NAME OF THE PROJECT

CAR PRICE PREDICTION

Submitted by:

KHUSHBOO GUPTA

ACKNOWLEDGMENT

First and foremost, I would like to thank Flip Robo Technologies to provide me a chance to work on this project. It was a great experience to work on this project under your guidance.

I would like to present my gratitude to the following websites:

- Zendesk
- Kaggle
- Datatrained Notes
- Sklearn.org
- Crazyegg
- Cars24.com
- Youtube.com

These websites were of great help and due to this, I was able to complete my project effectively and efficiently.

INTRODUCTION

- Business Problem Framing

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

- Conceptual Background of the Domain Problem

One should know how to scrap the data from a website using the selenium as we have to create a fresh data for our project. Basic EDA concepts and regression algorithms must be known to work on this project. One should know what factors is important to predict the Car Price and how it is going to affect the used car selling business. Why predicting the car prices is important and how can it is going to help the company?

- Review of Literature

To be able to predict used cars market value can help both buyers and sellers.

- **Used car sellers (dealers):** They are one of the biggest target group that can be interested in results of this study. If used car sellers better understand what makes a car desirable, what the important features are for a used car, then they may consider this knowledge and offer a better service.

- **Online pricing services:** There are websites that offers an estimate value of a car. They may have a good prediction model. However, having a second model may help them to give a better prediction to their users. Therefore, the model developed in this study may help online web services that tells a used car’s market value.

- **Individuals:** There are lots of individuals who are interested in the used car market at some points in their life because they wanted to sell their car or buy a used car. In this process, it’s a big corner to pay too much or sell less then it’s market value.

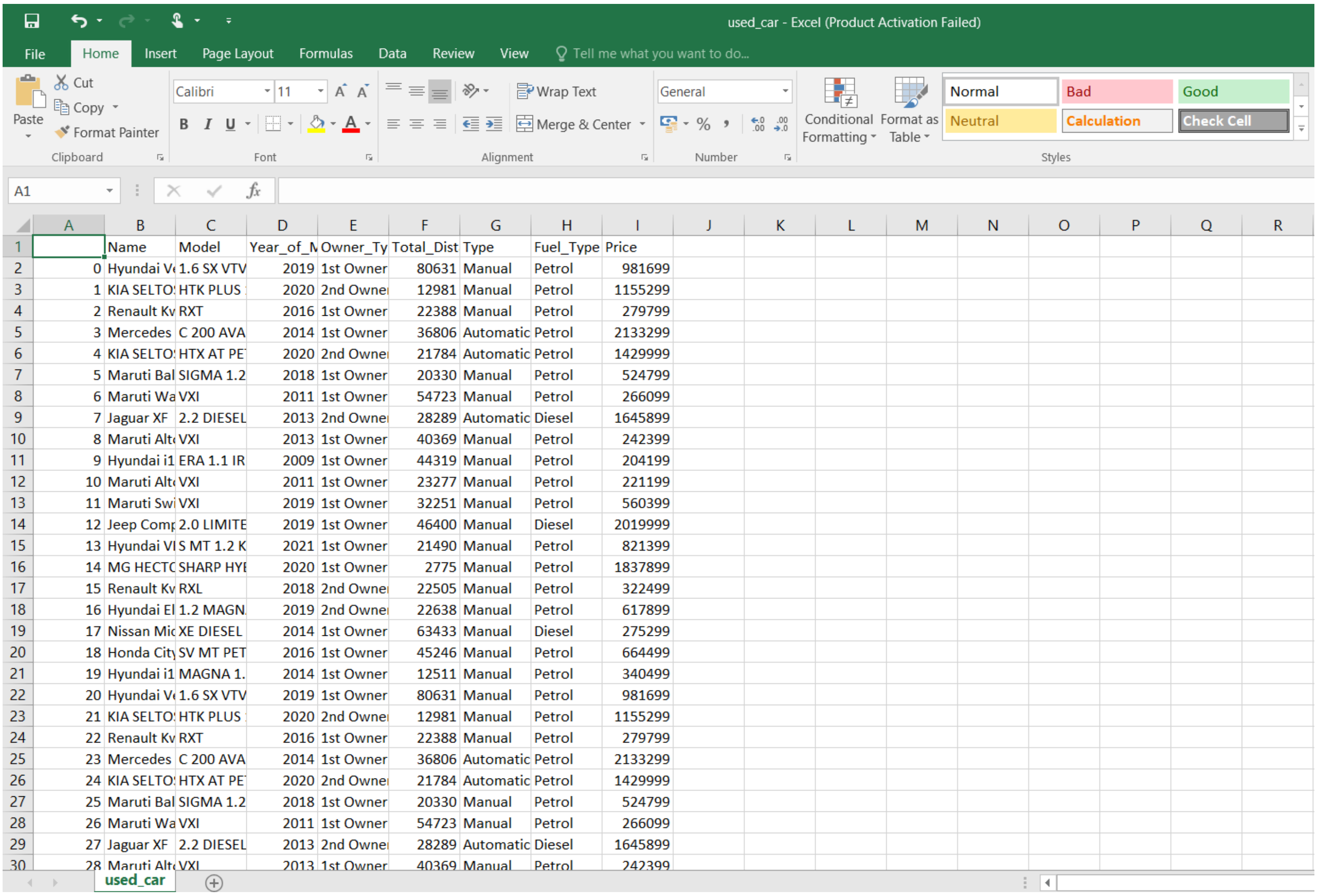
Analytical Problem Framing

- Data Sources and their formats

In this project our first task is to scrap the data from a car selling website. My source of data is Cars24 website from where I scrap the data of used cars for different locations including year of manufacturing, type of car, model, price etc.

To scrap the data, we will use the selenium. With the help of it scrap the data and save it in a csv file format for future use.

It contains 8 columns and 4983 rows. There are so many factors which can be used for the prediction of car price. It contains the factors on which the price of a used car can depend. Dataset contain both numerical as well as categorical data.



	Name	Model	Year_of_N	Owner_Ty	Total_Dist	Type	Fuel_Type	Price
0	Hyundai Ve	1.6 SX VTV	2019	1st Owner	80631	Manual	Petrol	981699
1	KIA SELTOS	HTK PLUS	2020	2nd Owner	12981	Manual	Petrol	1155299
2	Renault Kv	RXT	2016	1st Owner	22388	Manual	Petrol	279799
3	Mercedes	C 200 AVA	2014	1st Owner	36806	Automatic	Petrol	2133299
4	KIA SELTOS	HTX AT PE	2020	2nd Owner	21784	Automatic	Petrol	1429999
5	Maruti Bal	SIGMA 1.2	2018	1st Owner	20330	Manual	Petrol	524799
6	Maruti Wa	VXI	2011	1st Owner	54723	Manual	Petrol	266099
7	Jaguar XF	2.2 DIESEL	2013	2nd Owner	28289	Automatic	Diesel	1645899
8	Maruti Alto	VXI	2013	1st Owner	40369	Manual	Petrol	242399
9	Hyundai i1	ERA 1.1 IR	2009	1st Owner	44319	Manual	Petrol	204199
10	Maruti Alto	VXI	2011	1st Owner	23277	Manual	Petrol	221199
11	Maruti Swi	VXI	2019	1st Owner	32251	Manual	Petrol	560399
12	Jeep Comp	2.0 LIMITE	2019	1st Owner	46400	Manual	Diesel	2019999
13	Hyundai VI	S MT 1.2 K	2021	1st Owner	21490	Manual	Petrol	821399
14	MG HECTO	SHARP HYE	2020	1st Owner	2775	Manual	Petrol	1837899
15	Renault Kv	RXL	2018	2nd Owner	22505	Manual	Petrol	322499
16	Hyundai El	1.2 MAGN	2019	2nd Owner	22638	Manual	Petrol	617899
17	Nissan Mic	XE DIESEL	2014	1st Owner	63433	Manual	Diesel	275299
18	Honda City	SV MT PET	2016	1st Owner	45246	Manual	Petrol	664499
19	Hyundai i1	MAGNA 1.	2014	1st Owner	12511	Manual	Petrol	340499
20	Hyundai Ve	1.6 SX VTV	2019	1st Owner	80631	Manual	Petrol	981699
21	KIA SELTOS	HTK PLUS	2020	2nd Owner	12981	Manual	Petrol	1155299
22	Renault Kv	RXT	2016	1st Owner	22388	Manual	Petrol	279799
23	Mercedes	C 200 AVA	2014	1st Owner	36806	Automatic	Petrol	2133299
24	KIA SELTOS	HTX AT PE	2020	2nd Owner	21784	Automatic	Petrol	1429999
25	Maruti Bal	SIGMA 1.2	2018	1st Owner	20330	Manual	Petrol	524799
26	Maruti Wa	VXI	2011	1st Owner	54723	Manual	Petrol	266099
27	Jaguar XF	2.2 DIESEL	2013	2nd Owner	28289	Automatic	Diesel	1645899
28	Maruti Alto	VXI	2013	1st Owner	40369	Manual	Petrol	242399

- Libraries Used

  I am using different libraries to explore the datatset.

  1. Pandas – It is used to load and store the dataset. We can discuss the dataset with the pandas different attributes like .info, .columns, .shape
  2. Seaborn – It is used to plot the different types of plots like catplot, lineplot, countplot and more to have a better visualization of the dataset.
  3. Matplotlib.pyplot – It helps to give a proper description to the plotted graph by seaborn and make our graph more informative.
  4. Numpy – It is the library to perform the numerical analysis to the dataset

Load the Dataset

Importing the Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

In [2]:
```
pd.set_option('display.max_rows',None)   #displaying the maximum rows
pd.set_option('display.max_columns',None)   #displaying the maximum columns
df=pd.read_csv('used_car.csv')
```

In [11]:
```
df.head()  #first five rows of the dataset
```

Out[11]:

	Unnamed: 0	Name	Model	Year_of_Manufacturing	Owner_Type	Total_Distance_Travelled	Type	Fuel_Type	Price
0	0	Hyundai Verna	1.6 SX VTVT (O)	2019	1st Owner	80631	Manual	Petrol	981699
1	1	KIA SELTOS	HTK PLUS 1.5 PETROL	2020	2nd Owner	12981	Manual	Petrol	1155299
2	2	Renault Kwid	RXT	2016	1st Owner	22388	Manual	Petrol	279799
3	3	Mercedes Benz C Class	C 200 AVANTGARDE	2014	1st Owner	36806	Automatic	Petrol	2133299
4	4	KIA SELTOS	HTX AT PETROL	2020	2nd Owner	21784	Automatic	Petrol	1429999

In [12]:
```
df.tail()  #last five rows of the dataset
```

Out[12]:

	Unnamed: 0	Name	Model	Year_of_Manufacturing	Owner_Type	Total_Distance_Travelled	Type	Fuel_Type	Price
4978	4978	Maruti Alto	LXI	2019	1st Owner	18274	Manual	Petrol	335699
4979	4979	Maruti Alto	LXI	2019	1st Owner	16523	Manual	Petrol	344999
4980	4980	Maruti Alto 800	LXI	2019	1st Owner	21722	Manual	Petrol	339599
4981	4981	Maruti Alto	LXI	2019	1st Owner	17735	Manual	Petrol	344999
4982	4982	Maruti Dzire	VXI	2019	1st Owner	12651	Manual	Petrol	694799

We have successfully load our dataset for the further processes.

Checking the Attributes

- First & last five rows of both the dataset
- Shape of the datasets
- Columns present in the datasets
- Brief info about the datasets
- Null values present in both the dataset
- Unique values in each column
- Datatypes of each column

Importing the Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

In [2]:
```
pd.set_option('display.max_rows',None)    #displaying the maximum rows
pd.set_option('display.max_columns',None)    #displaying the maximum columns
df=pd.read_csv('used_car.csv')
```

In [11]:
```
df.head()  #first five rows of the dataset
```

Out[11]:

	Unnamed: 0	Name	Model	Year_of_Manufacturing	Owner_Type	Total_Distance_Travelled	Type	Fuel_Type	Price
0	0	Hyundai Verna	1.6 SX VTVT (O)	2019	1st Owner	80631	Manual	Petrol	981699
1	1	KIA SELTOS	HTK PLUS 1.5 PETROL	2020	2nd Owner	12981	Manual	Petrol	1155299
2	2	Renault Kwid	RXT	2016	1st Owner	22388	Manual	Petrol	279799
3	3	Mercedes Benz C Class	C 200 AVANTGARDE	2014	1st Owner	36806	Automatic	Petrol	2133299
4	4	KIA SELTOS	HTX AT PETROL	2020	2nd Owner	21784	Automatic	Petrol	1429999

In [12]:
```
df.tail()  #last five rows of the dataset
```

Out[12]:

	Unnamed: 0	Name	Model	Year_of_Manufacturing	Owner_Type	Total_Distance_Travelled	Type	Fuel_Type	Price
4978	4978	Maruti Alto	LXI	2019	1st Owner	18274	Manual	Petrol	335699
4979	4979	Maruti Alto	LXI	2019	1st Owner	16523	Manual	Petrol	344999
4980	4980	Maruti Alto 800	LXI	2019	1st Owner	21722	Manual	Petrol	339599
4981	4981	Maruti Alto	LXI	2019	1st Owner	17735	Manual	Petrol	344999
4982	4982	Maruti Dzire	VXI	2019	1st Owner	12651	Manual	Petrol	694799

In [13]:
```
df.shape  #total rows & columns in the dataset
```
Out[13]:
```
(4983, 9)
```

Dataset contains 4983 rows and 9 columns

In [14]:
```
df.columns  #columns present in the dataset
```
Out[14]:
```
Index(['Unnamed: 0', 'Name', 'Model', 'Year_of_Manufacturing', 'Owner_Type',
       'Total_Distance_Travelled', 'Type', 'Fuel_Type', 'Price'],
      dtype='object')
```

A brief info about the dataset

In [15]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4983 entries, 0 to 4982
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Unnamed: 0                4983 non-null   int64 
 1   Name                      4983 non-null   object
 2   Model                     4983 non-null   object
 3   Year_of_Manufacturing     4983 non-null   int64 
 4   Owner_Type                4983 non-null   object
 5   Total_Distance_Travelled  4983 non-null   int64 
 6   Type                      4941 non-null   object
 7   Fuel_Type                 4983 non-null   object
 8   Price                     4983 non-null   int64 
dtypes: int64(4), object(5)
memory usage: 350.5+ KB
```

In [16]:
```
df.nunique()  #total unique values in each column in the dataset
```
Out[16]:
```
Unnamed: 0                  4983
Name                         120
Model                        610
Year_of_Manufacturing         15
Owner_Type                     4
Total_Distance_Travelled    3481
Type                           2
Fuel_Type                      4
Price                       2743
dtype: int64
```

```
In [17]: df.dtypes #datatype of wach column
```

```
Out[17]: Unnamed: 0                   int64
         Name                        object
         Model                       object
         Year_of_Manufacturing        int64
         Owner_Type                  object
         Total_Distance_Travelled     int64
         Type                        object
         Fuel_Type                   object
         Price                        int64
         dtype: object
```

Checking the null values

```
In [18]: df.isnull().sum()
```

```
Out[18]: Unnamed: 0                   0
         Name                         0
         Model                        0
         Year_of_Manufacturing        0
         Owner_Type                   0
         Total_Distance_Travelled     0
         Type                        42
         Fuel_Type                    0
         Price                        0
         dtype: int64
```

Type column has 42 null values which has to be handled

Now we have checked the attributes for the dataset and get a rough idea about the dataset like the no of rows & columns, datatype & null values in the dataset.

Dealing with the Null Values

In the dataset null values are present, so we have to handled them for better model learning. As we have categorical data so we have to handled it accordingly. We will use the most occurring value in the column to fill the null value.

In [4]:
```
df['Type'].mode()  #checking the value which occur most of the time
```
Out[4]:
```
0    Manual
dtype: object
```

Filling the null values

In [5]:
```
df['Type']=df['Type'].fillna('Manual')
```

In [6]:
```
df.isnull().sum()
```
Out[6]:
```
Name                        0
Model                       0
Year_of_Manufacturing       0
Owner_Type                  0
Total_Distance_Travelled    0
Type                        0
Fuel_Type                   0
Price                       0
dtype: int64
```

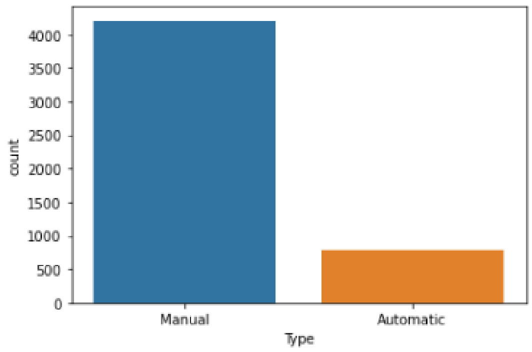
Hence, there is no null value now

EXPLORATORY DATA ANALYSIS

Univariate Analysis

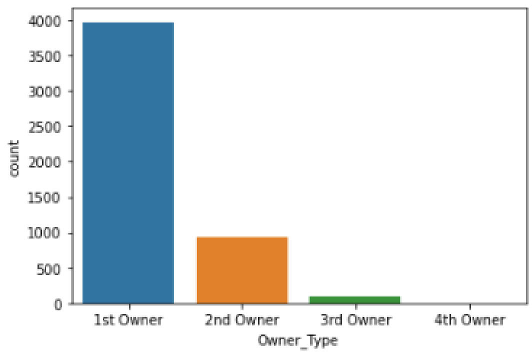
Let's understand each variable one by one and try to interpret about them.

```
In [29]: sns.countplot(df['Type'])
Out[29]: <AxesSubplot:xlabel='Type', ylabel='count'>
```



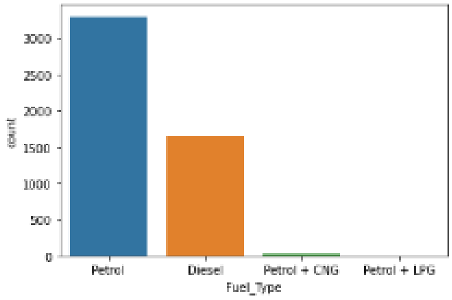
Most of the used cars are manual type.

```
In [30]: sns.countplot(df['Owner_Type'])
Out[30]: <AxesSubplot:xlabel='Owner_Type', ylabel='count'>
```



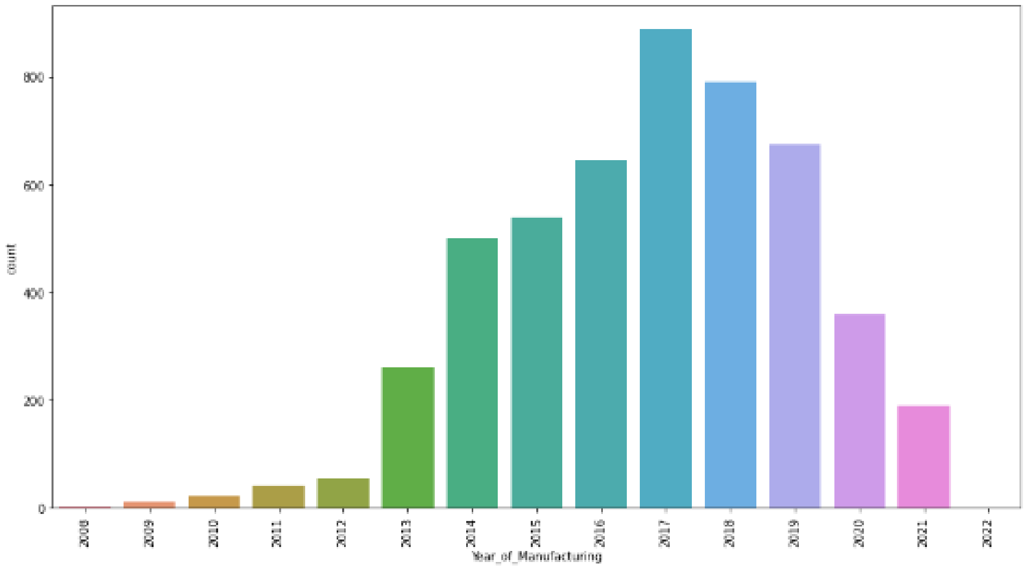
Most of the cars have their 1st owner who are seling the car.

In [31]:
```
sns.countplot(df['Fuel_Type'])
```
Out[31]: <AxesSubplot:xlabel='Fuel_Type', ylabel='count'>



Petrol based cars are on number one in the list followed by the diesel

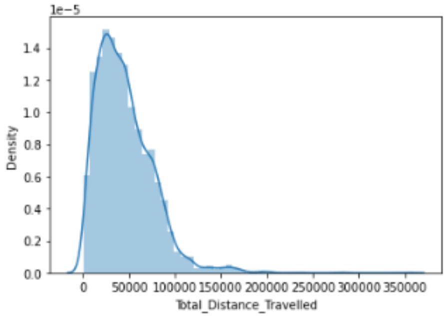
In [33]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=90)
sns.countplot(df['Year_of_Manufacturing'])
```
Out[33]: <AxesSubplot:xlabel='Year_of_Manufacturing', ylabel='count'>



2017 year cars have the highest number in the dataset followed by the 2018 & 2019

```
In [36]: sns.distplot(df['Total_Distance_Travelled'])
```

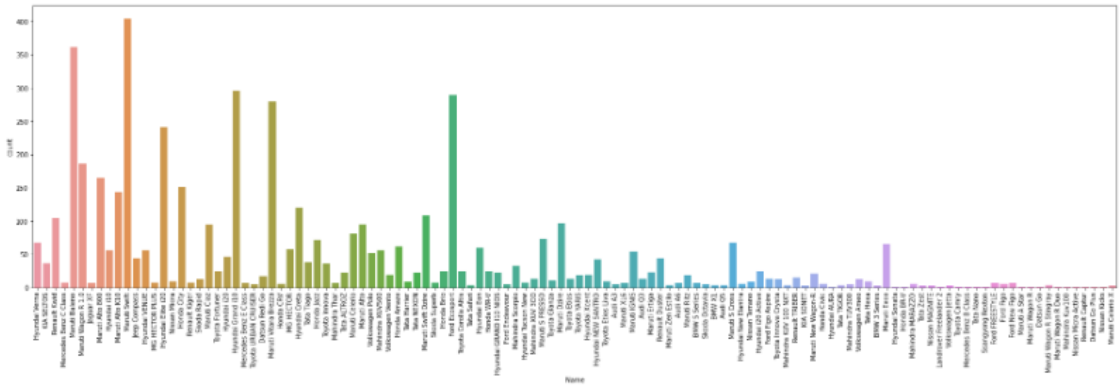
Out[36]: <AxesSubplot:xlabel='Total_Distance_Travelled', ylabel='Density'>



Least travelled cars have high density.

```
In [38]: plt.figure(figsize=(30,8))
         plt.xticks(rotation=90)
         sns.countplot(df['Name'])
```

Out[38]: <AxesSubplot:xlabel='Name', ylabel='count'>



Maruti swift cars number is the most followed by Maruti baleno

Observations

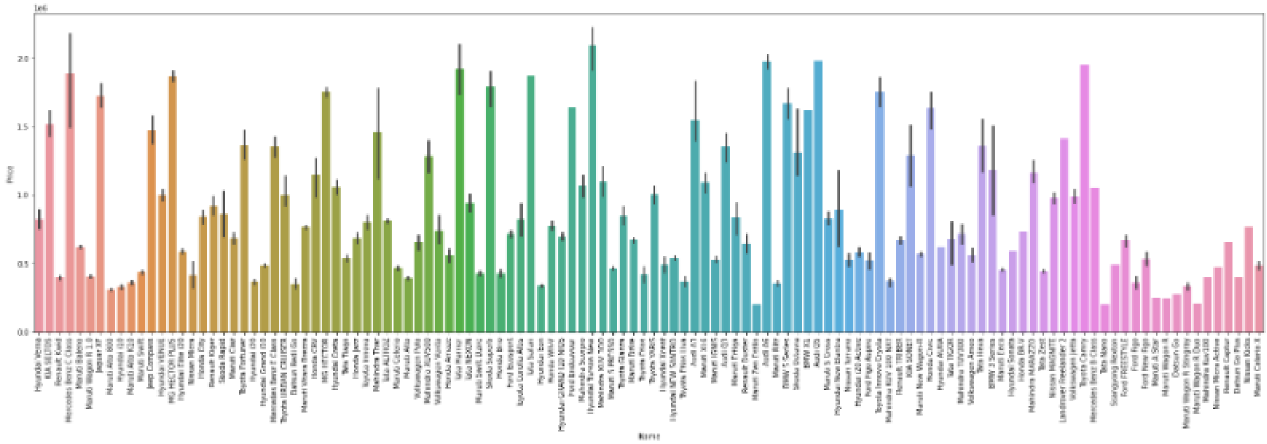
- Most of the used cars are manual type.
- Most of them are 1st owner who are selling their car.
- Petrol based cars are on number one in the list followed by the diesel
- 2017 year cars have the highest number in the dataset followed by the 2018 & 2019
- Density of least travelled cars is high.
- Maruti swift cars number is the most followed by Maruti Baleno.

Bivariate Analysis

Let's understand each variable relation with the target variable and interpret how target variable vary with the inputs.

In [40]:
```
plt.figure(figsize=(30,8))
plt.xticks(rotation=90)
sns.barplot(df['Name'],df['Price'])
```

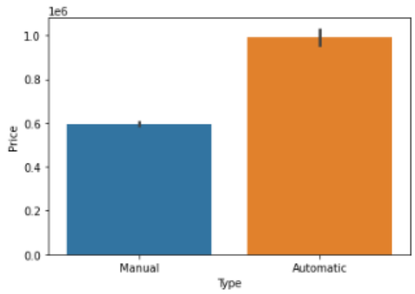
Out[40]: <AxesSubplot:xlabel='Name', ylabel='Price'>



Mercedes Benz C Class price is the highest followed by Hyundai Tuscon New

In [43]:
```
sns.barplot(df['Type'],df['Price'])
```

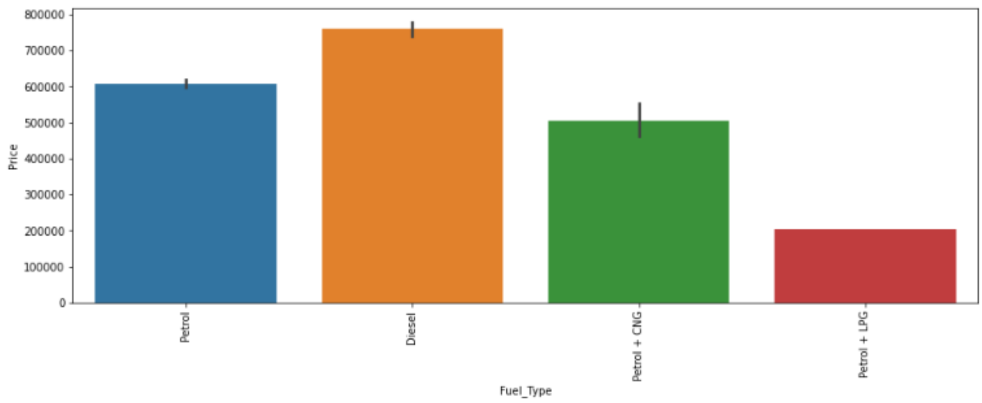
Out[43]: <AxesSubplot:xlabel='Type', ylabel='Price'>



Automatic car price is more than the manual one

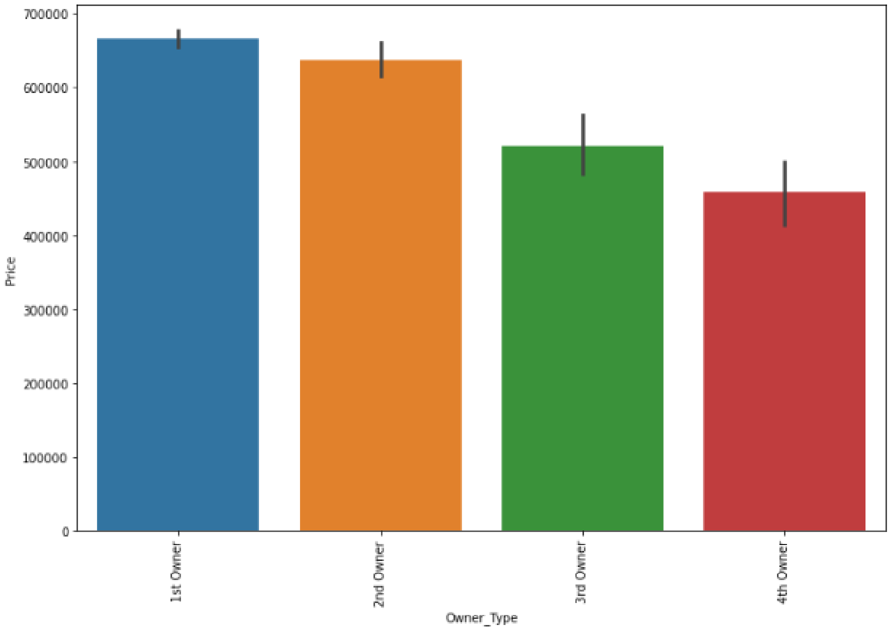
In [45]:
```
plt.figure(figsize=(15,5))
plt.xticks(rotation=90)
sns.barplot(df['Fuel_Type'],df['Price'])
```

Out[45]: <AxesSubplot:xlabel='Fuel_Type', ylabel='Price'>

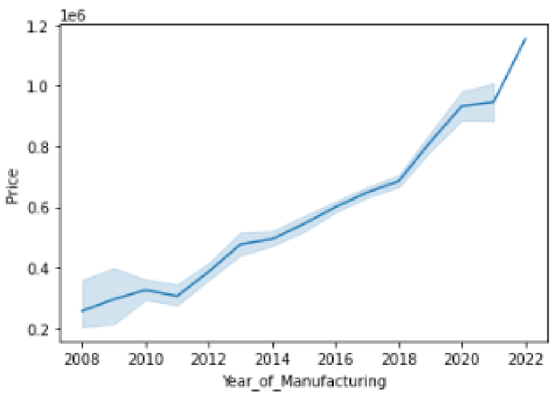


Diesel based car is costlier followed by Petrol

In [47]:
```
plt.figure(figsize=(12,8))
plt.xticks(rotation=90)
sns.barplot(df['Owner_Type'],df['Price'])
```
Out[47]: <AxesSubplot:xlabel='Owner_Type', ylabel='Price'>



In [28]:
```
sns.lineplot(df['Year_of_Manufacturing'],df['Price'])
```
Out[28]: <AxesSubplot:xlabel='Year_of_Manufacturing', ylabel='Price'>



As we go for the recent manufacturing model price will be high on the other side old cars have lower rates

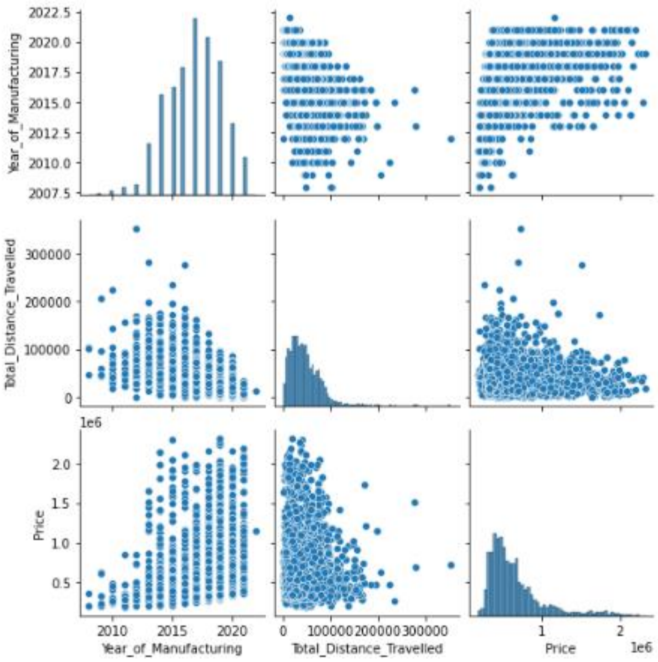
Observation

- Mercedes Benz C Class price is the highest followed by Hyundai Tuscon New
- Automatic car price is more than the manual one
- Diesel based car is costlier followed by Petrol
- 1st owner based car's price is more. 2nd owner based car price is also high.
- As we go for the recent manufacturing model price will be high on the other side old cars have lower rates

Multivariate Analysis

Using the pairplot function plot each column relation with each other to have a better understanding.

```
In [16]:  sns.pairplot(df)
Out[16]:  <seaborn.axisgrid.PairGrid at 0x1ee720d8d30>
```



Label Encoding & Correlation

As we have some categorical data we have to encoded those columns for machine learning model. We will use Label Encoder from sklearn.preprocessing.

We will describe the statistical summary of the dataset and find the correlation of each column.

Find the skewness of each column as well for each column, if it is out of acceptable range then we have to scale the skewness also.

Label Encoding

In [7]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in df.columns:
    if df[i].dtypes=='object':
        df[i]=le.fit_transform(df[i].astype(str))
df.head()
```

Out[7]:

	Name	Model	Year_of_Manufacturing	Owner_Type	Total_Distance_Travelled	Type	Fuel_Type	Price
0	35	129	2019	0	80631	1	1	981699
1	42	293	2020	1	12981	1	1	1155299
2	91	348	2016	0	22388	1	1	279799
3	81	221	2014	0	36806	0	1	2133299
4	42	294	2020	1	21784	0	1	1429999

Statistical Summary

In [55]:
```
df.describe()
```

Out[55]:

	Name	Model	Year_of_Manufacturing	Owner_Type	Total_Distance_Travelled	Type	Fuel_Type	Price
count	4983.000000	4983.000000	4983.000000	4983.000000	4983.000000	4983.000000	4983.000000	4.983000e+03
mean	53.699378	343.774232	2016.782260	0.224162	45354.276139	0.843668	0.675296	6.581687e+05
std	26.973384	164.271456	2.281862	0.463569	30543.047924	0.363206	0.482248	3.469485e+05
min	0.000000	0.000000	2008.000000	0.000000	71.000000	0.000000	0.000000	1.909990e+05
25%	29.000000	207.000000	2015.000000	0.000000	22775.000000	1.000000	0.000000	4.263490e+05
50%	59.000000	380.000000	2017.000000	0.000000	40336.000000	1.000000	1.000000	5.590990e+05
75%	71.000000	484.000000	2018.000000	0.000000	62285.500000	1.000000	1.000000	7.637990e+05
max	119.000000	609.000000	2022.000000	3.000000	353688.000000	1.000000	3.000000	2.314599e+06

- We have outliers in the dataset
- Little skewness present in the dataset

Correlation

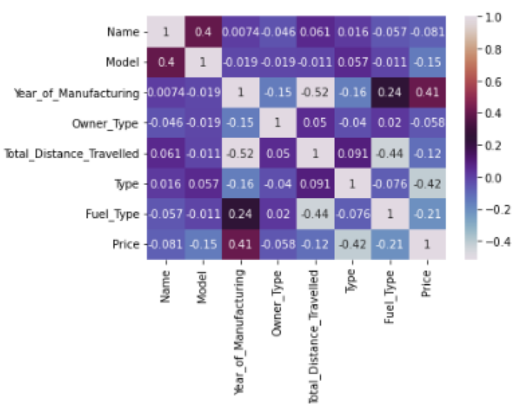
In [56]:
```
corr=df.corr()
corr
```

Out[56]:

	Name	Model	Year_of_Manufacturing	Owner_Type	Total_Distance_Travelled	Type	Fuel_Type	Price
Name	1.000000	0.396621	0.007379	-0.046251	0.060823	0.016039	-0.056684	-0.080916
Model	0.396621	1.000000	-0.019361	-0.019073	-0.011084	0.056502	-0.011246	-0.150977
Year_of_Manufacturing	0.007379	-0.019361	1.000000	-0.153660	-0.515338	-0.160721	0.243455	0.405246
Owner_Type	-0.046251	-0.019073	-0.153660	1.000000	0.050248	-0.039791	0.020377	-0.057535
Total_Distance_Travelled	0.060823	-0.011084	-0.515338	0.050248	1.000000	0.090934	-0.442991	-0.119865
Type	0.016039	0.056502	-0.160721	-0.039791	0.090934	1.000000	-0.075570	-0.415276
Fuel_Type	-0.056684	-0.011246	0.243455	0.020377	-0.442991	-0.075570	1.000000	-0.209325
Price	-0.080916	-0.150977	0.405246	-0.057535	-0.119865	-0.415276	-0.209325	1.000000

In [57]:
```
sns.heatmap(corr,annot=True,cmap='twilight')
```

Out[57]: <AxesSubplot:>



Checking Skewness

In [58]:
```
df.skew()
```

Out[58]:
```
Name                       0.181928
Model                     -0.379018
Year_of_Manufacturing     -0.361817
Owner_Type                 1.981246
Total_Distance_Travelled   1.534313
Type                      -1.893177
Fuel_Type                 -0.560006
Price                      1.848527
dtype: float64
```

The skewness in the dataset is very little so we can go ahead with further process

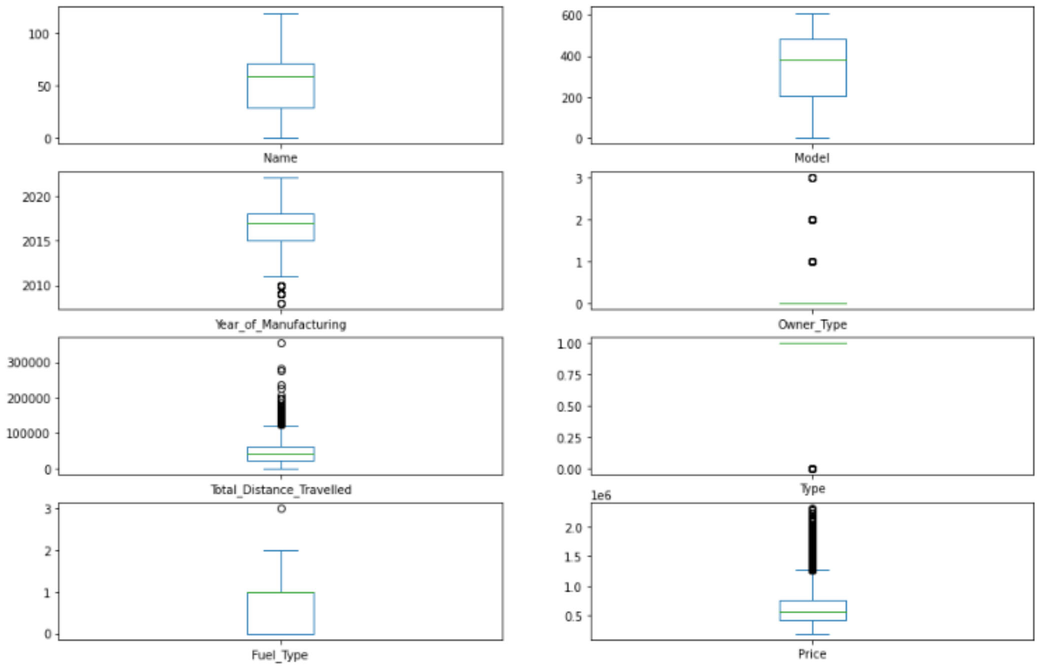
We have encoded our categorical data and find the correlation between each column. As skewness is acceptable so there is no need to scale the skewness.

Removing the Outliers

Plotting the outliers

In [61]:
```
df.plot(kind='box',subplots=True,layout=(4,2),figsize=(15,10))
```

Out[61]:
```
Name                          AxesSubplot(0.125,0.71587;0.352273x0.16413)
Model                      AxesSubplot(0.547727,0.71587;0.352273x0.16413)
Year_of_Manufacturing        AxesSubplot(0.125,0.518913;0.352273x0.16413)
Owner_Type                AxesSubplot(0.547727,0.518913;0.352273x0.16413)
Total_Distance_Travelled     AxesSubplot(0.125,0.321957;0.352273x0.16413)
Type                      AxesSubplot(0.547727,0.321957;0.352273x0.16413)
Fuel_Type                       AxesSubplot(0.125,0.125;0.352273x0.16413)
Price                        AxesSubplot(0.547727,0.125;0.352273x0.16413)
dtype: object
```



Removing outliers

In [9]:
```
from scipy.stats import zscore
z=np.abs(zscore(df))
threshold=3
print(np.where(z>3))
df_new=df[(z<3).all(axis=1)]
df=df_new
df.shape
```

```
(array([   3,    9,   12,   14,   23,   29,   32,   34,   65,   67,   74,
          76,   80,   98,  108,  122,  124,  137,  142,  150,  167,  173,
         188,  193,  294,  304,  332,  338,  357,  368,  368,  372,  375,
         376,  397,  413,  417,  424,  437,  452,  514,  538,  552,  555,
         635,  675,  684,  701,  704,  712,  713,  723,  729,  735,  767,
         775,  777,  796,  798,  806,  809,  815,  839,  850,  855,  859,
         901,  964,  974,  989, 1035, 1079, 1081, 1087, 1094, 1098, 1121,
        1122, 1128, 1128, 1136, 1137, 1154, 1199, 1211, 1230, 1297, 1326,
        1363, 1365, 1384, 1388, 1389, 1390, 1398, 1407, 1410, 1417, 1435,
        1442, 1465, 1485, 1493, 1499, 1501, 1515, 1517, 1530, 1553, 1581,
        1638, 1654, 1678, 1679, 1694, 1696, 1699, 1699, 1705, 1708, 1710,
        1736, 1743, 1767, 1778, 1844, 1859, 1863, 1866, 1878, 1970, 2044,
        2046, 2082, 2100, 2107, 2146, 2155, 2182, 2184, 2236, 2247, 2249,
        2255, 2257, 2262, 2285, 2288, 2298, 2311, 2328, 2330, 2338, 2355,
        2364, 2368, 2384, 2386, 2405, 2457, 2463, 2498, 2505, 2519, 2526,
        2527, 2552, 2592, 2609, 2620, 2654, 2662, 2690, 2692, 2693, 2698,
        2700, 2714, 2727, 2749, 2757, 2774, 2810, 2810, 2812, 2836, 2854,
        2876, 2879, 2885, 2887, 2908, 2933, 2938, 2940, 2973, 2993, 2998,
        2999, 3000, 3005, 3010, 3034, 3044, 3052, 3098, 3128, 3169, 3175,
        3177, 3194, 3197, 3199, 3224, 3265, 3267, 3289, 3301, 3305, 3332,
        3337, 3346, 3347, 3352, 3361, 3361, 3371, 3380, 3383, 3388, 3404,
        3405, 3410, 3412, 3420, 3446, 3449, 3453, 3454, 3456, 3457, 3459,
        3459, 3471, 3477, 3481, 3491, 3509, 3513, 3518, 3536, 3539, 3567,
        3600, 3618, 3626, 3636, 3637, 3644, 3648, 3649, 3675, 3698, 3698,
        3700, 3737, 3759, 3785, 3797, 3800, 3806, 3856, 3859, 3870, 3890,
        3909, 3918, 3929, 3941, 3942, 3988, 3991, 4004, 4035, 4056, 4060,
        4061, 4077, 4079, 4096, 4099, 4109, 4125, 4131, 4134, 4136, 4140,
        4185, 4227, 4239, 4308, 4316, 4398, 4433, 4443, 4511, 4548, 4594,
        4640, 4669, 4692, 4698, 4714, 4721, 4725, 4752, 4771, 4772, 4794,
        4906], dtype=int64), array([7, 2, 7, 7, 7, 2, 7, 7, 3, 7, 7, 7, 3, 3, 7, 7, 3, 7, 7, 7, 4, 3,
        7, 7, 3, 4, 3, 4, 4, 3, 4, 4, 2, 7, 7, 4, 4, 4, 7, 7, 4, 3, 3, 4,
        4, 3, 7, 7, 7, 7, 2, 7, 7, 3, 7, 3, 3, 7, 3, 3, 7, 4, 7, 7, 7, 7,
        7, 4, 4, 3, 3, 7, 4, 4, 4, 7, 4, 4, 3, 4, 7, 2, 3, 4, 7, 3, 4, 4,
        3, 7, 7, 7, 7, 7, 7, 2, 3, 7, 3, 7, 3, 7, 3, 7, 7, 3, 4, 7, 7, 7,
        4, 4, 4, 4, 3, 3, 3, 4, 7, 4, 4, 2, 4, 7, 7, 7, 3, 4, 4, 3, 4, 3,
        7, 7, 7, 3, 7, 4, 7, 7, 4, 7, 7, 7, 7, 3, 4, 7, 7, 7, 4, 7, 4, 3,
        4, 3, 7, 7, 7, 4, 7, 7, 7, 7, 7, 3, 3, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 4, 3, 7, 7, 7, 4, 7, 3, 7, 4, 7, 7, 7, 7, 4, 7, 7, 7, 7, 3, 3,
        7, 3, 4, 3, 3, 7, 7, 4, 4, 7, 7, 7, 7, 7, 7, 3, 7, 7, 3, 3, 2, 3,
        3, 3, 7, 3, 2, 4, 3, 3, 3, 7, 7, 4, 3, 4, 3, 2, 2, 3, 3, 7, 7, 2,
        3, 3, 7, 4, 3, 3, 3, 3, 4, 7, 3, 3, 3, 4, 3, 4, 3, 7, 3, 2, 2, 6,
        7, 7, 7, 7, 3, 7, 7, 4, 4, 7, 7, 7, 3, 4, 7, 3, 3, 3, 7, 4, 3, 4,
        7, 7, 7, 4, 4, 3, 3, 3, 3, 4, 3, 3, 3, 3, 7, 7, 7, 4, 4, 4, 3, 3,
        4, 7, 7, 7, 4, 3, 7, 3, 3, 4, 3, 4], dtype=int64))
```

Out[9]: (4670, 8)

We have some outliers present in the dataset, so let's handle them also. As the outliers in the dataset will affect our ML model. We need to remove all the outliers present in the dataset.

There is something called zscore which indicates how many standard deviations away an element is from the mean. We consider the points as outliers whose zscore is above 3 or less than -3. So we need to remove all such points from our dataset.

Using the threshold, we have removed all the points where the zscore is greater than 3. Now the total number of rows after removing the outliers are 4670.

MODEL BUILDING

We will import important libraries for the building the ML model and defining the different models for our easiness.

Finding the best random state for the train test split.

Model Building

```
In [13]: #importing the different machine learning models

         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error,mean_absolute_error
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.svm import SVR
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.neighbors import KNeighborsRegressor
         from sklearn.metrics import r2_score
```

```
In [14]: # defining the different models

         lg=LinearRegression()
         rdr=RandomForestRegressor()
         svr=SVR()
         dtr=DecisionTreeRegressor()
         knr=KNeighborsRegressor()
```

Finding the best random state

```
In [15]: model=[lg,rdr,svr,dtr,knr]
         maxAcc=0
         maxRS=0
         for i in range(40,60):
             x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=.20)
             lg.fit(x_train,y_train)
             pred=lg.predict(x_test)
             acc=r2_score(y_test,pred)
             if acc>maxAcc:
                 maxAcc=acc
                 maxRS=i
         print('Best Accuracy score is', maxAcc , 'on random state', maxRS)

         Best Accuracy score is 0.4459634857166468 on random state 45
```

```
In [16]: x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=45,test_size=.20)
```

Regression Algorithms

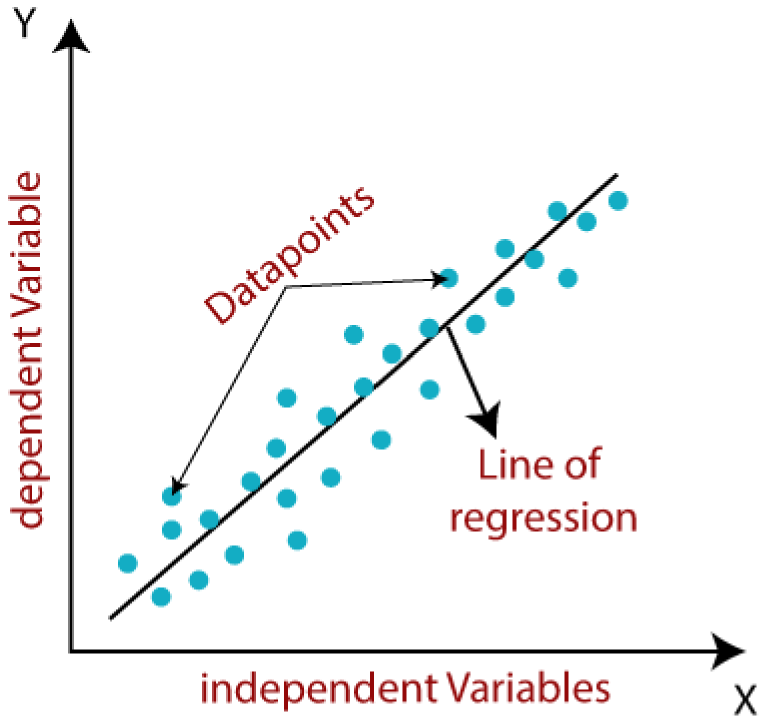
We have use five different regression algorithms to find the best model for our problem.

1. Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

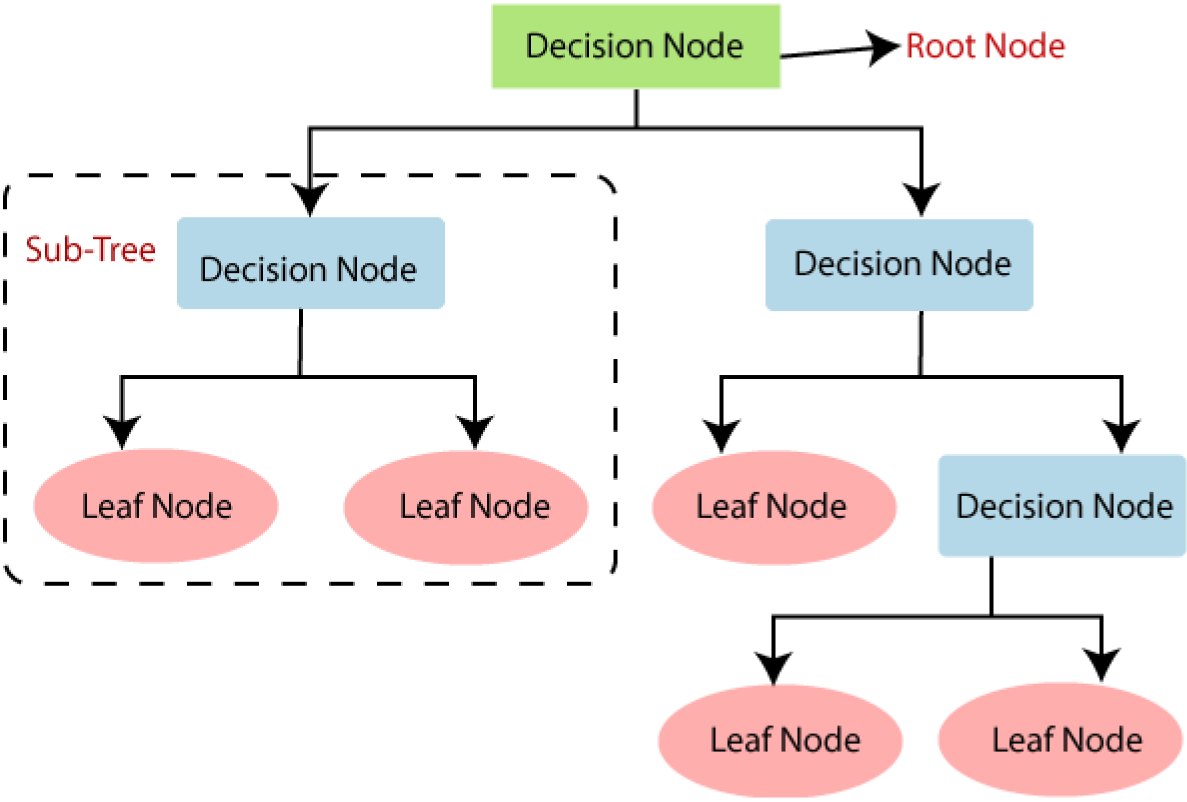
Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.



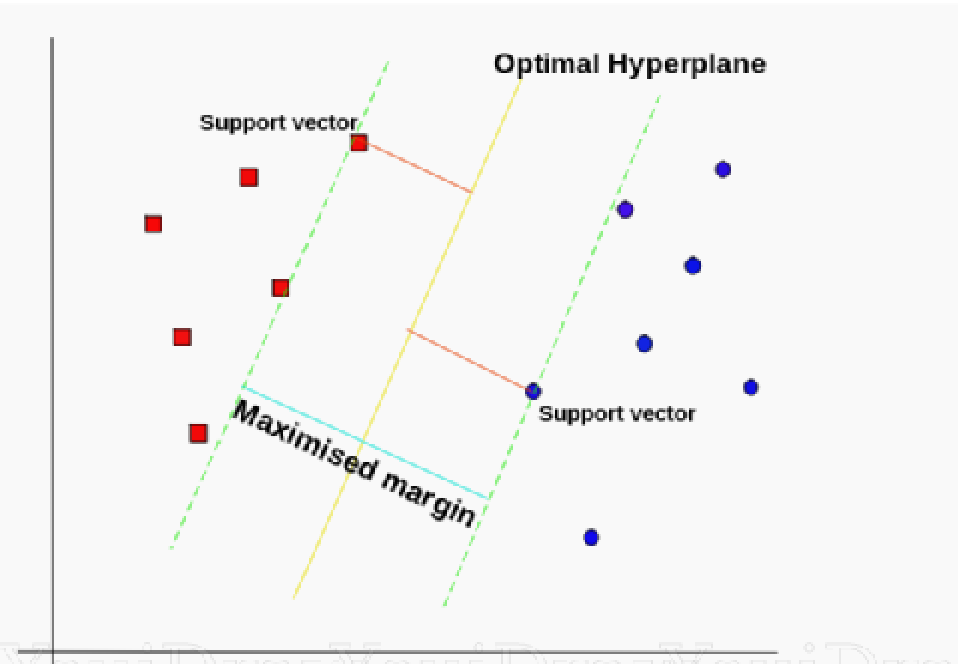
2. Decision Tree Regressor

Decision Tree can be used both in classification and regression problem. Decision trees are predictive models that use a set of binary rules to calculate a target value. Each individual tree is a fairly simple model that has branches, nodes and leaves. Decision tree regression 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**. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.



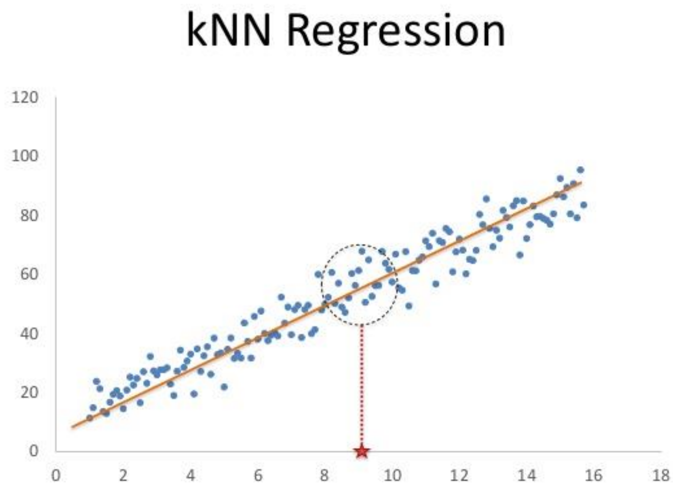
3. Support Vector Regressor

Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.



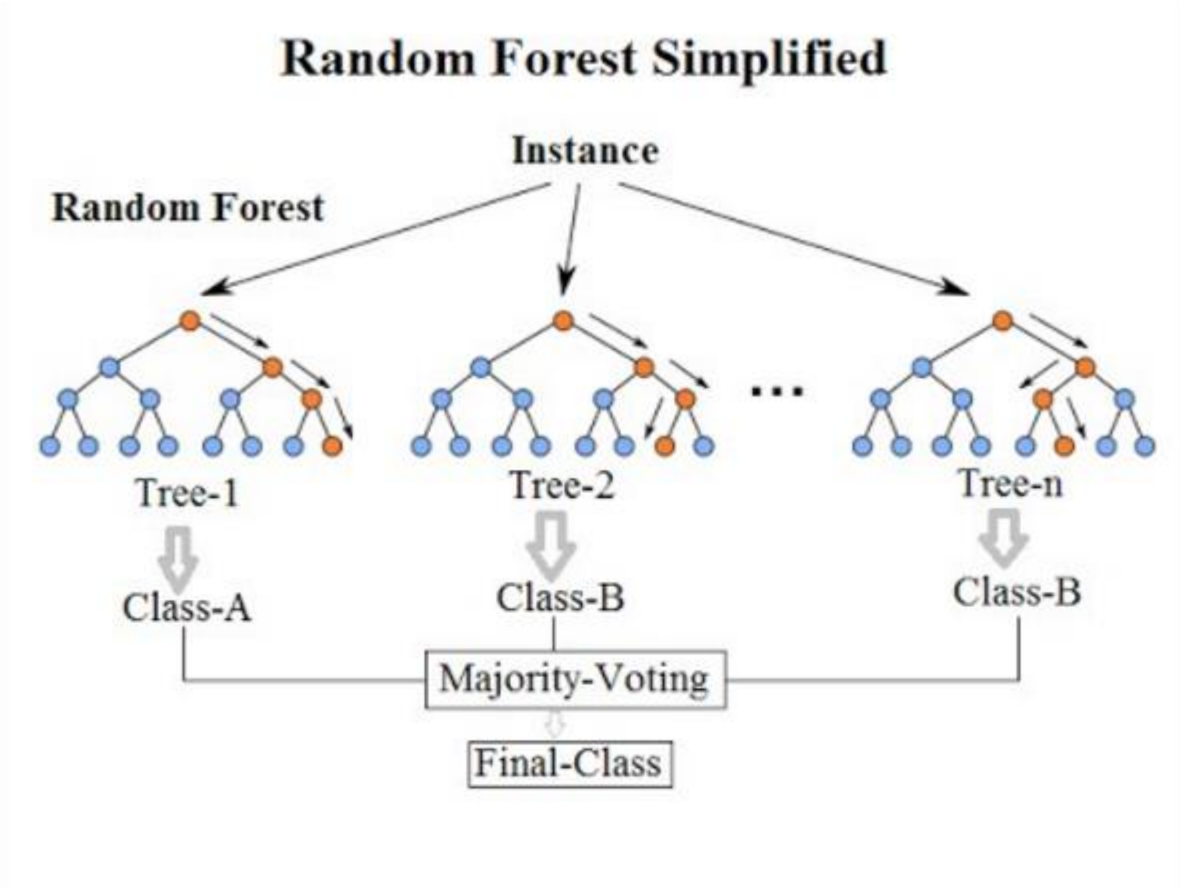
4. K Neighbor Regression

KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same *neighbourhood*. The size of the neighbourhood needs to be set by the analyst or can be chosen using cross-validation (we will see this later) to select the size that minimises the mean-squared error.



5. Random Forest Regression

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.



Libraries Used for Regression Models

- Linear Regression
  - ➢ from sklearn.linear_model import LinearRegression
- Decision Tree Regressor
  - ➢ from sklearn.tree import DecisionTreeRegressor
- Support Vector Regressor
  - ➢ from sklearn.svm import SVR
- Kneighbor Regressor
  - ➢ from sklearn.neighbors import KNeighborsRegressor
- Random Forest Regressor
  - ➢ from sklearn.ensemble import RandomForestRegressor

Model Accuracy:

MODEL	ACCURACY
Linear Regression	0.4459634857166468
Decision Tree Regressor	0.8856552523096618
Support Vector Regressor	-0.08005490482792754
Kneighbor Regressor	0.04292499937121386
Random Forest Regressor	0.8815066126350098

Linear Regression

In [17]:
```
lg.fit(x_train,y_train)
pred1=lg.predict(x_test)
acc=r2_score(y_test,pred1)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.4459634857166468
```

Decision Tree Regressor

In [18]:
```
dtr.fit(x_train,y_train)
pred2=dtr.predict(x_test)
acc=r2_score(y_test,pred2)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.8856552523096618
```

Support Vector Regressor

In [19]:
```
svr.fit(x_train,y_train)
pred3=svr.predict(x_test)
acc=r2_score(y_test,pred3)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  -0.08005490482792754
```

K Neighbor Regressor

In [20]:
```
knr.fit(x_train,y_train)
pred4=knr.predict(x_test)
acc=r2_score(y_test,pred4)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.04292499937121386
```

Random Forest Regressor

In [21]:
```
rdr.fit(x_train,y_train)
pred5=rdr.predict(x_test)
acc=r2_score(y_test,pred5)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.942351061631132
```

Hence, we are getting the best accuracy score through the Random Forest Classifier Model. We will go ahead with this to find the cross val score and hypermeter tuning.

Cross Val Score & Hypermeter Tuning

Cross-validation provides information about how well a classifier generalizes, specifically the range of expected errors of the classifier. Cross Val Score tells how the model is generalized at a particular cross validation.

At CV=6 we get the best results i.e. the Random Forest Classifier more generalized at cv=6, so we calculate the hyper parameters at this value.

We will find which parameters of random forest classifier are the best foe our model. We will do this using Grid Search CV method & also calculate the accuracy score at those best parameters.

Cross Val Score

In [22]:
```
from sklearn.model_selection import cross_val_score
for i in range(3,7):
    cr=cross_val_score(rdr,x,y,cv=i)
    cr_mean=cr.mean()
    print("at cv= ", i)
    print('cross val score = ',cr_mean*100)
```

```
at cv=  3
cross val score =  85.942859785015
at cv=  4
cross val score =  90.8769072158919
at cv=  5
cross val score =  91.41279286544679
at cv=  6
cross val score =  91.7821441798227
```

Hypermeter Tuning

In [23]:
```
from sklearn.model_selection import GridSearchCV
# creating parameters
para={'criterion':['squared_error','absolute_error','poisson'],
      'max_features':['sqrt','log2'],
      'max_depth':[1,2,3,4,5]}

GCV=GridSearchCV(rdr,para,cv=6,scoring='accuracy')
GCV.fit(x_train,y_train)
GCV.best_params_
```

Out[23]:
```
{'criterion': 'squared_error', 'max_depth': 1, 'max_features': 'sqrt'}
```

In [24]:
```
GCV_pred=GCV.best_estimator_.predict(x_test)
r2_score(y_test,GCV_pred)
```

Out[24]:
```
0.184987654844048
```

For Random Forest we are not getting very good results. The r2_score is very low for the best parameters. So we will move to the next best model and find the best parameters for Decision Tree Regressor and calculate the accuracy.

Cross val Score

```
In [26]:  from sklearn.model_selection import cross_val_score
          for i in range(3,7):
              cr=cross_val_score(dtr,x,y,cv=i)
              cr_mean=cr.mean()
              print("at cv= ", i)
              print('cross val score = ',cr_mean*100)

          at cv=  3
          cross val score =  76.33747044735803
          at cv=  4
          cross val score =  85.46379663489655
          at cv=  5
          cross val score =  87.27113441425047
          at cv=  6
          cross val score =  87.28937269229745
```

Hypermeter Tuning

```
In [28]:  from sklearn.model_selection import GridSearchCV
          # creating parameters
          para={'criterion':['squared_error','absolute_error','poisson','friedman_mse'],
                'max_features':['sqrt','log2','auto'],
                'max_depth':[1,2,3,4,5],
                'splitter':['best','random']}

          GCV=GridSearchCV(dtr,para,cv=6,scoring='accuracy')
          GCV.fit(x_train,y_train)
          GCV.best_params_

Out[28]:  {'criterion': 'absolute_error',
           'max_depth': 5,
           'max_features': 'auto',
           'splitter': 'random'}

In [29]:  GCV_pred=GCV.best_estimator_.predict(x_test)
          r2_score(y_test,GCV_pred)

Out[29]:  0.4663703959145542
```

With decision tree regressor we are getting a bettor accuracy score so we will save the decision tree model

We are getting the better accuracy for the decision tree Regressor, so we will save this model for our future predictions.

Saving the Model

Saving the best model – Decision Tree Regression in this case for future predictions. Let’s see what are the actual test data and what our model predicts.

Saving the Model

In [32]:
```
import pickle
filename='used_car_price.pkl'
pickle.dump(dtr, open(filename,'wb'))
```

Conclusion

In [31]:
```
a=np.array(y_test)
pred=np.array(pred2)
Used_Car_Price=pd.DataFrame({'Actual':a,'Predicted':pred})
Used_Car_Price
```

3	607499	476349.0
4	557399	565499.0
5	770199	720999.0
6	421399	428999.0
7	1097099	1097099.0
8	340499	340499.0
9	490199	490199.0
10	436699	415099.0
11	863599	863599.0
12	723699	846699.0
13	407999	292599.0
14	522999	460599.0
15	818599	1055199.0

Hence up to some good extensions our model predicted so well.

CONCLUSION

Conclusion of the Study

The results of this study suggest following outputs which might be useful for the client to evaluate the price on the basis of new data:

- There are lot of things that is going to decide the price of a car. As we see above in our visualizations, a lot of things affect the price like year of manufacturing, brand, distance travelled & many more. One needs to analyse every aspect to have good hands on the prediction of the price.
- With the machine learning it become easier to predict the price but yes it is not 100% accurate, it provides an idea and accordingly we can analyse the market and prepare the strategies to grab the opportunities.

- ## Learning Outcomes of the Study in respect of Data Science
    - As our first step to collect the fresh or new data for the project in accordance with today's trend, scraping of data is not an easy task but yes every difficulty teaches something & I learn so much exceptions handling and error handling during the first phase of project.
    - Second phase is to create the machine learning model for the client to have a good hands on the prediction after the Covid 19 situation, which is an interesting part. Data pre-processing, null values handling, data cleaning and data engineering every time handle in a different way which is a good learning part.
    - Every model whose accuracy is higher doesn't mean it predicts always best. As we see we are getting the Random Forest accuracy highest but its hyperparameters accuracy is very low so we have to double check on with another algorithm as well.