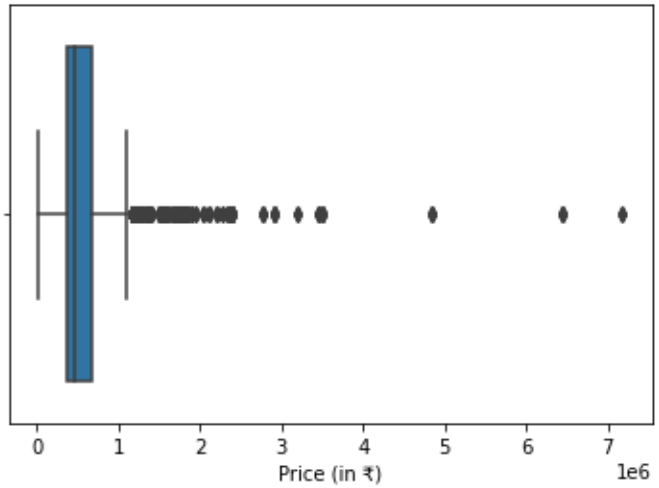

```
rf=RandomForestRegressor()
rf.fit(xtrain,ytrain)
pred_train_rf=rf.predict(xtrain)
pred_test_rf=rf.predict(xtest)
print('Random Forest Regressor Score:',rf.score(xtrain,ytrain))
print('Random Forest Regressor r2_score:',r2_score(ytest,pred_test_rf))
print("Mean squared error of Random Forest Regressor:",mean_squared_error(ytest,pred_test_rf))
print("Root Mean Square error of Random Forest Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_rf)))
```

```
Random Forest Regressor Score: 0.9806729731863234
Random Forest Regressor r2_score: 0.8756911676636284
Mean squared error of Random Forest Regressor: 34595440751.717865
Root Mean Square error of Random Forest Regressor: 185998.49663832734
```

7. Visualizations

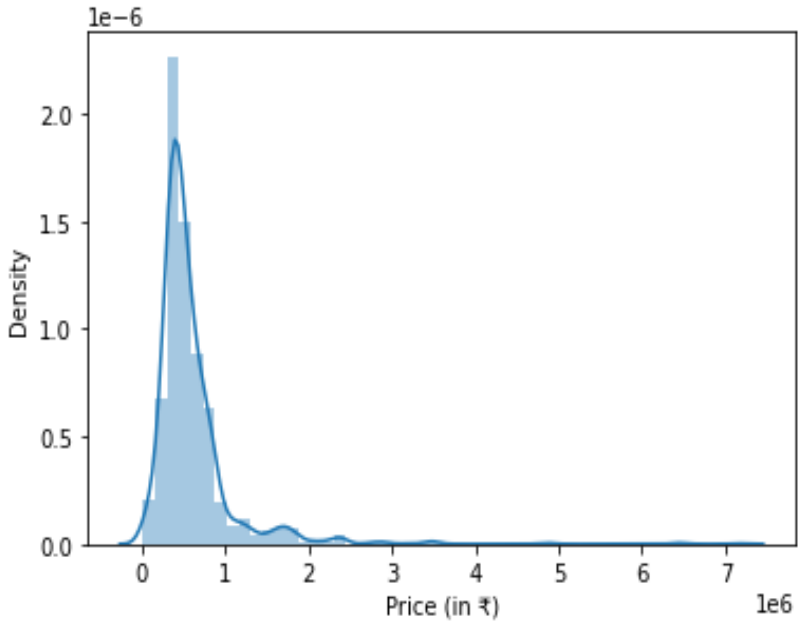
We have plotted histograms and distribution plot in univariate analysis,which interpreted that all the columns are equally important but the columns like brand, variant, location, date and total driven kilometers have a wide range of data spread hence we will not perform it's bivariate analysis.

```
In [13]: sn.boxplot(df['Price (in ₹)'])
Out[13]: <AxesSubplot:xlabel='Price (in ₹)'>
```



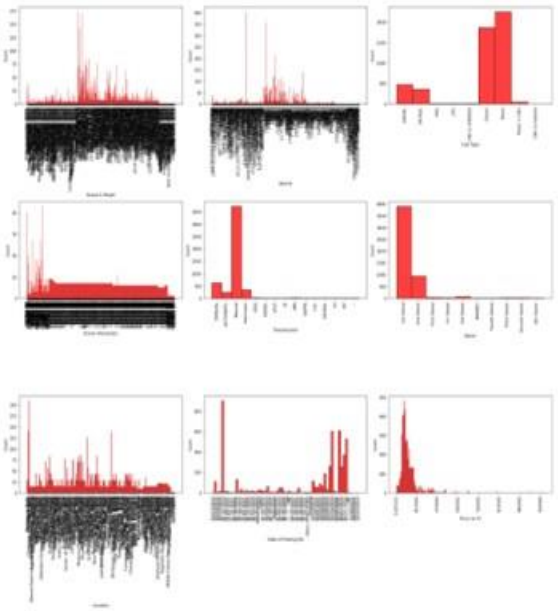
There are many outliers but since it's the target variable, hence we will not treat the outliers.

```
In [14]: sn.distplot(df['Price (in ₹)'])
Out[14]: <AxesSubplot:xlabel='Price (in ₹)', ylabel='Density'>
```



The data is very tightly distributed here and is almost normalized.

HISTOGRAM



~ Brands, Varients, Driven Kilometers & Location have a wide range of values in them.

~ Maximum Cars run on either Petrol or diesel. Only few goes for CNG and other fuels.

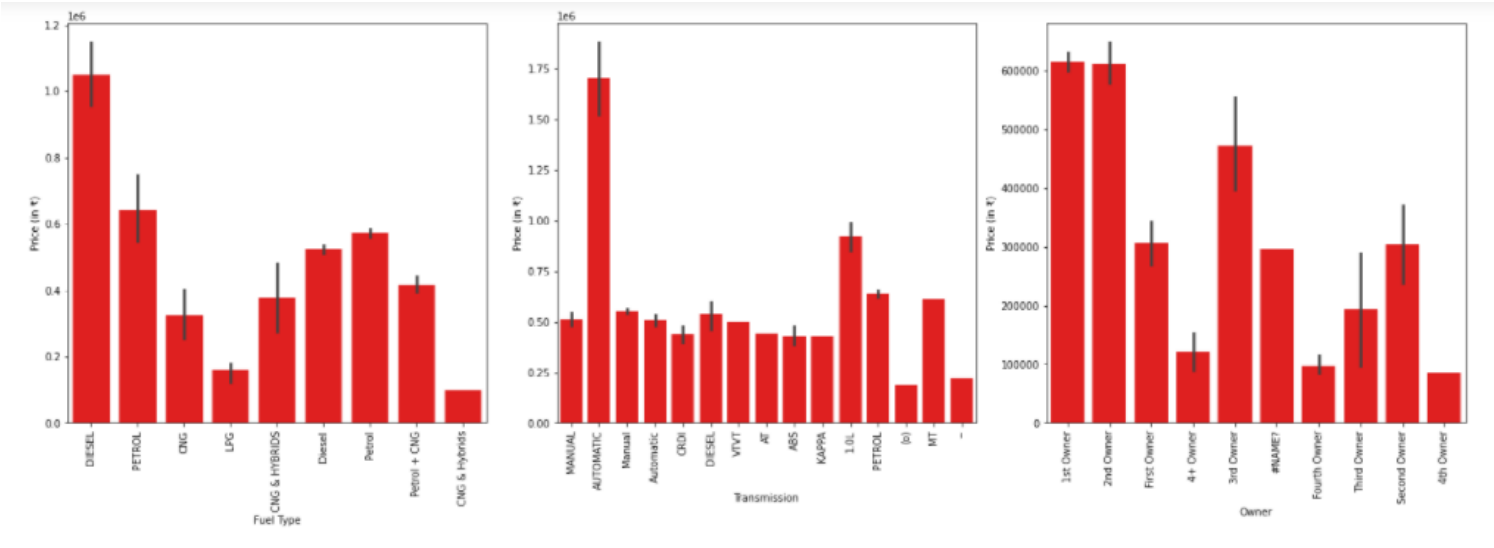
~ Maximum Cars have Manual transnission.

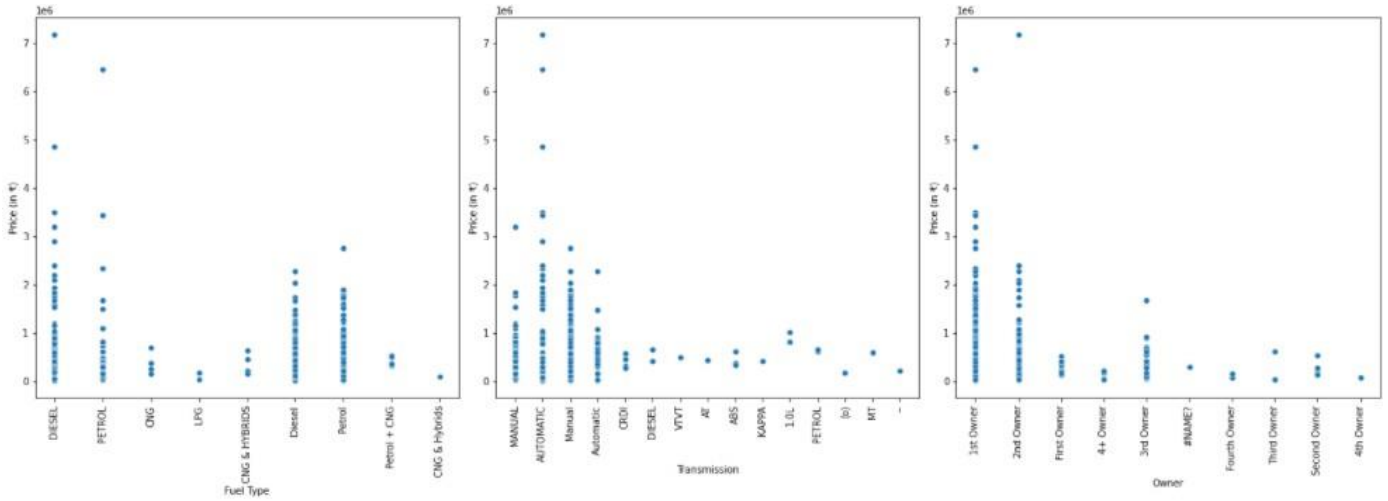
~ Maximum cars are being sold by their very 1st Owner.

~ We have collected the cars posted online in last one month, from 25th December 2021 to 27th January 2022.

~ Almost all the cars have a price rnging in between 270000 to 1165101.

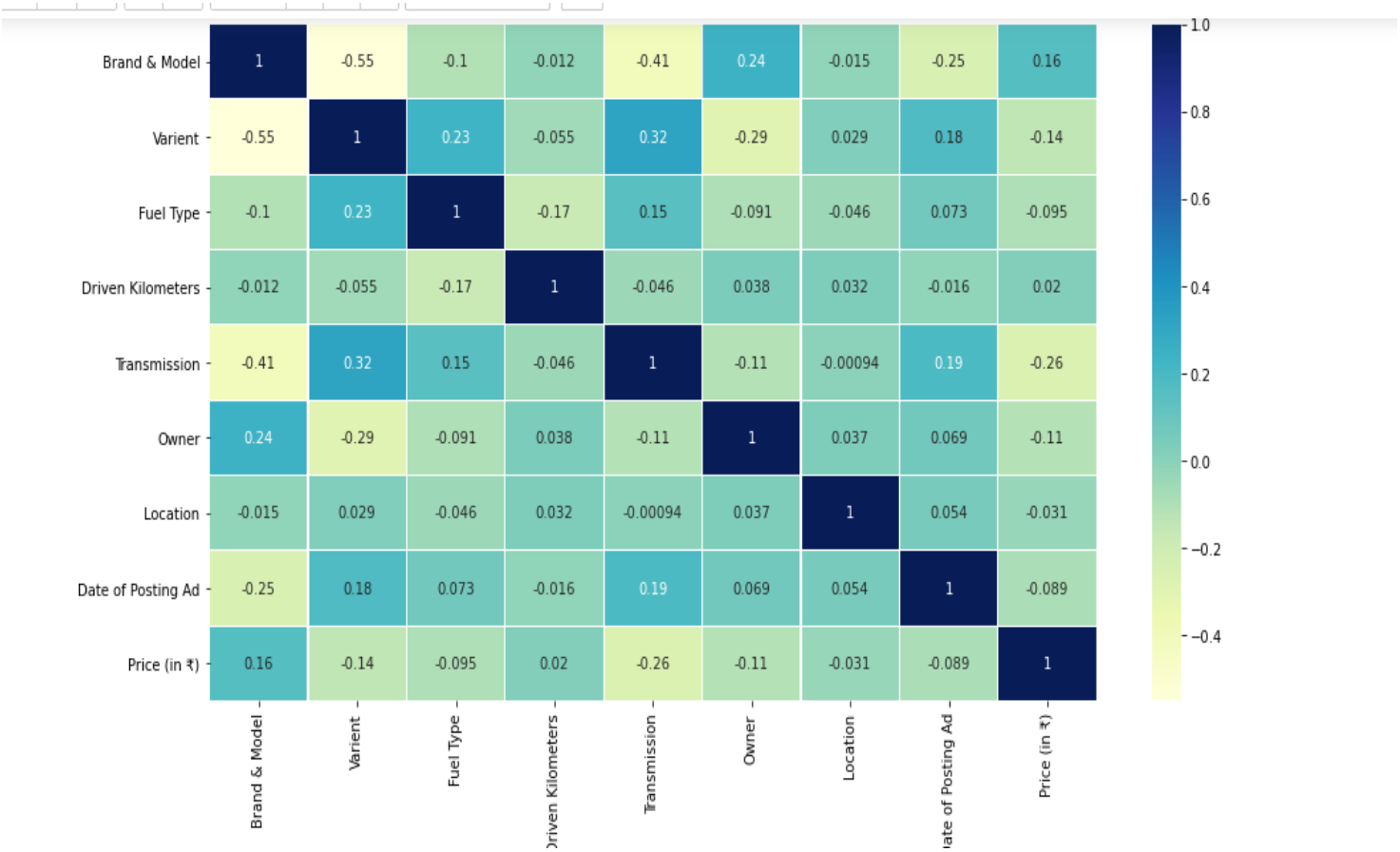
From bivariate analysis we conclude that, Since Brands, variants, Driven Kilometers & Location have a wide range of values in them, we will not perform bivariate analysis for them as they will not give us any specific details. Now by plotting graph of Fuel type, Transmission and Owner against Price, we conclude that Car that uses Diesel, have automatic Transmission and Has only 1 owner is more likely to have a high price.





Just like bar graph, we can see that Price range is likely to be high for cars using Diesel as fuel, or having Automatic Transmission or is owned by only 1 Owner.

The multivariate analysis done by plotting heat map says that there is no multicollinearity in the **dataset.**



Chapter 8

Implementation (Coding)

1. Web Scraping

In [ ]:

```
import time  #importing libraries
import selenium
import pandas as pd
from selenium import webdriver
from selenium.webdriver.firefox.service import Service
from selenium.webdriver.common.by import By
s=Service(r"C:\Browserdrivers\geckodriver.exe")  #opening firefox
driver = webdriver.Firefox(service=s)
driver.maximize_window()
driver.get("https://www.olx.in/cars_c84")  #getting the webpage of olx

brand=[] #empty list
varient=[] #empty list
manyear=[] #empty list
kms=[] #empty list
date=[] #empty list
fuel=[] #empty list
nowners=[] #empty list
loc=[] #empty list
transmission=[] #empty list
color=[] #empty list
sellertype=[] #empty list
price=[] #empty list
url=[] #empty list

time.sleep(2)

#fetching urls of each car to get more data
for k in range(45):

    urls=driver.find_elements(By.XPATH,"//li[@class='_31j8e']/a")
    for i in urls[:]:
        url.append(i.get_attribute('href'))

    if k==45:
        break

    #clicking on load more button
    nxt_button = driver.find_element(By.XPATH,"//div[@class='JbJAl']/button")
    nxt_button.click()
    time.sleep(5)

for i in url:      #fetching data from Each Car's URL
    driver.get(i)
    time.sleep(2)

    #fetching brand, model & manufacturing year of the car
    b=driver.find_element(By.XPATH,"//div[@class='_35xN1']")
    brand.append(b.text)

    #fetching the variant of the car
    v=driver.find_element(By.XPATH,"//div[@class='_3tLee']")
    varient.append(v.text)

    #fetching the posting ad date of the car
    d=driver.find_element(By.XPATH,"//div[@class='_1l939']/div[3]/div[2]/div[2]")
    date.append(d.text)

    #fetching the fuel type of the car
    f=driver.find_element(By.XPATH,"//div[@class='aOxkz']/div[1]/div")
    fuel.append(f.text)

    #fetching the total number of driven kms of the car
    k=driver.find_element(By.XPATH,"//div[@class='aOxkz']/div[2]/div")
    kms.append(k.text)

    #fetching the number of previous owners of the car
    no=driver.find_element(By.XPATH,"//div[@class='_1l939']/div[1]/div[2]/div[2]")
    nowners.append(no.text + ' Owner')

    #fetching the location of seller of the car
    l=driver.find_element(By.XPATH,"//div[@class='_1l939']/div[2]/div[2]/div[2]")
    loc.append(l.text)

    #fetching the transmission of the car
    t=driver.find_element(By.XPATH,"//div[@class='aOxkz']/div[3]/div")
    transmission.append(t.text)

    #fetching the seller of the car
    s=driver.find_element(By.XPATH,"//span[@class='_1hYGL']")
    sellertype.append(s.text)

    #fetching the price of the car
    p=driver.find_element(By.XPATH,"//div[@class='_3FkyT']")
    price.append(p.text)
```

```
driver.get("https://www.cars24.com/buy-used-cars/")  #getting the webpage of cars24

#clicking the view all button
viewall=driver.find_element(By.XPATH,"//div[@class='_2AvHl']/button/span").click()

#fetching brand along with model
b=driver.find_elements(By.XPATH,"//h2[@class='_3FpCg']")
for i in b:
    brand.append(i.text)

#fetching the variant of the car
v=driver.find_elements(By.XPATH,"//p[@class='cvakB']")
for i in v:
    i=i.text.split(' ')[:-1]
    varient.append(i)

#fetching the total driven kilometers
km=driver.find_elements(By.XPATH,"//ul[@class='bVR0c']/li[1]")
for i in km:
    i=i.text
    kms.append(i)

#fetching the transmission of the car
t=driver.find_elements(By.XPATH,"//p[@class='cvakB']")
for i in t:
    i=i.text.split(' ')[-1]
    transmission.append(i)

#fetching the number of previous owners of the car
no=driver.find_elements(By.XPATH,"//ul[@class='bVR0c']/li[2]")
for i in no:
    nowners.append(i.text)

#fetching the fuel type of the car
f=driver.find_elements(By.XPATH,"//ul[@class='bVR0c']/li[3]")
for i in f:
    fuel.append(i.text)

#fetching urls of car
urls=driver.find_elements(By.XPATH,"//a[@class='_9Ue0B']")
for i in urls[:]:
    url.append(i.get_attribute('href'))

#fetching the price of the car
p=driver.find_elements(By.XPATH,"//div[@class='_7udZZ']/span")
for i in p:
    price.append(i.text)

time.sleep(2)

for i in url:      #fetching data from Each Car's URL
    driver.get(i)
    time.sleep(2)

    #fetching the location of seller of the car
    l=driver.find_element(By.XPATH,"//div[@class='media-body _1PZm9']/strong")
    loc.append(l.text)


driver.get("https://autoportal.com/usedcars/used-cars-in-mumbai/")  #getting the webpage of autoportal

time.sleep(2)

#fetching urls of each car to get more data
for k in range(30):

    urls=driver.find_elements(By.XPATH,"//div[@class='desc']/a")
    for i in urls[:]:
        url.append(i.get_attribute('href'))

    if k==30:
        break

    #clicking on load more button
    nxt_button = driver.find_element(By.XPATH,"//span[@class='next']/a")
    nxt_button.click()
    time.sleep(5)

for i in url:      #fetching data from Each Car's URL
    driver.get(i)
    time.sleep(2)

    #fetching brand, model & manufacturing year of the car
    b=driver.find_element(By.XPATH,"//h1")
    brand.append(b.text)

    #fetching the variant of the car
    v=driver.find_element(By.XPATH,"//div[@class='row-sm'][1]/div[1]")
    varient.append(v.text)
```

```
    #fetching the posting ad date of the car
    d=driver.find_element(By.XPATH,"//div[@class='updated']")
    date.append(d.text)

    #fetching the fuel type of the car
    f=driver.find_element(By.XPATH,"//div[@class='row-sm'][1]/div[4]")
    fuel.append(f.text)

    #fetching the total number of driven kms of the car
    k=driver.find_element(By.XPATH,"//div[@class='row-sm'][1]/div[2]")
    kms.append(k.text)

    #fetching the number of previous owners of the car
    no=driver.find_element(By.XPATH,"//div[@class='row-sm'][1]/div[3]")
    nowners.append(no.text + ' Owner')

    #fetching the location of seller of the car
    l=driver.find_element(By.XPATH,"//div[@class='row-sm'][1]/div[5]")
    loc.append(l.text)

    #fetching the transmission of the car
    t=driver.find_element(By.XPATH,"//div[@class='row-sm'][1]/div[6]")
    transmission.append(t.text)

    #fetching the price of the car
    p=driver.find_element(By.XPATH,"//div[@class='item price-in']")
    price.append(p.text)


driver.get("https://www.cardekho.com/used-cars")  #getting the webpage of cardekho

viewcars=driver.find_element(By.XPATH,"//a[@class='button']").click() #clicking the button to view all 1989 car

#fetching brand along with model
b=driver.find_elements(By.XPATH,"//div[@class='gsc_col-xs-7  carsName']/a")
for i in b:
    brand.append(i.text)

#fetching the variant of the car
v=driver.find_elements(By.XPATH,"//div[@class='gsc_col-xs-7  carsName']/div")
for i in v:
    varient.append(i)

#fetching the total driven kilometers
km=driver.find_elements(By.XPATH,"//div[@class='truncate dotlist']/span[1]")
for i in km:
    i=i.text
    kms.append(i)

#fetching the transmission of the car
t=driver.find_elements(By.XPATH,"//div[@class='truncate dotlist']/span[3]")
for i in t:
    i=i.text.split(' ')[-1]
    transmission.append(i)

#fetching the fuel type of the car
f=driver.find_elements(By.XPATH,"//div[@class='truncate dotlist']/span[2]")
for i in f:
    fuel.append(i.text)

#fetching urls of car
urls=driver.find_elements(By.XPATH,"//div[@class='gsc_col-xs-7  carsName']/a")
for i in urls[:]:
    url.append(i.get_attribute('href'))

#fetching the price of the car
p=driver.find_elements(By.XPATH,"//span[@class='amnt']")
for i in p:
    price.append(i.text)

time.sleep(2)

for i in url:      #fetching data from Each Car's URL
    driver.get(i)
    time.sleep(2)

    #fetching the location of seller of the car
    l=driver.find_element(By.XPATH,"//div[@class='_1PZm9']")
    loc.append(l.text)

    #fetching the number of previous owners of the car
    no=driver.find_element(By.XPATH,"//ul[@class='gsc_row detailsList']/li[6]/div[1]/div[2]")
    nowners.append(no.text)


Car = pd.DataFrame({})   #creating a datafrmae
Car['Brand & Model']=brand
Car['Varient']=varient
```

```
Car['Fuel Type']=fuel
Car['Driven Kilometers']=kms
Car['Transmission']=transmission
Car['Owner']=nowners
Car['Location']=loc
Car['Date of Posting Ad']=date
Car['Price (in ₹)']=price
```

In [ ]:
```
Car.to_csv('CarPrice.csv',index=False)    #saving the dataset in csv format
```


Data is collected from various sources using selenium and saved in CSV file. Data is scraped for approx. 5000 cars and 9 features each form websites. So this dataset contain ~5000 rows with 9 columns. Price is our Target variable which is to be predicted. We can see that some of numerical variable like torque, mileage are by default come with object data types.

2. Car price prediction

Used Car Price Prediction using Machine Learning

In [1]:
```
import numpy as np
import pandas as pd
import scipy
from scipy.stats import zscore
import matplotlib.pyplot as plt
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import power_transform
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import cross_val_score as cvs
from sklearn.model_selection import GridSearchCV
import seaborn as sn
import warnings
warnings.filterwarnings('ignore')
```

```
Necessary libraries successfully imported.
```

Importing All the necessary libraries.

Reading and Understanding the (Scraped) Data

In [2]:
```
df = pd.read_csv('CarPrice.csv')
```

Out[2]:

	Brand & Model	Varient	Fuel Type	Driven Kilometers	Transmission	Owner	Location	Date of Posting Ad
0	Mahindra Xuv500 (2013)	W8 Dual Tone	DIESEL	58,000 KM	MANUAL	1st Owner	Pitampura, Delhi	01/27/2022
1	Hyundai Creta (2020)	1.6 SX Option Executive Diesel	DIESEL	43861.0 KM	MANUAL	1st Owner	Ahiritola, Kolkata	01/23/2022

	Brand & Model	Varient	Fuel Type	Driven Kilometers	Transmission	Owner	Location	Date of Posting Ad
2	Hyundai Verna (2019)	VTVT 1.4 EX	PETROL	17,000 KM	MANUAL	2nd Owner	Chelavoor, Pantheeramkavu	01/25/2022

Here we read the Excel file in jupyter notebook.

In [3]:
```
df = pd.DataFrame(data=df)
```

Out[3]:

	Brand & Model	Varient	Fuel Type	Driven Kilometers	Transmission	Owner	Location	Date of Posting Ad	Pr (i
5045	maruti suzuki 800 (1970)	EX 5 Speed	LPG	500,000 Km	Automatic	1st Owner	Alipur	01/23/2022	50
5046	renault duster (2012)	2012-2015 110PS Diesel RxZ	Diesel	111,000 Km	Manual	1st Owner	MDDA Colony Near Kedarpuram, Dehradun	01/23/2022	320
5047	hyundai santro (2021)	Sportz AMT	Petrol	5,500 Km	Automatic	1st Owner	Pragati Nagar, Ahmedabad	01/23/2022	599
5048	hyundai verna (2013)	VTVT 1.6 EX	Petrol	61231.0 Km	Manual	1st Owner	Malad West, Mumbai	01/27/2022	395
5049	maruti suzuki swift dzire (2019)	VXI	Diesel	58,000 Km	Manual	1st Owner	Kukatpally, Hyderabad	01/23/2022	580

Here we are loading the dataset into DataFrame.

Data Inspection

In [4]:

Out[4]: (5050, 9)

There are 5050 rows and 9 columns in the dataset.

In [5]:

Out[5]:

```
Brand & Model         object
Varient               object
Fuel Type             object
Driven Kilometers     object
Transmission          object
```

All the columns are of object datatype except the target variable, 'Price (in ₹)' which is of integer data type.

In [6]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5050 entries, 0 to 5049
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Brand & Model       5050 non-null   object
 1   Varient             5011 non-null   object
 2   Fuel Type           5050 non-null   object
 3   Driven Kilometers   5050 non-null   object
 4   Transmission        5050 non-null   object
 5   Owner               5050 non-null   object
 6   Location            5050 non-null   object
 7   Date of Posting Ad  5050 non-null   object
 8   Price (in ₹)        5050 non-null   int64
dtypes: int64(1), object(8)
memory usage: 355.2+ KB
```

There are 9 columns and varient column have null values and all the columns are of object data type except 'Price (in ₹)' which is of integer data type. The total memory usage of this dataset is 355.2 KB.

In [7]:

Out[7]:
```
Index(['Brand & Model', 'Varient', 'Fuel Type', 'Driven Kilometers',
       'Transmission', 'Owner', 'Location', 'Date of Posting Ad',
       'Price (in ₹)'],
      dtype='object')
```

There are 9 columns:

1. 'Brand & Model' : It gives us the brand of the car alongwith its model name and manufacturing year
2. 'Varient' : It gives us the varient of particular car model
3. 'Fuel Type' : It gives us the type of fuel used by the car
4. 'Driven Kilometers' : It gives us the total distance in kms covered by car
5. 'Transmission' : It tells us whether the gear transmission is Manual or Automatic
6. 'Owner' : It tells us the total numbers of owners car had previously
7. 'Location' : It gives us the locaion of the car
8. 'Date of Posting Ad' : It teels us when the advertisement for selling that car was posted online
9. 'Price (in ₹)' : It gives us the price of the car.

Here 'Price (in ₹)' is our target variable.

Data Cleaning

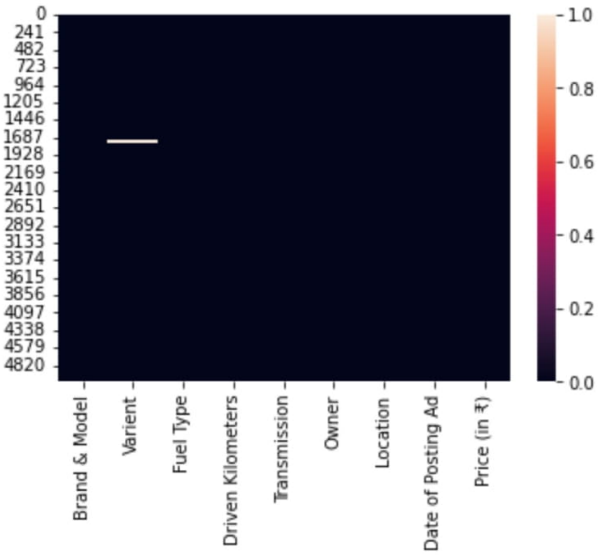
In [8]:
```
#Checking for Null values
```

Out[8]:
```
Brand & Model          0
Varient               39
Fuel Type              0
Driven Kilometers      0
Transmission           0
Owner                  0
Location               0
Date of Posting Ad     0
Price (in ₹)           0
dtype: int64
```

There are 39 null values in column 'Varient' and since it is of object type, we will use Mode to fill the rows with null value.

In [9]:

Out[9]: <AxesSubplot:>



This is the visualization of having just a small number null values in the dataset.

In [10]:
```
from sklearn.impute import SimpleImputer
imp=SimpleImputer(missing_values=np.nan,strategy='most_frequent')
df['Varient']=imp.fit_transform(df['Varient'].values.reshape(-1,1))
```

Treating the null values with mode using Simple Imputer

In [11]:
```
#Checking for Null values
```

Out[11]:
```
Brand & Model         0
Varient               0
Fuel Type             0
Driven Kilometers     0
Transmission          0
Owner                 0
Location              0
Date of Posting Ad    0
Price (in ₹)          0
dtype: int64
```

Now there are no null values in the dataset.

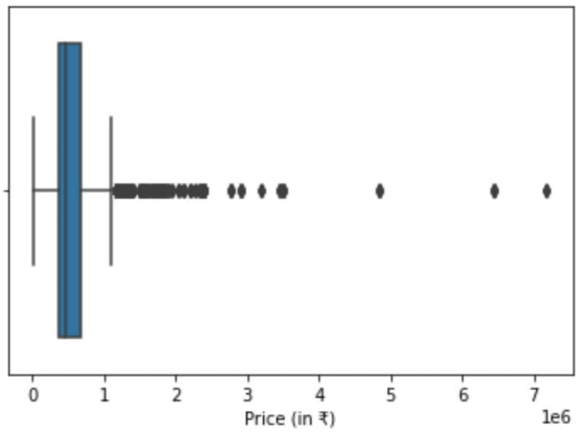
In [12]:

Since all the variables are of object data type, we will not check for outliers or skewness.

Exploratory Data Analysis

In [13]:

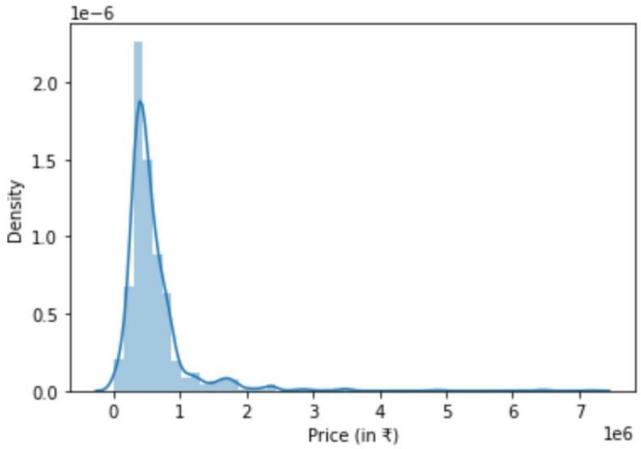
Out[13]:
```
<AxesSubplot:xlabel='Price (in ₹)'>
```



There are many outliers but since it's the target variable, hence we will not treat the outliers.

In [14]:

Out[14]: <AxesSubplot:xlabel='Price (in ₹)', ylabel='Density'>



The data is very tightly distributed here and is almost normalized.

In [ ]:
```
collist=df.columns.values
plt.figure(figsize=(20,20))
for i in range(0,len(collist)):
    plt.subplot(3,3,i+1)
    sn.histplot(data=df[collist[i]],color='red').set_xticklabels(labels=df[col
```

~ Brands, Varients, Driven Kilometers & Location have a wide range of values in them.

~ Maximum Cars run on either Petrol or diesel. Only few goes for CNG and other fuels.

~ Maximum Cars have Manual transnission.

~ Maximum cars are being sold by their very 1st Owner.

~ We have collected the cars posted online in last one month, from 25th December 2021 to 27th January 2022.

~ Almost all the cars have a price rnging in between 270000 to 1165101.

In [ ]:
```
newcollist=['Fuel Type','Transmission','Owner']
plt.figure(figsize=(20,20))
for i in enumerate(newcollist):
    plt.subplot(3,3,i[0]+1)
    sn.barplot(data=df,x=i[1],y='Price (in ₹)',color='red')
    plt.xticks(rotation=90)
```

Since Brands, Varients, Driven Kilometers & Location have a wide range of values in them, we will not perform bivariate analysis for them as they will not give us any specific details. Now by plotting graph of Fuel type, Transmission and Owner against Price, we conclude that Car that

uses Diesel, have automatic Transmission and Has only 1 owner is more likely to have a high price

In [ ]:
```
newcollist=['Fuel Type','Transmission','Owner']
plt.figure(figsize=(20,20))
for i in enumerate(newcollist):
    plt.subplot(3,3,i[0]+1)
    sn.scatterplot(data=df,x=i[1],y='Price (in ₹)')
    plt.xticks(rotation=90)
    plt.tight_layout()
```

Just like bar graph, we can see that Price range is likely to be high for cars using Diesel as fuel, or having Automatic Transmission or is owned by only 1 Owner.

In [ ]:
```
le = LabelEncoder()
for column in df.drop(['Price (in ₹)'],axis=1).columns:
    df[column]=le.fit_transform(df[column])
```

Transforming the data from object to ordinal type.

Multivariate Analysis

In [ ]:
```
plt.figure(figsize=(14,10))
sn.heatmap(round(df.describe()[1:].transpose(),2),lw=2,linecolor='black',annot
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.title('Variable Summary')
```

It gives us the statistical summary(which includes mean, median, standard deviation, minimum value, maximum value and quartile values) of all the numerical columns.

In [ ]:
```
plt.figure(figsize=(15,8))
```

There is no multicollinearity in the dataset.

Model Building

In [ ]:
```
x= df.drop(['Price (in ₹)'],axis=1)
```

Separating feature and target variables into x and y.

In [ ]:
```
x=power_transform(x,method='yeo-johnson')
scale = StandardScaler()
```

Power transforming and scaling the feature variables.

In [ ]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_
svr = SVR()
svr.fit(xtrain,ytrain)
pred_train_svr=svr.predict(xtrain)
pred_test_svr=svr.predict(xtest)
print('SVR Regressor Score:',svr.score(xtrain,ytrain))
print('SVR Regressor r2_score:',r2_score(ytest,pred_test_svr))
print("Mean squared error of SVR Regressor:",mean_squared_error(ytest,pred_tes
```

The Accuracy of SVR is in negative which stats that this is not the correct model to apply here.

In [ ]:
```
lr= LinearRegression()
lr.fit(xtrain,ytrain)
lr.coef_
pred_train=lr.predict(xtrain)
pred_test=lr.predict(xtest)
print('Linear Regression Score:',lr.score(xtrain,ytrain))
print('Linear Regression r2_score:',r2_score(ytest,pred_test))
print("Mean squared error of Linear Regression:",mean_squared_error(ytest,pred
```

The accuracy of Linear Regression is only 6%

In [ ]:
```
sgd=SGDRegressor()
sgd.fit(xtrain,ytrain)
pred_train_sgd=sgd.predict(xtrain)
pred_test_sgd=sgd.predict(xtest)
print('SGD Regressor Score:',sgd.score(xtrain,ytrain))
print('SGD Regressor r2_score:',r2_score(ytest,pred_test_sgd))
print("Mean squared error of SGD Regressor:",mean_squared_error(ytest,pred_tes
```

The accuracy of SGD Regressor is also very poor, it's only 6%

In [ ]:
```
knr = KNeighborsRegressor()
knr.fit(xtrain,ytrain)
pred_train_knr=knr.predict(xtrain)
pred_test_knr=knr.predict(xtest)
print('K Neighbors Regressor Score:',knr.score(xtrain,ytrain))
print('K Neighbors Regressor r2_score:',r2_score(ytest,pred_test_knr))
print("Mean squared error of K Neighbors Regressor:",mean_squared_error(ytest,
```

The accuracy of K Neighbors Regressor is 60% which is okay.

In [ ]:
```
dtr=DecisionTreeRegressor(criterion='mse')
dtr.fit(xtrain,ytrain)
pred_train_dtr=dtr.predict(xtrain)
pred_test_dtr=dtr.predict(xtest)
print('Decision Tree Regressor Score:',dtr.score(xtrain,ytrain))
print('Decision Tree Regressor r2_score:',r2_score(ytest,pred_test_dtr))
print("Mean squared error of Decision Tree Regressor:",mean_squared_error(ytes
print("Root Mean Square error of Decision Tree Regressor:",np.sqrt(mean_square
```

The accuracy of Decision Tree Regressor is 80.2 % which is in acceptable range.

In [ ]:
```
rf=RandomForestRegressor()
rf.fit(xtrain,ytrain)
pred_train_rf=rf.predict(xtrain)
pred_test_rf=rf.predict(xtest)
print('Random Forest Regressor Score:',rf.score(xtrain,ytrain))
print('Random Forest Regressor r2_score:',r2_score(ytest,pred_test_rf))
print("Mean squared error of Random Forest Regressor:",mean_squared_error(ytes
```

The accuracy of Random Forest Regressor is 87.56% which is quite good.

Cross Validation Score

In [ ]:
```
print('Cross Validation Score of SVR is',(cvs(svr,x,y,cv=5).mean())*100)
print('Cross Validation Score of Linear Regression is',(cvs(lr,x,y,cv=5).mean(
print('Cross Validation Score of SGD Regressor is',(cvs(sgd,x,y,cv=5).mean())*
print('Cross Validation Score of KNeighbors Regressor is',(cvs(knr,x,y,cv=5).m
print('Cross Validation Score of Decision Tree Regressor is',(cvs(dtr,x,y,cv=5
```

After comparing r2_score and Cross validation score, we will select Random Forest Regressor for Hyper Parameter Tuning.

Hyper Parameter Tuning

In [ ]:
```
parameter = { 'bootstrap': [True, False],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],}

gvc = GridSearchCV(RandomForestRegressor(),parameter,cv=5)
gvc.fit(xtrain,ytrain)
```

Getting all the best parameter to apply in our selected model.

In [ ]:
```
pricecar = RandomForestRegressor(bootstrap=False,min_samples_leaf=1,max_featur
pricecar.fit(xtrain,ytrain)
pred=pricecar.predict(xtest)
acc=r2_score(ytest,pred)
print('Score of Hyper Parameter Tuned Ranfom Forest Regressor is:',pricecar.sc
print('Accuracy for predicting price of car is', (acc*100),'%')
print("Mean squared error of Hyper Parameter Tuned Random Forest Regressor:",m
```

The accuracy of Model 'PriceCar' (Random Forest Regressor) after applying Hyper Tuned Parameters is found to be 87.79% and the score is 0.98 which is quite good.

Conclusion

In [ ]:
```
a= np.array(ytest)
predicted = np.array(pricecar.predict(xtest))
Price=pd.DataFrame({"Original":a,"Predicted":predicted},index=range(len(a)))
```

Here, we can see that all the predicted prices are either equal or nearly equal to the original prices of the car. Hence we conclude that our model 'pricecar' is working very well. And we shall save it for further use.

Model Saving

In [ ]:
```
import pickle
filename = 'PriceCar.pkl'
```

We have saved our best model.

Chapter 9

User Manual

Installing Python and Jupyter Notebook

Installing Python:

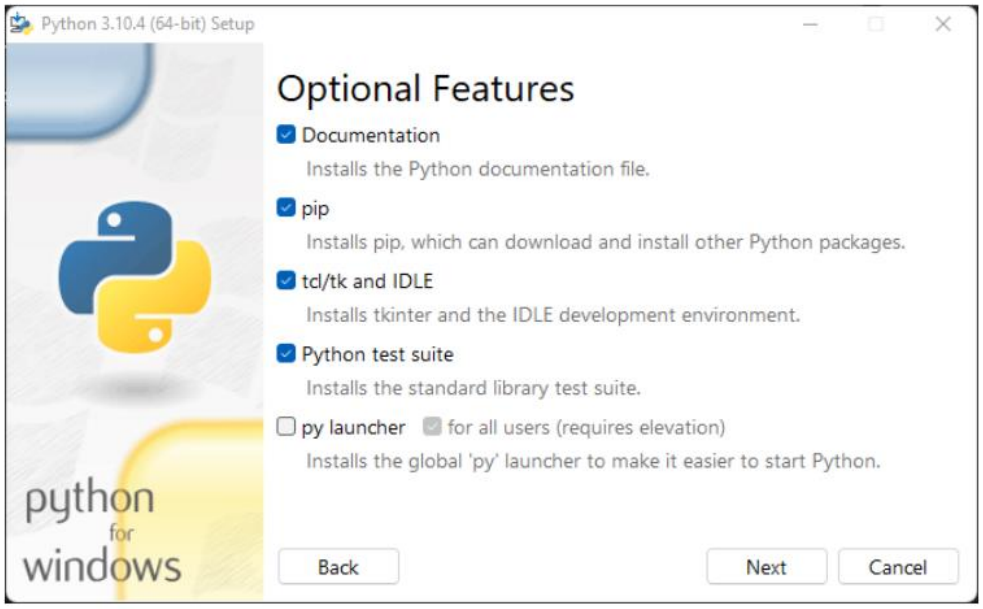
First of all, we need to go to the official website of Python.

Click on the **Downloads** section.

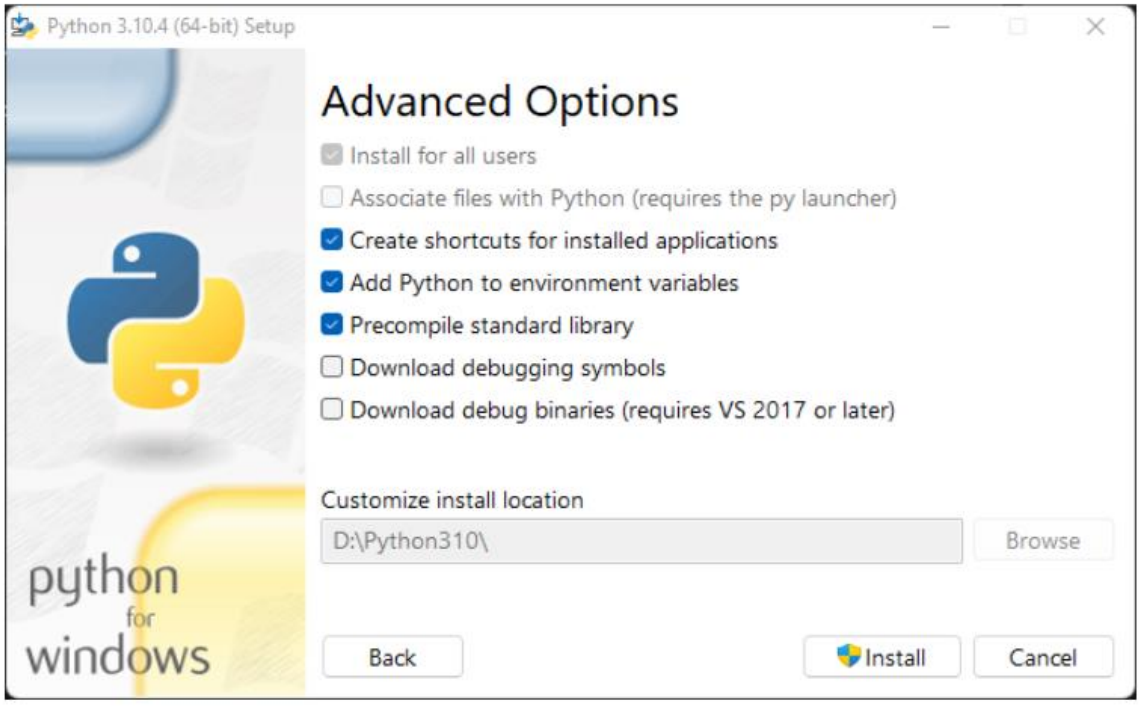
Here you will get the latest version. Just click on the Download `Python 3.10.4`. By the time you are reading this article, Python might have been updated, in which case the version would be different. Simply download the version it shows you.

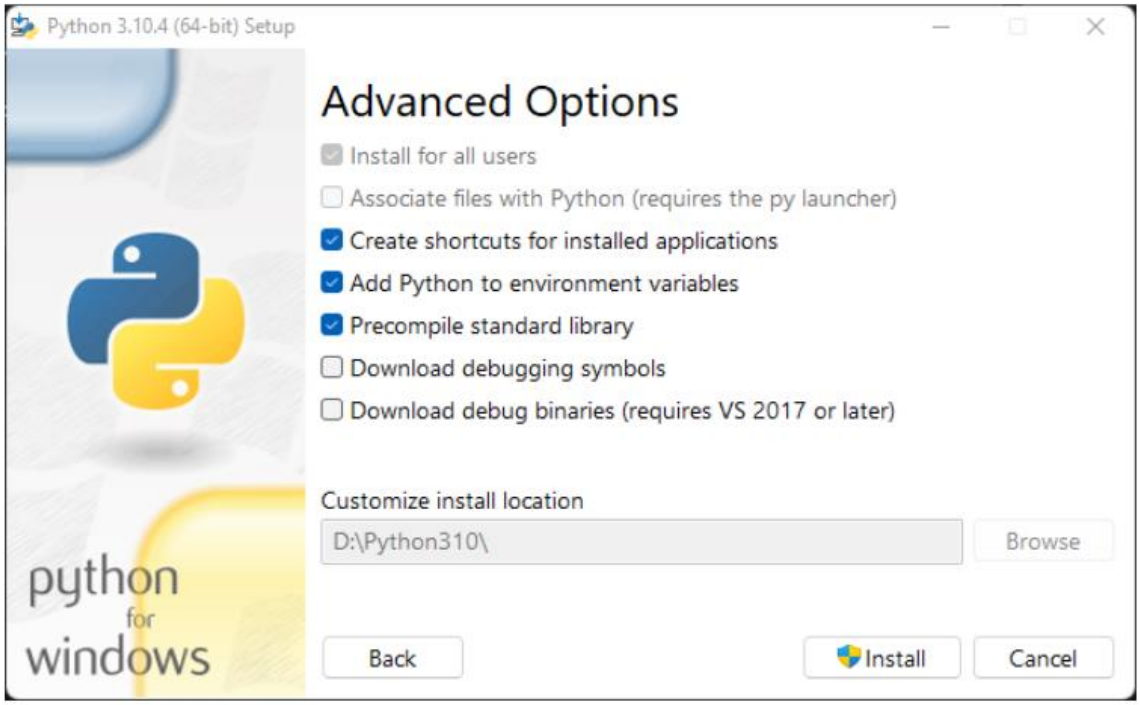
After downloading the file, we will get an executable file like this. Simply double click on that file and the installation wizard will open.

Click on `Customize installation`



Make sure to check all of the boxes, like above. Then click `Next`.





Installing Jupyter Notebook using pip:

PIP is a package management system used to install and manage software packages/libraries written in Python.

To install Jupyter using pip, we need to first check if pip is updated in our system. Use the following command to update pip:

Open Command prompt in Windows

```
python -m pip install --upgrade pip
```

After updating the pip version, follow the instructions provided below to install Jupyter:

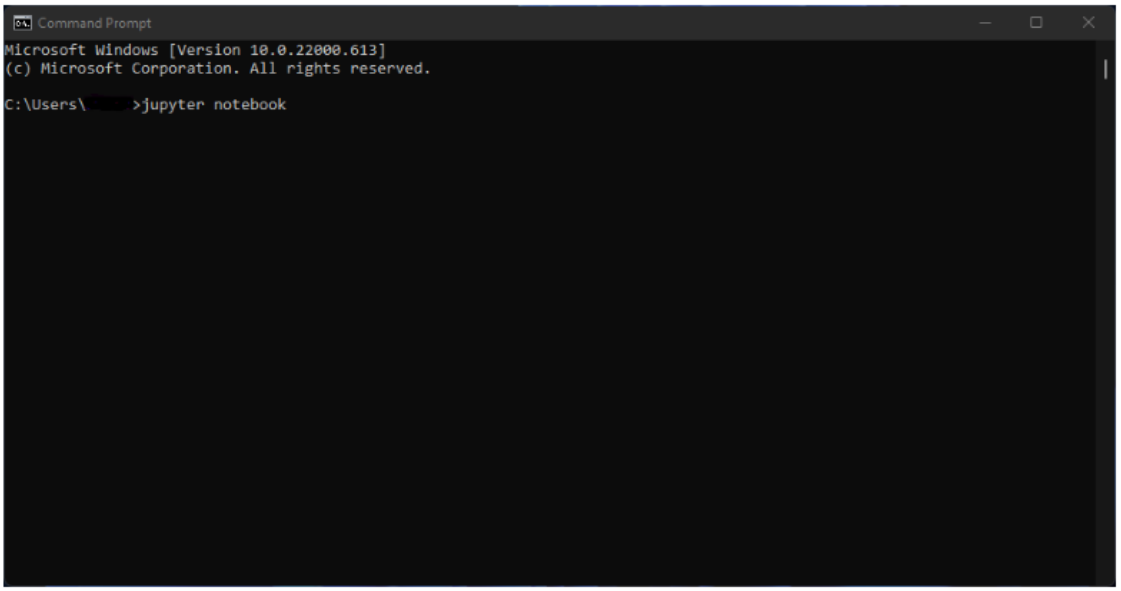
```
python -m pip install jupyter
```

Thats it!

Launching Jupyter Notebook:

Use the following command to launch Jupyter using command-line:

```
jupyter notebook
```



Running the Project:

1. Install python packages – Selenim, NumPy, Pandas, Time, Scikit-learn, Scipy, Matplotlib.
2. Open cmd, type jupyter notebook to open notebook.
3. Navigate to the project directory, select the web scraping.ipynb file to initialize web scraping script.
4. Web Scraping script requires geckodriver, which needs to be downloaded, here it has been enclosed with the project itself.
5. Once the scraping script finishes running and saves the dataset into a csv file, select and open the Car Price Prediction Using Machine Learning.ipynb file.
6. Run and obtain the results for analysis.

Chapter 10

Conclusion

Key Findings and Conclusions of the Study

The Prediction of car prices has aroused the interest of researchers as it requires a considerable amount of work and experience at the hands of specialists in the sector. For a prediction with reliability and accuracy, we analyzed a considerable number of attributes that were unique. We used six different ML approaches to discover the best and develop a used car price prediction model.

The respective performances of different algorithms were then compared to discover the one that best suited the existing data set. The final model chosen for the car price prediction was implemented in a python program.

The various algorithms were then put up in comparison to each other on the basis of the performances respectively to find and choose the one that suited the best for our use case of the gathered dataset.

Also, the model was tested with test data, yielding an accuracy of **87.76** percent (for the then obtained and considered dataset).

Learning Outcomes of the Study in respect of DataScience

Employing popular and sophisticated algorithms from various Python libraries, algorithmic paradigms based on machine learning could be successfully constructed through the scope of this project. Data pre-processing and data-cleaning are performed as the beginning steps on the dataset. We trimmed the tuples that contained null values, which accounted for less than 1% of the total. The discovery revealed a positive relationship between the price and the miles travelled, also of the year of registration of the vehicle and the kilometers travelled.

Negative correlation is related to the notion of inverse proportion, on the other hand, positive correlation is related to the concept of direct proportion. The model was trained using over 5000 tuples.

Chapter 11

Limitations of this project and Scope for Future Improvements

As a part of future work, we aim at the variable choices over the algorithms that were used in the project. We could only explore two algorithms whereas many other algorithms which exist and might be more accurate. More specifications will be added in a system or providing more accuracy in terms of price in the system that are

As part of future work, we were looking at various options for the algorithms chosen and used in this project. We were only able to study two algorithms, while there might be other algorithms that may have higher accuracy. Several other characteristics and features of the vehicle will be added to the system so that more accurate prices will be provided by the system.

Several features that can further increase the accuracy of car price prediction include:

1) Horsepower
2) Battery power
3) Suspension
4) Cylinder
5) Torque
6) Boot space
7) Wheelbase
8) Sunroof
9) Music System

Since there are substantial improvements made in the field of science and technology on even daily basis, the technology used in vehicle industry also improves hand in hand, the next upgrade to this project should not cease to include the following:

7. Hybrid Cars
8. Electric Cars
9. Hydrogen cell-fueled Cars
10. Driverless Cars

Chapter 12

References

References:

1. https://www.olx.in/
2. https://www.cardekho.com/
3. https://www.cars24.com/
4. https://autoportal.com/
5. https://www.python.org/
6. https://pypi.org/
7. https://jupyter.org/

8. IBM Knowledge Centre. (n.d). Use of KNN. Retrieved from: https://www.ibm.com/support/knowledgecenter/SSHRBY/com.ibm.swg.im.dashdb.analytics.doc/doc/r_knn_usage.html

9. Gareth, J., Daniela, W., Trevor, H., & Tibshirani, R. (2013). *An Introduction to Statistical*

10. *Learning* (Vol. 8). https://doi.org/10.1016/j.peva.2007.06.006

11. Hurwitz, E., & Marwala, T. (2012). Common mistakes when applying computational intelligence and machine learning to stock market modelling. *arXiv preprint arXiv:1208.4429*.

12. Pal, N., Arora, P., Kohli, P., Sundararaman, D., & Palakurthy, S. S. (2018, April). How Much Is My Car Worth? A Methodology for Predicting Used Cars' Prices Using Random Forest. In *Future of Information and Communication Conference* (pp. 413–422). Springer, Cham.

13. Raschka, S., & Mirjalili, V. (2017). *Python machine learning*. Packt Publishing Ltd.

Chapter 13

Plagiarism Report

ORIGINALITY REPORT

5%	4%	1%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	1000projects.org Internet Source	2%
2	Submitted to Coventry University Student Paper	1%
3	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
4	Submitted to University of Greenwich Student Paper	1%
5	www.irjmets.com Internet Source	<1%
6	www.ijcseonline.org Internet Source	<1%
7	Submitted to The Robert Gordon University Student Paper	<1%
8	pt.scribd.com Internet Source	<1%
9	Min-Chang Kang, Doo-Yeol Yoo, Rishi Gupta. "Machine learning-based prediction for	<1%

compressive and flexural strengths of steel fiber-reinforced concrete", Construction and Building Materials, 2021
Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		