FLIP ROBO

CAR PRICE PREDICTION

Submitted by:

Jyuthika Mankar

ACKNOWLEDGMENT

I am thankful to Flip Robo Technologies, situated in Bengaluru for giving me this opportunity to work as an Intern in their company. The experience which I am accumulating with the kind of projects Flip Robo Technologies is providing are nurturing. “Housing: Price Prediction” is one of the projects given to me for Machine Learning and data analysis. The dataset is provided by the company and working on that data has been proved to be a learning experience in every way. I would like to thank my SME Ms Sapna Verma for being considerate always to the interns and resolving the issues whenever they arose regarding the current project.

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. With the change in market due to covid 19 impact, the used cars business personals are facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. Hence, we have to make car price valuation model.

- Conceptual Background of the Domain Problem

  The growing world of e-commerce is not just restricted to buying electronics and clothing but everything that you expect in a general store. Keeping the general store perspective aside and looking at the bigger picture, every day there are thousands or perhaps millions of deals happening in the digital marketplace. One of the most booming markets in the digital space is that of the automobile industry wherein the buying and selling of used cars take place. Sometimes we need to walk up to the dealer or individual sellers to get a used car price quote. However, buyers and sellers face a major stumbling block when it comes to their used car valuation or say their second-hand car valuation. Traditionally, you would go to a showroom and get your vehicle inspected before learning about the price. So instead of doing all these stuffs we can build a machine learning model using different features of the used cars to predict the exact and valuable car price.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

In our scrapped dataset, our target variable "Used Car Price " is a continuous variable. Therefore, we will be handling this modelling problem as regression.

This project is done in two parts:

- Data Collection phase
- Model Building phase

Data Collection phase:

Almost 5000 used cars data was scrapped from various car selling websites. More the data better the model. Data can be scrapped of the used cars from websites (OLX, OLA, Car Dekho, Cars24 etc.) and fetched for different locations. The number of columns for data doesn't have limit, it's up to the creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometres, fuel, number of owners, location and at last target variable Price of the car. This data is to give a hint about important variables in used car model. Changes can be made by adding or removing the column completely depends on the website from which the data is fetched. All types of cars in the data for example- SUV, Sedans, Coupe, minivan, Hatchback were tried to be included.

Model Building phase:

After collecting the data, we need to build a machine learning model. Before model building, we have to do all data pre-processing steps and trying different models with different hyper parameters and selecting the best model. We have to follow the complete life cycle of data science. Include all the below steps mentioned:

1. Data Cleaning

2. Exploratory Data Analysis (EDA)

3. Data Pre-processing and Visualisation

4. Model Building

5. Model Evaluation

6. Selecting the best model

- ## Data Sources and their formats

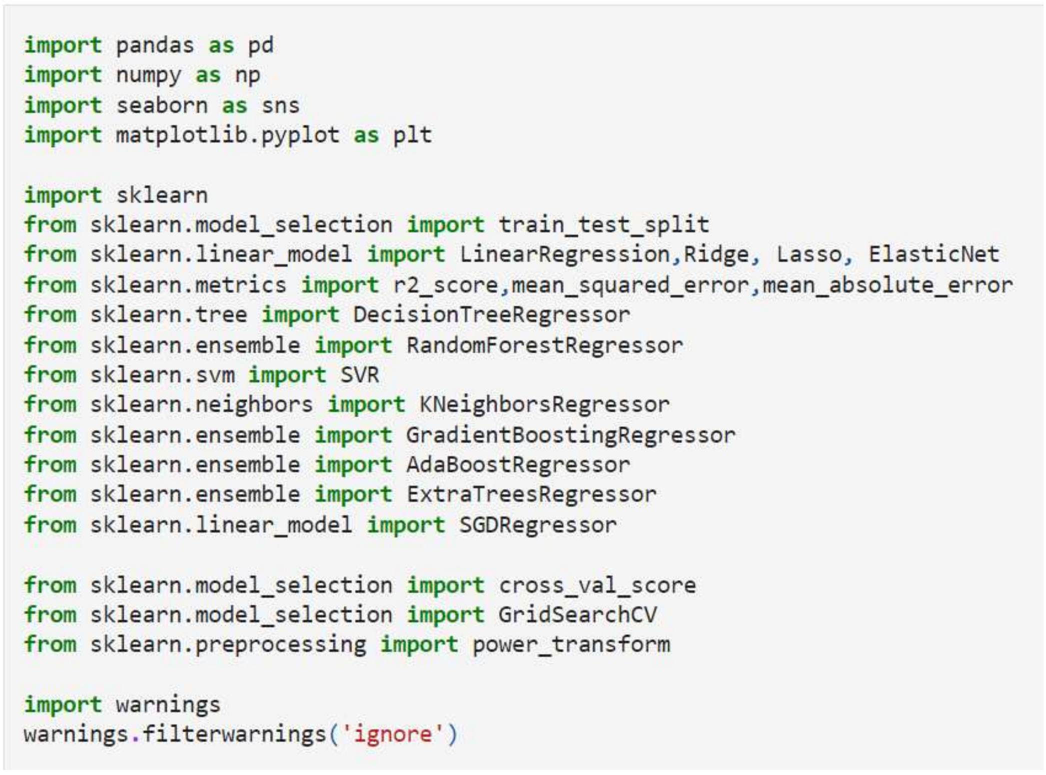
The data was scrapped from https://www.cars24.com/ from different locations in India and saved in the form of CSV (Comma Separated Value) as well as .xlsx format consisting 6 columns (5 features and 1 label) with almost 5000 number of records as explained below:

– Used Car Model - This shows the car model names
– Year of Manufacture - Gives us the year in which the car was made
– Kilometres Driven - Number of kilometres the car the driven reflecting on the Odometer
– Fuel Type - Shows the fuel type used by the vehicle
– Transmission Type - Gives us the manual or automatic gear shifting mechanism
– Used Car Price - Lists the selling price of the used cars

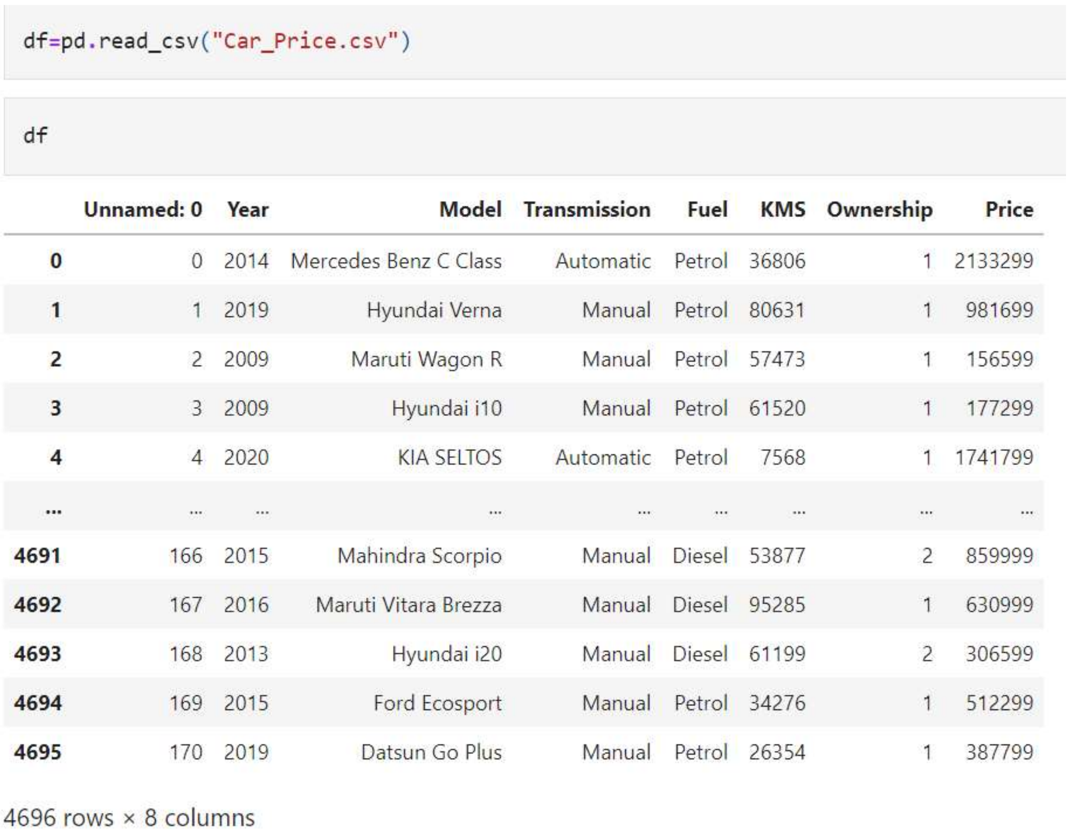
Our dataset includes a target label "Used Car Price" column and the remaining feature columns can be used to determine or help in predicting the price of the used cars. Since price is a continuous value, it makes this to be a Regression problem.

- ## Data Pre-processing Done

Following Libraries were imported for the pre-processing and ML.


```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Ridge, Lasso, ElasticNet
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.linear_model import SGDRegressor

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import power_transform

import warnings
warnings.filterwarnings('ignore')
```

Data was loaded from the .csv file and displayed for a better understanding as shown below

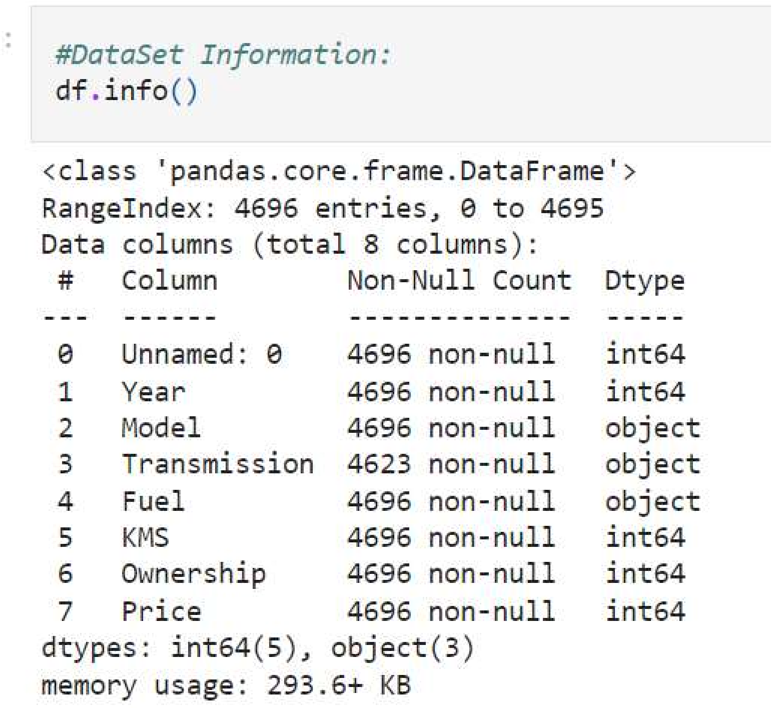

```python
df=pd.read_csv("Car_Price.csv")
```

```python
df
```

	Unnamed: 0	Year	Model	Transmission	Fuel	KMS	Ownership	Price
0	0	2014	Mercedes Benz C Class	Automatic	Petrol	36806	1	2133299
1	1	2019	Hyundai Verna	Manual	Petrol	80631	1	981699
2	2	2009	Maruti Wagon R	Manual	Petrol	57473	1	156599
3	3	2009	Hyundai i10	Manual	Petrol	61520	1	177299
4	4	2020	KIA SELTOS	Automatic	Petrol	7568	1	1741799
...	...	...	...	...	...	...	...	...
4691	166	2015	Mahindra Scorpio	Manual	Diesel	53877	2	859999
4692	167	2016	Maruti Vitara Brezza	Manual	Diesel	95285	1	630999
4693	168	2013	Hyundai i20	Manual	Diesel	61199	2	306599
4694	169	2015	Ford Ecosport	Manual	Petrol	34276	1	512299
4695	170	2019	Datsun Go Plus	Manual	Petrol	26354	1	387799

4696 rows × 8 columns

Dataset had 4696 rows and 8 columns.

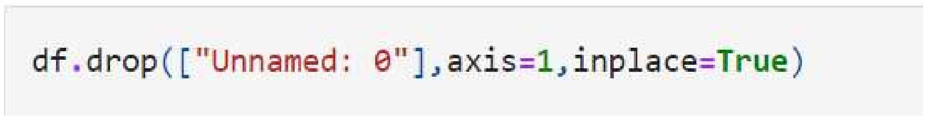


```
#DataSet Information:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4696 entries, 0 to 4695
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    4696 non-null   int64 
 1   Year          4696 non-null   int64 
 2   Model         4696 non-null   object
 3   Transmission  4623 non-null   object
 4   Fuel          4696 non-null   object
 5   KMS           4696 non-null   int64 
 6   Ownership     4696 non-null   int64 
 7   Price         4696 non-null   int64 
dtypes: int64(5), object(3)
memory usage: 293.6+ KB
```

The dataset info showed that it has 8 columns and 4696 rows. where Model, Transmission and Fuel are objective datatype, while other columns are integer datatype.

We dropped the *Unnamed:* 0 column as it had nothing but the serial number in recurring order when we scrapped the data using Selenium.



```
df.drop(["Unnamed: 0"],axis=1,inplace=True)
```

Next, we checked the null values in the dataset and found that the transmission column has few null values. After which, the null values were confirmed graphically with a heatmap.

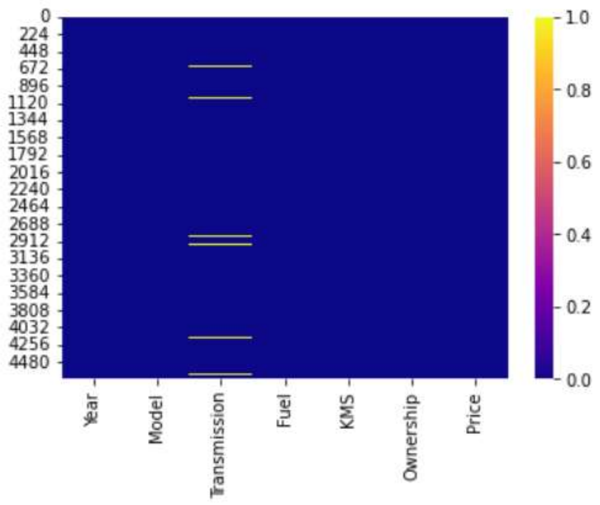
```
df.isnull().sum()
```

```
Year             0
Model            0
Transmission    73
Fuel             0
KMS              0
Ownership        0
Price            0
dtype: int64
```

Transmission column shows 73 null values.

```
sns.heatmap(df.isnull(),cmap='plasma')
```

```
<AxesSubplot:>
```



We used the Imputation method to fill those null values using the mode method, as the Transmission column is an object datatype

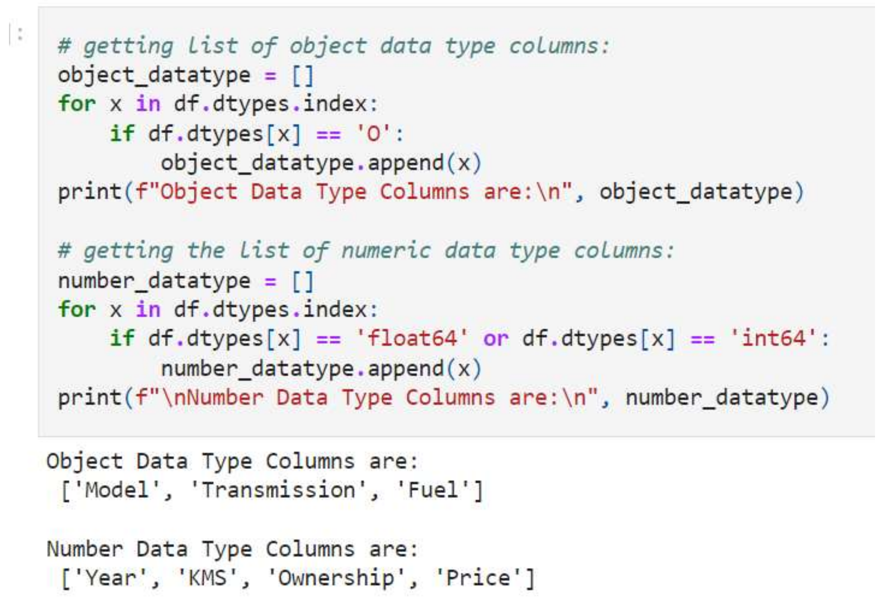
```
df['Transmission'] = df['Transmission'].fillna(df['Transmission'].mode()[0])
```

The Number of counts in each column were checked to get a better idea about the dataset columns.

```
df.nunique()
```

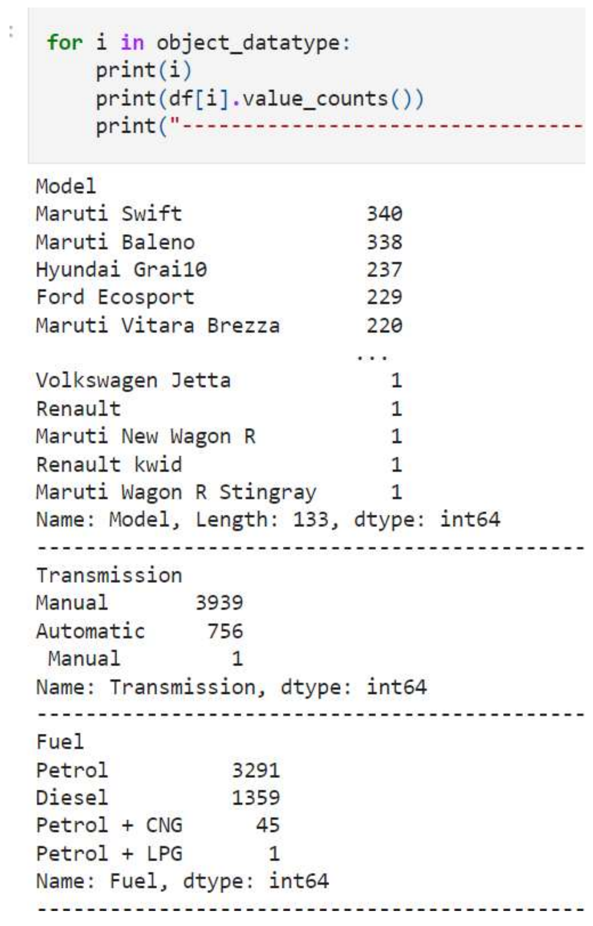
```
Year              15
Model            133
Transmission       3
Fuel               4
KMS             3349
Ownership          3
Price           2758
dtype: int64
```

Dataset columns were separated as per the datatype- Object and numerical and the counts for each column in detail was studied to see any redundancy or duplicate values in the columns.



```
# getting list of object data type columns:
object_datatype = []
for x in df.dtypes.index:
    if df.dtypes[x] == 'O':
        object_datatype.append(x)
print(f"Object Data Type Columns are:\n", object_datatype)

# getting the list of numeric data type columns:
number_datatype = []
for x in df.dtypes.index:
    if df.dtypes[x] == 'float64' or df.dtypes[x] == 'int64':
        number_datatype.append(x)
print(f"\nNumber Data Type Columns are:\n", number_datatype)
```

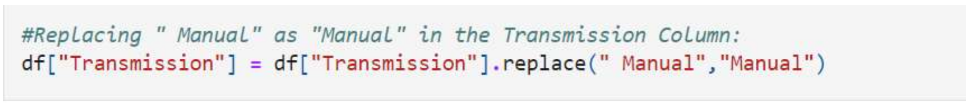
```
Object Data Type Columns are:
 ['Model', 'Transmission', 'Fuel']

Number Data Type Columns are:
 ['Year', 'KMS', 'Ownership', 'Price']
```



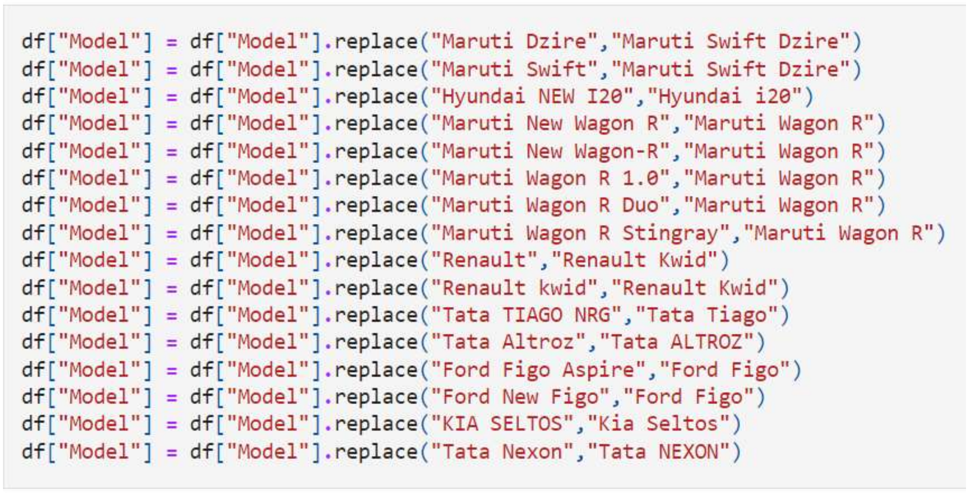
```
for i in object_datatype:
    print(i)
    print(df[i].value_counts())
    print("----------------------------------
```

```
Model
Maruti Swift              340
Maruti Baleno             338
Hyundai Grai10            237
Ford Ecosport             229
Maruti Vitara Brezza      220
                         ...
Volkswagen Jetta            1
Renault                     1
Maruti New Wagon R          1
Renault kwid                1
Maruti Wagon R Stingray     1
Name: Model, Length: 133, dtype: int64
---------------------------------------------
Transmission
Manual       3939
Automatic     756
 Manual         1
Name: Transmission, dtype: int64
---------------------------------------------
Fuel
Petrol          3291
Diesel          1359
Petrol + CNG      45
Petrol + LPG       1
Name: Fuel, dtype: int64
---------------------------------------------
```

Since the Model column showed some repetition of the car models, we clustered the repeated values under 1 name.



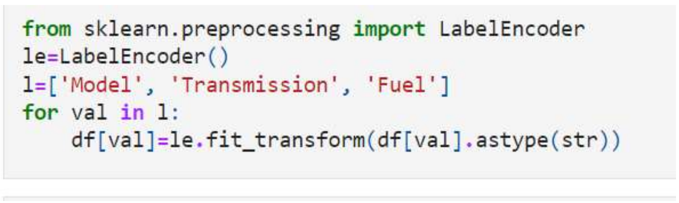
```
#Replacing " Manual" as "Manual" in the Transmission Column:
df["Transmission"] = df["Transmission"].replace(" Manual","Manual")
```



```
df["Model"] = df["Model"].replace("Maruti Dzire","Maruti Swift Dzire")
df["Model"] = df["Model"].replace("Maruti Swift","Maruti Swift Dzire")
df["Model"] = df["Model"].replace("Hyundai NEW I20","Hyundai i20")
df["Model"] = df["Model"].replace("Maruti New Wagon R","Maruti Wagon R")
df["Model"] = df["Model"].replace("Maruti New Wagon-R","Maruti Wagon R")
df["Model"] = df["Model"].replace("Maruti Wagon R 1.0","Maruti Wagon R")
df["Model"] = df["Model"].replace("Maruti Wagon R Duo","Maruti Wagon R")
df["Model"] = df["Model"].replace("Maruti Wagon R Stingray","Maruti Wagon R")
df["Model"] = df["Model"].replace("Renault","Renault Kwid")
df["Model"] = df["Model"].replace("Renault kwid","Renault Kwid")
df["Model"] = df["Model"].replace("Tata TIAGO NRG","Tata Tiago")
df["Model"] = df["Model"].replace("Tata Altroz","Tata ALTROZ")
df["Model"] = df["Model"].replace("Ford Figo Aspire","Ford Figo")
df["Model"] = df["Model"].replace("Ford New Figo","Ford Figo")
df["Model"] = df["Model"].replace("KIA SELTOS","Kia Seltos")
df["Model"] = df["Model"].replace("Tata Nexon","Tata NEXON")
```

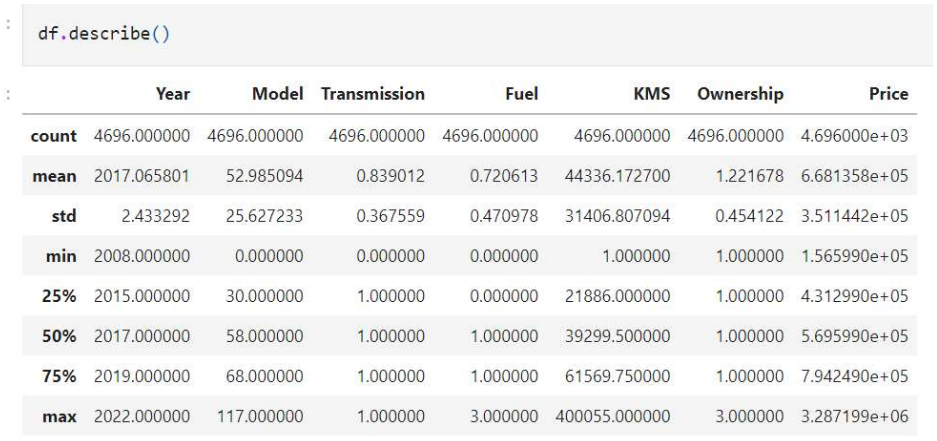
- Data Inputs- Logic- Output Relationships

We encoded the Object datatype columns using Label encoder, so Machine learning and model building becomes easier.



```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
l=['Model', 'Transmission', 'Fuel']
for val in l:
    df[val]=le.fit_transform(df[val].astype(str))
```

Moving to the statistical approach to study the dataset, we used the describe method in python to understand the variations in the dataset statistically.

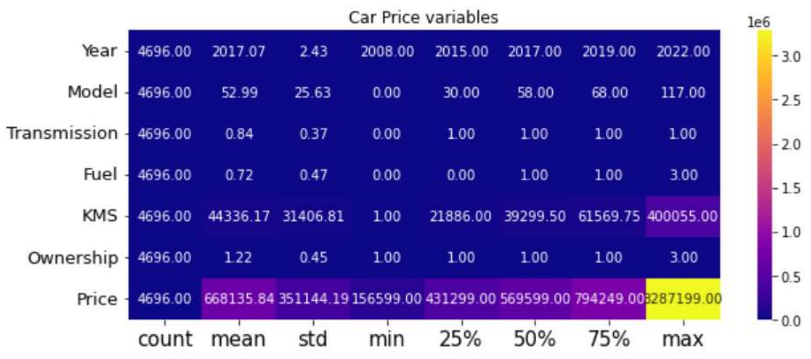


```
df.describe()
```

	Year	Model	Transmission	Fuel	KMS	Ownership	Price
count	4696.000000	4696.000000	4696.000000	4696.000000	4696.000000	4696.000000	4.696000e+03
mean	2017.065801	52.985094	0.839012	0.720613	44336.172700	1.221678	6.681358e+05
std	2.433292	25.627233	0.367559	0.470978	31406.807094	0.454122	3.511442e+05
min	2008.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.565990e+05
25%	2015.000000	30.000000	1.000000	0.000000	21886.000000	1.000000	4.312990e+05
50%	2017.000000	58.000000	1.000000	1.000000	39299.500000	1.000000	5.695990e+05
75%	2019.000000	68.000000	1.000000	1.000000	61569.750000	1.000000	7.942490e+05
max	2022.000000	117.000000	1.000000	3.000000	400055.000000	3.000000	3.287199e+06

Heatmap explains the dataset in a better was with respect to the skewness, standard deviation and outliers.

```
plt.figure(figsize=(10,4))
sns.heatmap(df.describe().transpose(),annot=True,fmt='0.2f',cmap='plasma')
plt.xticks(fontsize=15)
plt.yticks(fontsize=13)
plt.title("Car Price variables")
plt.show()
```



	count	mean	std	min	25%	50%	75%	max
Year	4696.00	2017.07	2.43	2008.00	2015.00	2017.00	2019.00	2022.00
Model	4696.00	52.99	25.63	0.00	30.00	58.00	68.00	117.00
Transmission	4696.00	0.84	0.37	0.00	1.00	1.00	1.00	1.00
Fuel	4696.00	0.72	0.47	0.00	0.00	1.00	1.00	3.00
KMS	4696.00	44336.17	31406.81	1.00	21886.00	39299.50	61569.75	400055.00
Ownership	4696.00	1.22	0.45	1.00	1.00	1.00	1.00	3.00
Price	4696.00	668135.84	351144.19	156599.00	431299.00	569599.00	794249.00	3287199.00

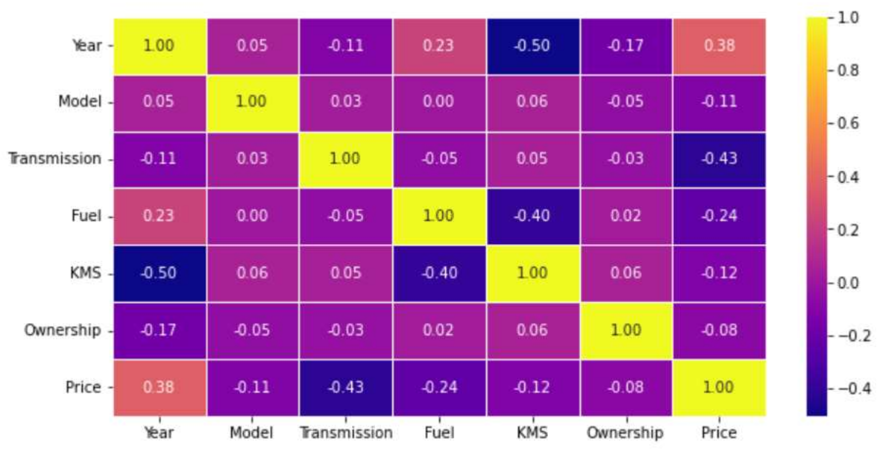
Here, we found that:

- The mean is greater than the median for the columns Price and KMS, this means that skewness is present.
- The standard deviation is high in the KMS column= data is spread
- The column KMS shows difference between 75 percentile and max, means outliers might be present.

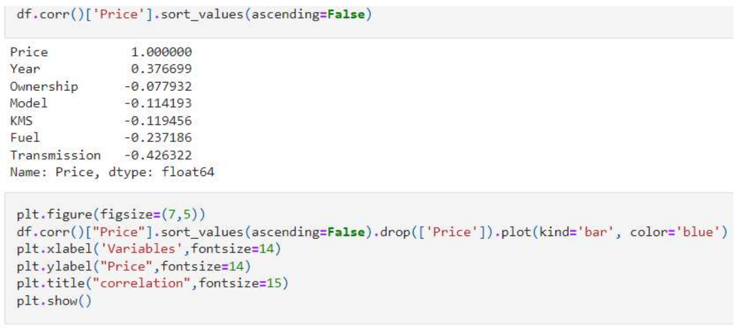
Next, we checked the correlation of the columns in the dataset with each other. Since our target was 'Price', we studied the correlation of the columns with the target.

```
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(),cmap='plasma',annot=True,linewidth=0.5,fmt='0.2f')
```

```
<AxesSubplot:>
```



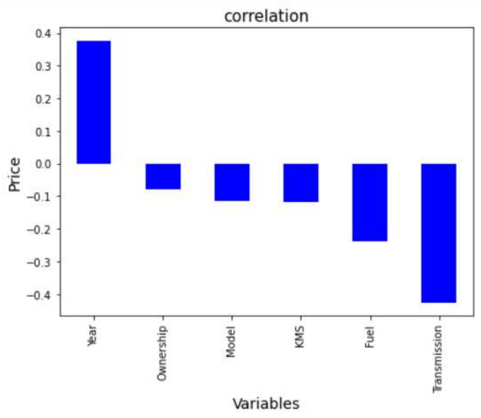
	Year	Model	Transmission	Fuel	KMS	Ownership	Price
Year	1.00	0.05	-0.11	0.23	-0.50	-0.17	0.38
Model	0.05	1.00	0.03	0.00	0.06	-0.05	-0.11
Transmission	-0.11	0.03	1.00	-0.05	0.05	-0.03	-0.43
Fuel	0.23	0.00	-0.05	1.00	-0.40	0.02	-0.24
KMS	-0.50	0.06	0.05	-0.40	1.00	0.06	-0.12
Ownership	-0.17	-0.05	-0.03	0.02	0.06	1.00	-0.08
Price	0.38	-0.11	-0.43	-0.24	-0.12	-0.08	1.00



```
df.corr()['Price'].sort_values(ascending=False)
```

```
Price           1.000000
Year            0.376699
Ownership      -0.077932
Model          -0.114193
KMS            -0.119456
Fuel           -0.237186
Transmission   -0.426322
Name: Price, dtype: float64
```

```
plt.figure(figsize=(7,5))
df.corr()["Price"].sort_values(ascending=False).drop(['Price']).plot(kind='bar', color='blue')
plt.xlabel('Variables',fontsize=14)
plt.ylabel("Price",fontsize=14)
plt.title("correlation",fontsize=15)
plt.show()
```



Following were the observations:

- Max Positive correlation with the target is seen with the column Year.
- Least negative correlation is with the columns Ownership, followed by Model and KMS.
- Maximum negative relation with target is with the column Transmission and Fuel.

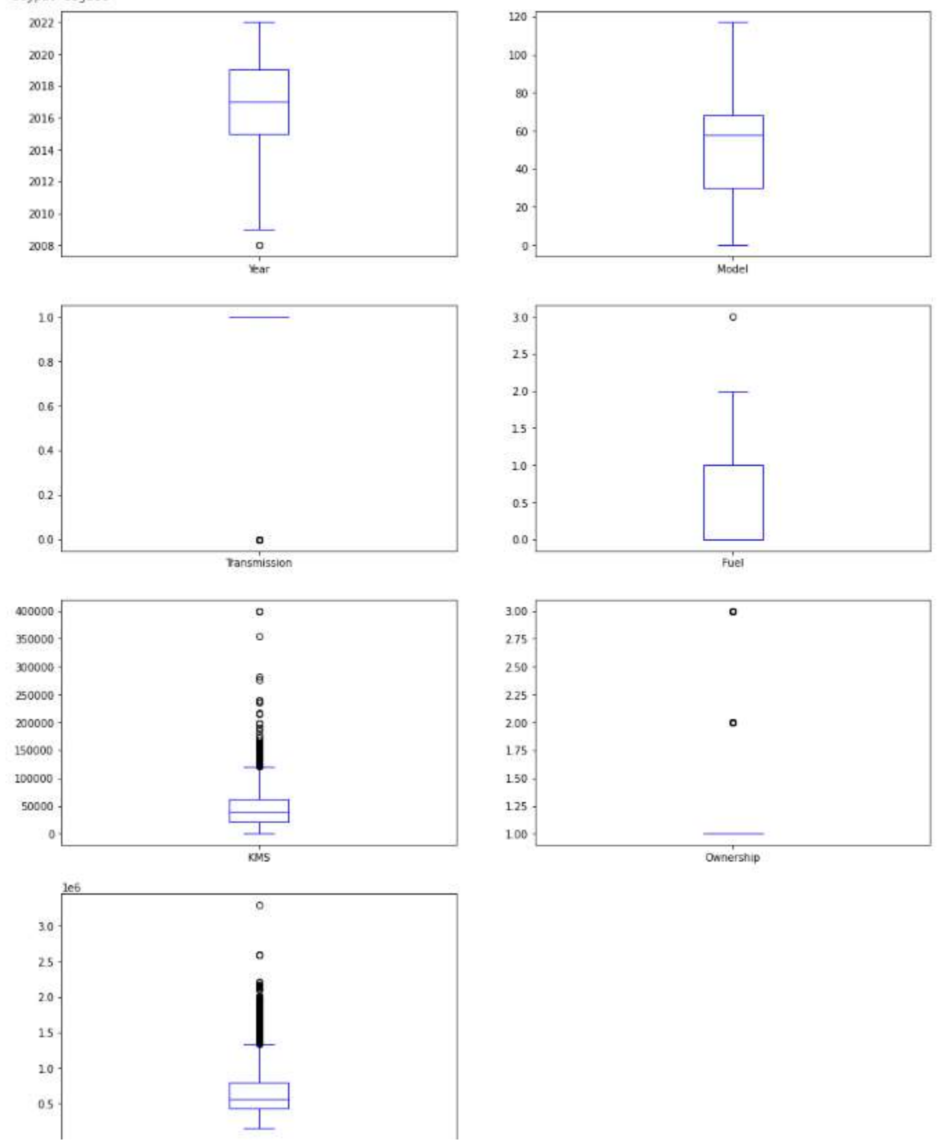
Next step was checking the skewness and the Outliers.

Outliers:

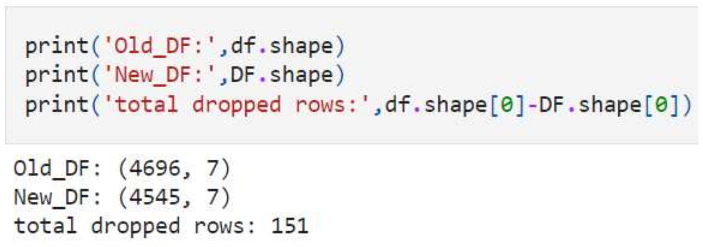
The Z score method was used for Outlier detection, where if the Z score was above 3, then that was considered as an outlier. Outliers seemed to be present in the Year, Ownership and KMS columns as these are the numerical columns.

A box plot was used to graphically view the outliers.

```
Price            AxesSubplot(0.125,0.125;0.352273x0.16413)
dtype: object
```



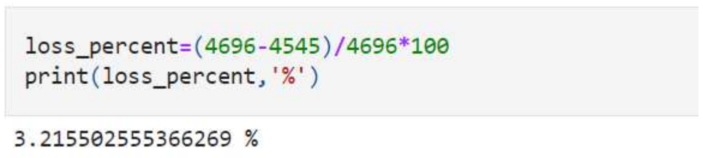
After removing the outliers, we got almost 3% data loss.



```
print('Old_DF:',df.shape)
print('New_DF:',DF.shape)
print('total dropped rows:',df.shape[0]-DF.shape[0])
```

```
Old_DF: (4696, 7)
New_DF: (4545, 7)
total dropped rows: 151
```

Percentage data loss

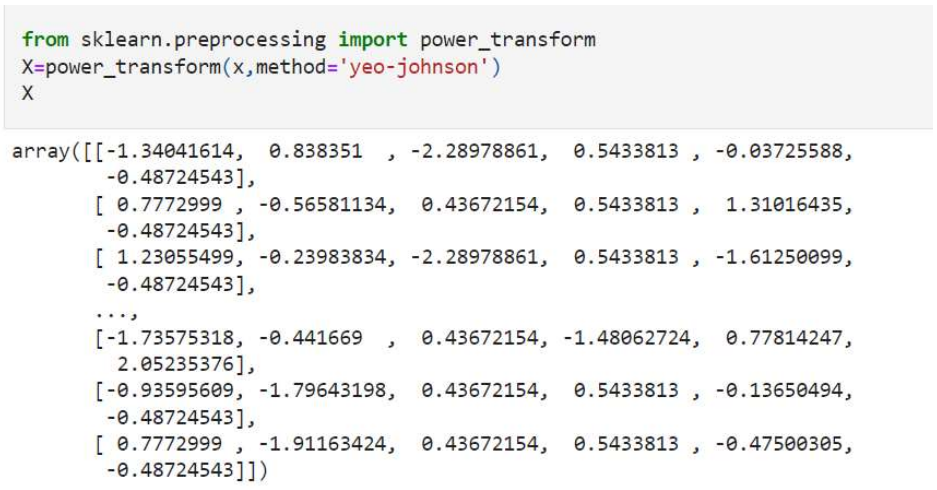


```
loss_percent=(4696-4545)/4696*100
print(loss_percent,'%')
```

```
3.215502555366269 %
```

Skewness removal:

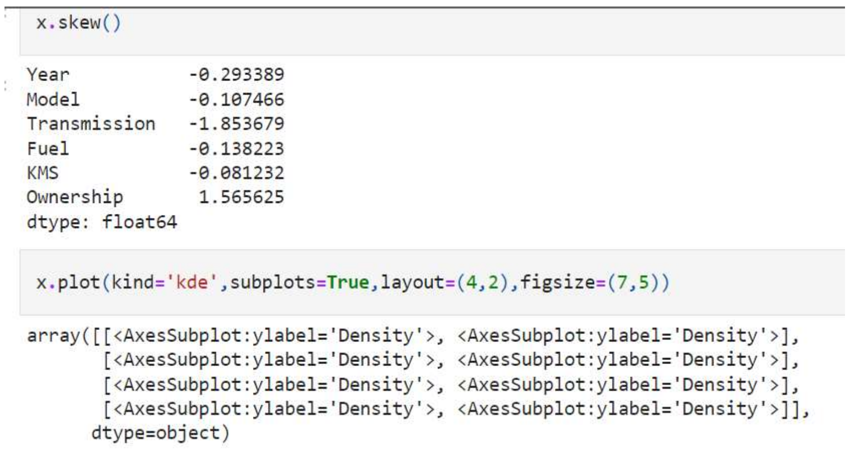
After outlier removal, we removed the skewness with the power transform method, where we used the ‘yeo-johnson’ method for skewness removal. A value above +/- 5 is generally taken as a threshold to measure the skewness. Anything above it is considered as skewed data.



```
from sklearn.preprocessing import power_transform
X=power_transform(x,method='yeo-johnson')
X
```

```
array([[-1.34041614,  0.838351  , -2.28978861,  0.5433813 , -0.03725588,
        -0.48724543],
       [ 0.7772999 , -0.56581134,  0.43672154,  0.5433813 ,  1.31016435,
        -0.48724543],
       [ 1.23055499, -0.23983834, -2.28978861,  0.5433813 , -1.61250099,
        -0.48724543],
       ...,
       [-1.73575318, -0.441669  ,  0.43672154, -1.48062724,  0.77814247,
         2.05235376],
       [-0.93595609, -1.79643198,  0.43672154,  0.5433813 , -0.13650494,
        -0.48724543],
       [ 0.7772999 , -1.91163424,  0.43672154,  0.5433813 , -0.47500305,
        -0.48724543]])
```

Finally, the skewness was checked:

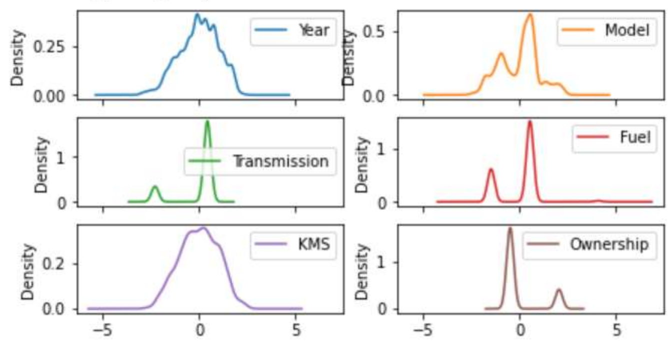


```
x.skew()
```

```
Year            -0.293389
Model           -0.107466
Transmission    -1.853679
Fuel            -0.138223
KMS             -0.081232
Ownership        1.565625
dtype: float64
```

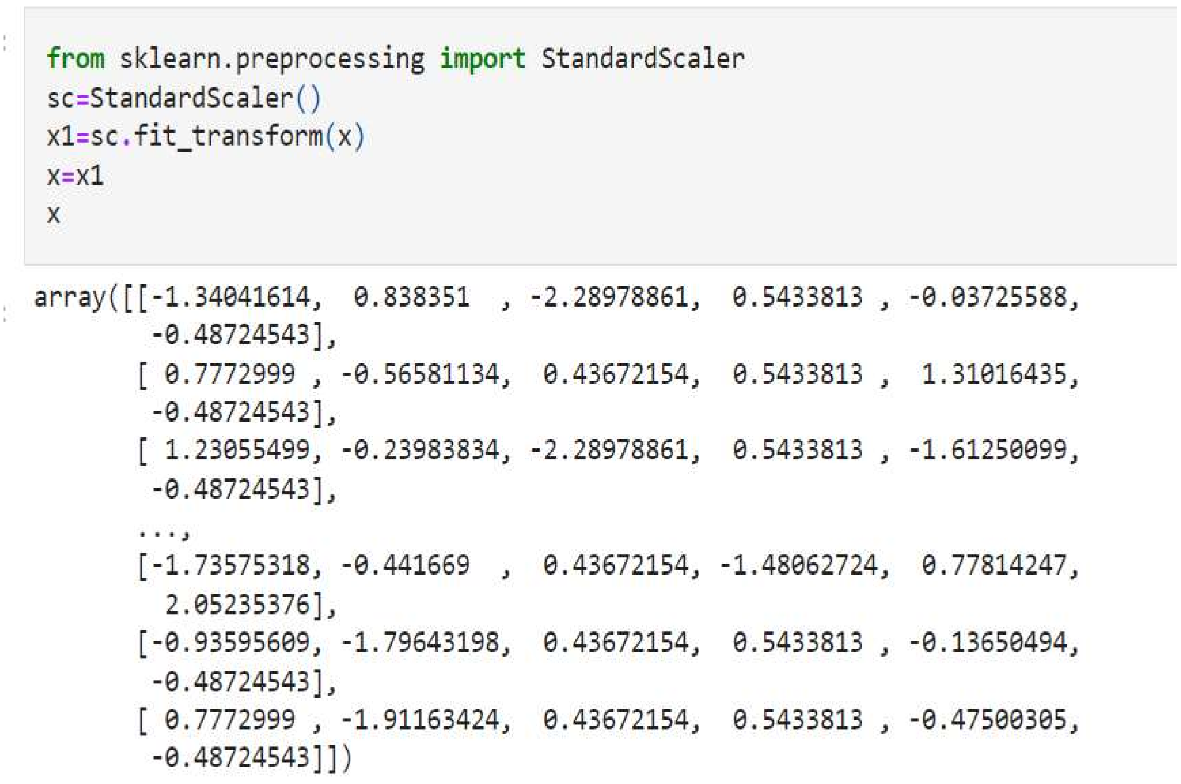
```
x.plot(kind='kde',subplots=True,layout=(4,2),figsize=(7,5))
```

```
array([[<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>]],
      dtype=object)
```



Skewness is not present anymore as seen from the above graphs.

The data was later scaled using the standard scaler which can be the used for model building.



```python
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x1=sc.fit_transform(x)
x=x1
x
```

```
array([[-1.34041614,  0.838351  , -2.28978861,  0.5433813 , -0.03725588,
        -0.48724543],
       [ 0.7772999 , -0.56581134,  0.43672154,  0.5433813 ,  1.31016435,
        -0.48724543],
       [ 1.23055499, -0.23983834, -2.28978861,  0.5433813 , -1.61250099,
        -0.48724543],
       ...,
       [-1.73575318, -0.441669  ,  0.43672154, -1.48062724,  0.77814247,
         2.05235376],
       [-0.93595609, -1.79643198,  0.43672154,  0.5433813 , -0.13650494,
        -0.48724543],
       [ 0.7772999 , -1.91163424,  0.43672154,  0.5433813 , -0.47500305,
        -0.48724543]])
```

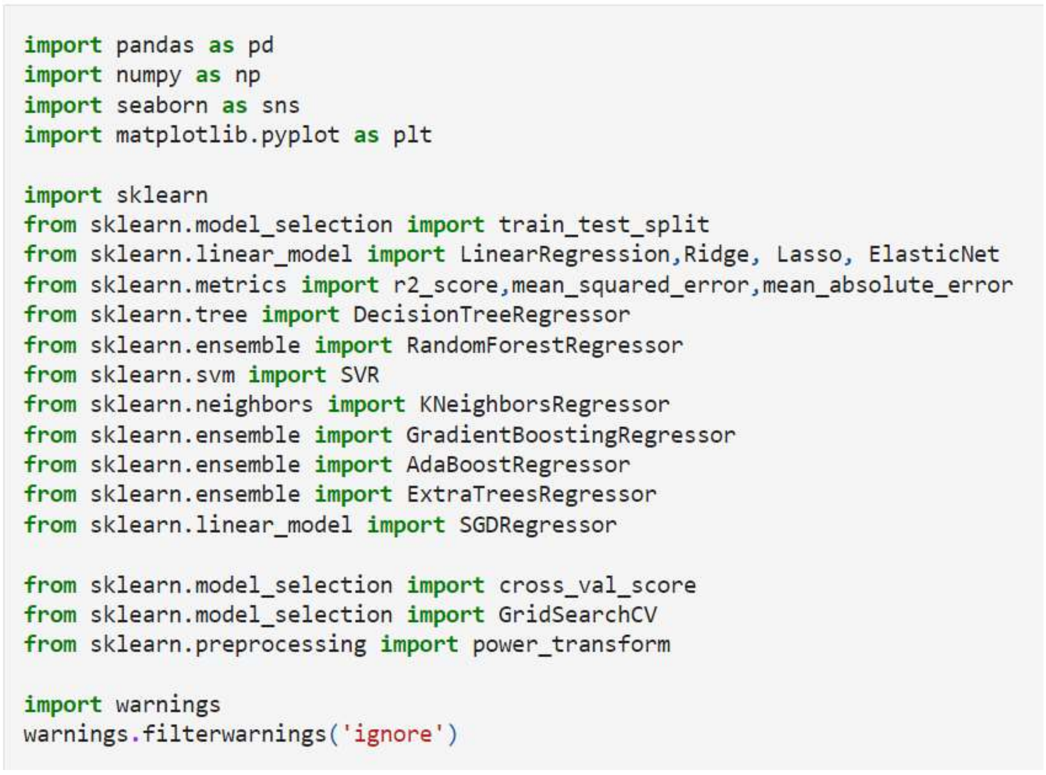
- State the set of assumptions (if any) related to the problem under consideration

  After the data was scrapped from the websites for used cars information, the data was concatenated into an excel file and then in a csv file.

  The data was later cleaned a bit in the excel format and then converted in the .csv file, which was then used for observation and model building.

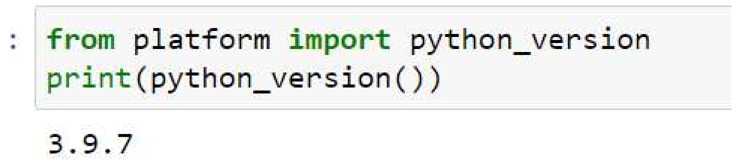
- Hardware and Software Requirements and Tools Used

  All the regression algorithms were imported in the python along with pandas, numpy, matplotlib, seaborne and Grid search CV.



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Ridge, Lasso, ElasticNet
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.linear_model import SGDRegressor

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import power_transform

import warnings
warnings.filterwarnings('ignore')
```

The python version we used was



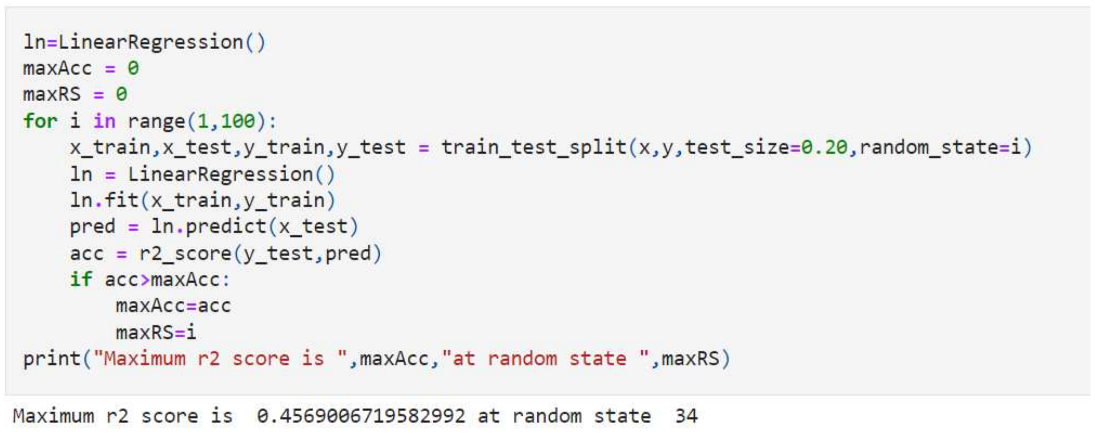
```
: from platform import python_version
  print(python_version())

3.9.7
```

Other details:
Windows 11

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

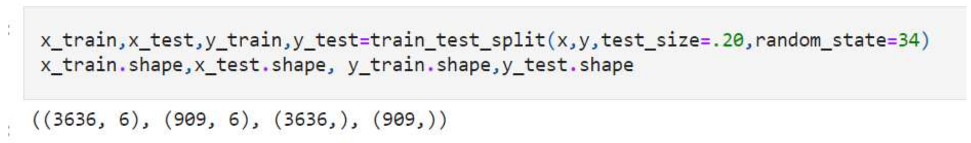
  Since the data is continuous, we will use the linear regression model building followed by ensemble methods, Cross validation score checks with R2 Scores, selecting the best Algorithm and the fine tuning it with its best parameters to get the best R2 score.



```
ln=LinearRegression()
maxAcc = 0
maxRS = 0
for i in range(1,100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=i)
    ln = LinearRegression()
    ln.fit(x_train,y_train)
    pred = ln.predict(x_test)
    acc = r2_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print("Maximum r2 score is ",maxAcc,"at random state ",maxRS)
```

```
Maximum r2 score is  0.4569006719582992 at random state  34
```

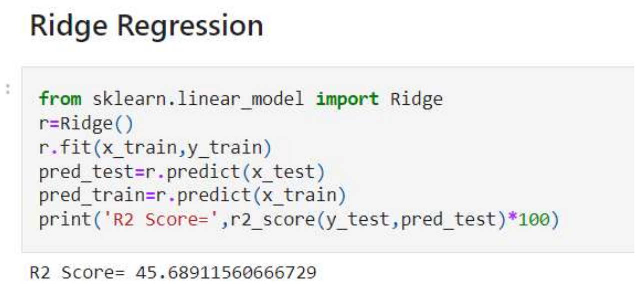
We used the regularization techniques to see which gives the best R2 score.



```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=34)
x_train.shape,x_test.shape, y_train.shape,y_test.shape
```

```
((3636, 6), (909, 6), (3636,), (909,))
```

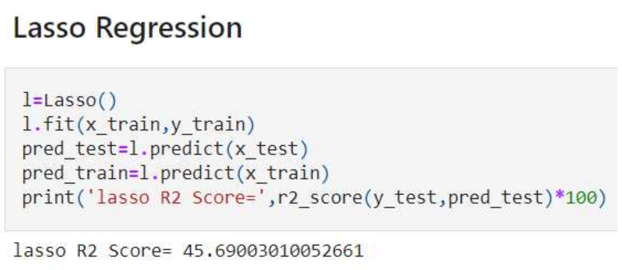
Test_train split was done with test_size 0.22, and random state 34.



Ridge Regression

```
from sklearn.linear_model import Ridge
r=Ridge()
r.fit(x_train,y_train)
pred_test=r.predict(x_test)
pred_train=r.predict(x_train)
print('R2 Score=',r2_score(y_test,pred_test)*100)
```

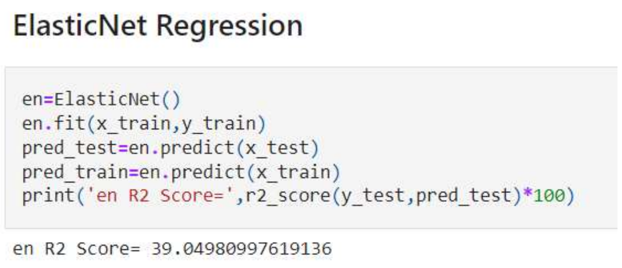
```
R2 Score= 45.68911560666729
```



Lasso Regression

```
l=Lasso()
l.fit(x_train,y_train)
pred_test=l.predict(x_test)
pred_train=l.predict(x_train)
print('lasso R2 Score=',r2_score(y_test,pred_test)*100)
```

```
lasso R2 Score= 45.69003010052661
```



ElasticNet Regression

```
en=ElasticNet()
en.fit(x_train,y_train)
pred_test=en.predict(x_test)
pred_train=en.predict(x_train)
print('en R2 Score=',r2_score(y_test,pred_test)*100)
```

```
en R2 Score= 39.04980997619136
```

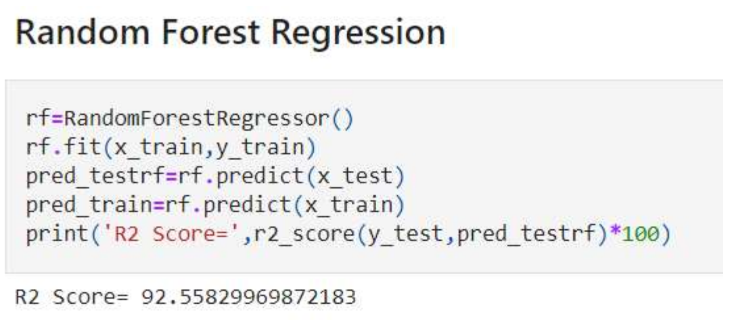
Ridge and Lasso gave the score of around 46%, while elastic net gave the R2 score of 39%.

- ## Testing of Identified Approaches (Algorithms)

Next, we checked the ensemble techniques. We used the Random Forest Regression, Support Vector Regression, KNeigbours Regression, Gradient Boosting Regression, Adaboost Regression, Extra Trees Regression, And Stochastic Gradient Descent Regression.

- ## Run and evaluate selected models

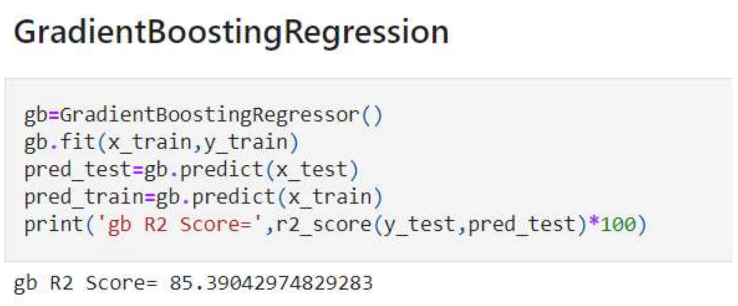
Of the listed algorithms, we found that Random Forest, Extra Trees and Gradient boosting gave the R2 scores 92.55%, 91.49% and 85.39% respectively.



Random Forest Regression

```
rf=RandomForestRegressor()
rf.fit(x_train,y_train)
pred_testrf=rf.predict(x_test)
pred_train=rf.predict(x_train)
print('R2 Score=',r2_score(y_test,pred_testrf)*100)
```

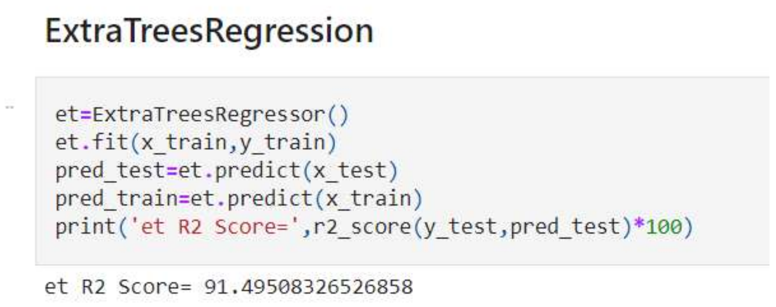
```
R2 Score= 92.55829969872183
```



GradientBoostingRegression

```
gb=GradientBoostingRegressor()
gb.fit(x_train,y_train)
pred_test=gb.predict(x_test)
pred_train=gb.predict(x_train)
print('gb R2 Score=',r2_score(y_test,pred_test)*100)
```

```
gb R2 Score= 85.39042974829283
```



ExtraTreesRegression

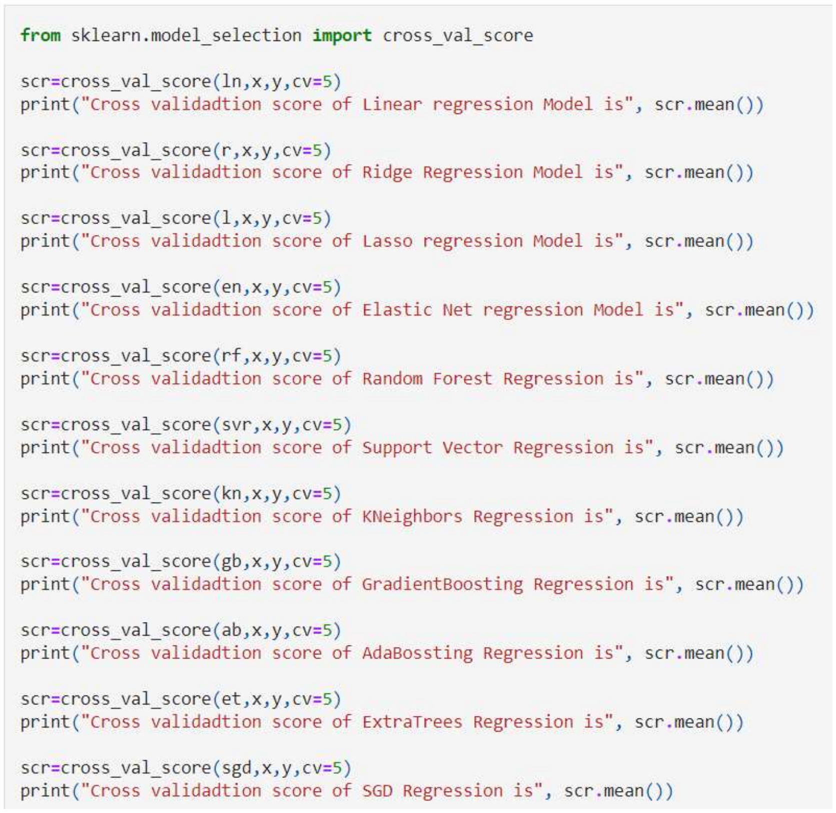
```
et=ExtraTreesRegressor()
et.fit(x_train,y_train)
pred_test=et.predict(x_test)
pred_train=et.predict(x_train)
print('et R2 Score=',r2_score(y_test,pred_test)*100)
```

```
et R2 Score= 91.49508326526858
```

We later performed the Cross validation with CV=5, and found the best possible algorithm based on the |R2Score-CV score|. The least value of this calculation will give us the best Algorithm.

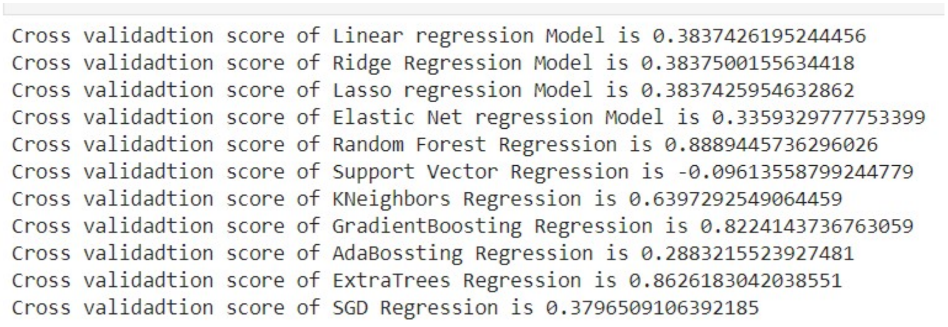
- Key Metrics for success in solving problem under consideration

  After getting the R2 score for the chosen algorithms, we found out the Cross-validation score for each algorithm.



```
from sklearn.model_selection import cross_val_score

scr=cross_val_score(ln,x,y,cv=5)
print("Cross validadtion score of Linear regression Model is", scr.mean())

scr=cross_val_score(r,x,y,cv=5)
print("Cross validadtion score of Ridge Regression Model is", scr.mean())

scr=cross_val_score(l,x,y,cv=5)
print("Cross validadtion score of Lasso regression Model is", scr.mean())

scr=cross_val_score(en,x,y,cv=5)
print("Cross validadtion score of Elastic Net regression Model is", scr.mean())

scr=cross_val_score(rf,x,y,cv=5)
print("Cross validadtion score of Random Forest Regression is", scr.mean())

scr=cross_val_score(svr,x,y,cv=5)
print("Cross validadtion score of Support Vector Regression is", scr.mean())

scr=cross_val_score(kn,x,y,cv=5)
print("Cross validadtion score of KNeighbors Regression is", scr.mean())

scr=cross_val_score(gb,x,y,cv=5)
print("Cross validadtion score of GradientBoosting Regression is", scr.mean())

scr=cross_val_score(ab,x,y,cv=5)
print("Cross validadtion score of AdaBossting Regression is", scr.mean())

scr=cross_val_score(et,x,y,cv=5)
print("Cross validadtion score of ExtraTrees Regression is", scr.mean())

scr=cross_val_score(sgd,x,y,cv=5)
print("Cross validadtion score of SGD Regression is", scr.mean())
```

  Following were the CV scores for each:



```
Cross validadtion score of Linear regression Model is 0.3837426195244456
Cross validadtion score of Ridge Regression Model is 0.3837500155634418
Cross validadtion score of Lasso regression Model is 0.3837425954632862
Cross validadtion score of Elastic Net regression Model is 0.33593297777753399
Cross validadtion score of Random Forest Regression is 0.8889445736296026
Cross validadtion score of Support Vector Regression is -0.09613558799244779
Cross validadtion score of KNeighbors Regression is 0.6397292549064459
Cross validadtion score of GradientBoosting Regression is 0.8224143736763059
Cross validadtion score of AdaBossting Regression is 0.2883215523927481
Cross validadtion score of ExtraTrees Regression is 0.8626183042038551
Cross validadtion score of SGD Regression is 0.3796509106392185
```

  Since the R2 Score of Gradient boosting Regression and Random Forest Regression were nearby to the CV score, we found out the absolute difference of R2 score and CV score. The Gradient Boosting Regression gave us the least value for the difference, and thus we concluded that Gradient Boosting Regression was the best algorithm for the model building.

Next step was to enhance the R2 score by finding the best parameters for the Gradient boosting Regression using the Grid search CV method, and hyper tuning the parameters.

```
from sklearn.model_selection import GridSearchCV

#creating parameters to pass in Grid serach for Gradient boosting
para={'criterion': ['squared_error', 'friedman_mse','mse'],'n_estimators':[100,200,300,400],
      'max_features': ['auto','sqrt','log2'],
      'max_depth': [8,9,10,11]}
```

```
GCV=GridSearchCV(GradientBoostingRegressor(),para,cv=5)
GCV.fit(x_train,y_train) #fiting the data in the model
GCV.best_params_        #printing the best parameter found by GCV
```

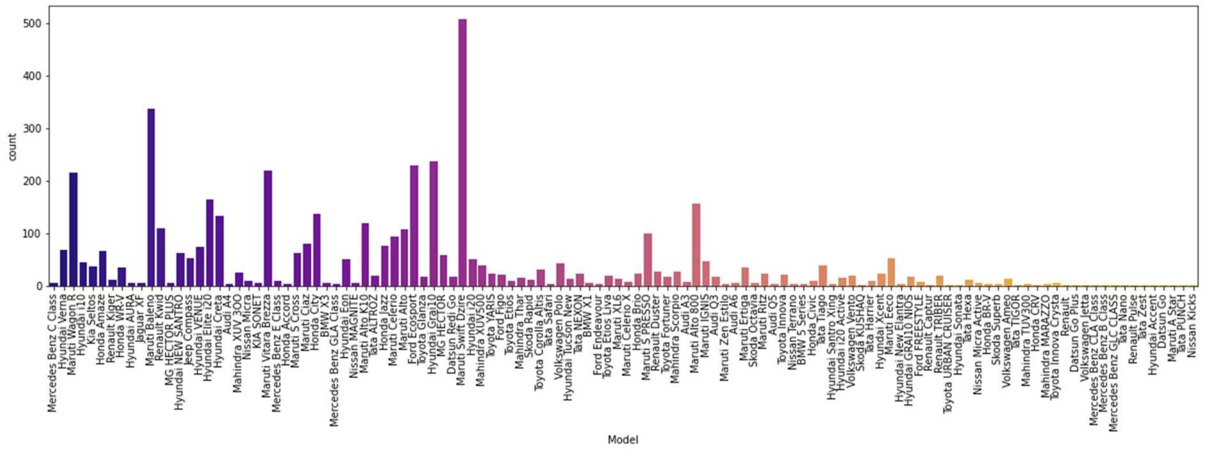
```
{'criterion': 'friedman_mse',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 200}
```

```
GB=GradientBoostingRegressor(criterion='friedman_mse', max_depth=8 ,max_features= 'auto',n_estimators= 200)
GB.fit(x_train,y_train)
predgb=GB.predict(x_test)
print('R2_score:',r2_score(y_test,predgb)*100)
print("Best R2 Score for GCV best estimator", GB, "is",r2_score(y_test,predgb)*100)
print('Error:')
print('Mean absolute error:',mean_absolute_error(y_test,predgb))
print('Mean squared error:',mean_squared_error(y_test,predgb))
print('Root mean squared error:',np.sqrt(mean_squared_error(y_test,predgb)))
```

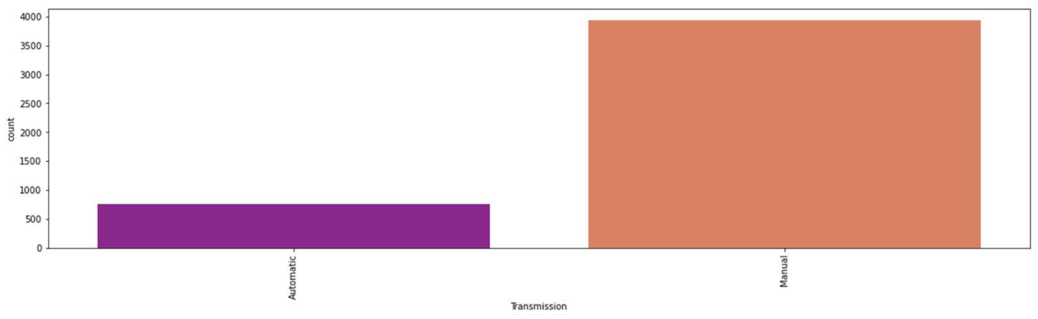
```
R2_score: 92.38906207157314
Best R2 Score for GCV best estimator GradientBoostingRegressor(max_depth=8, max_features='auto', n_estimators=200) is 92.38906207157314
Error:
Mean absolute error: 48948.58454631972
Mean squared error: 9133118147.631517
Root mean squared error: 95567.34875275927
```

- ## Visualizations

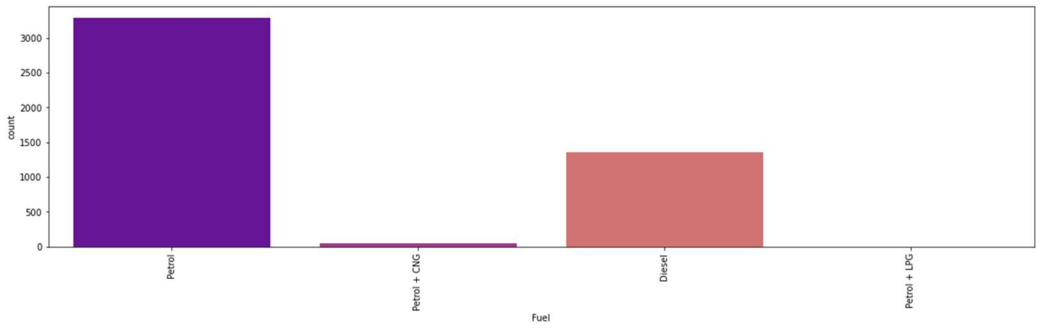
We performed Univariate, bivariate analysis on the data and plotted a pair plot.



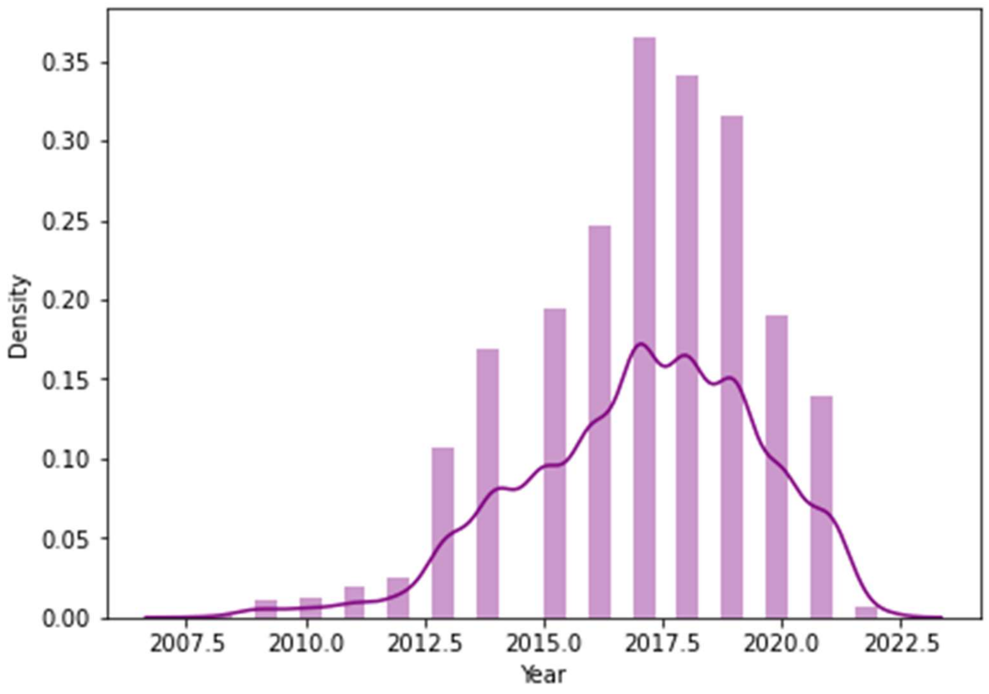
- Maruti swift dezire was the car which has the highest count in the used car category, followed by Maruti Baleno. The least count was for the luxury cars like Mercedes, Audi, Renault, Volkswagon etc.



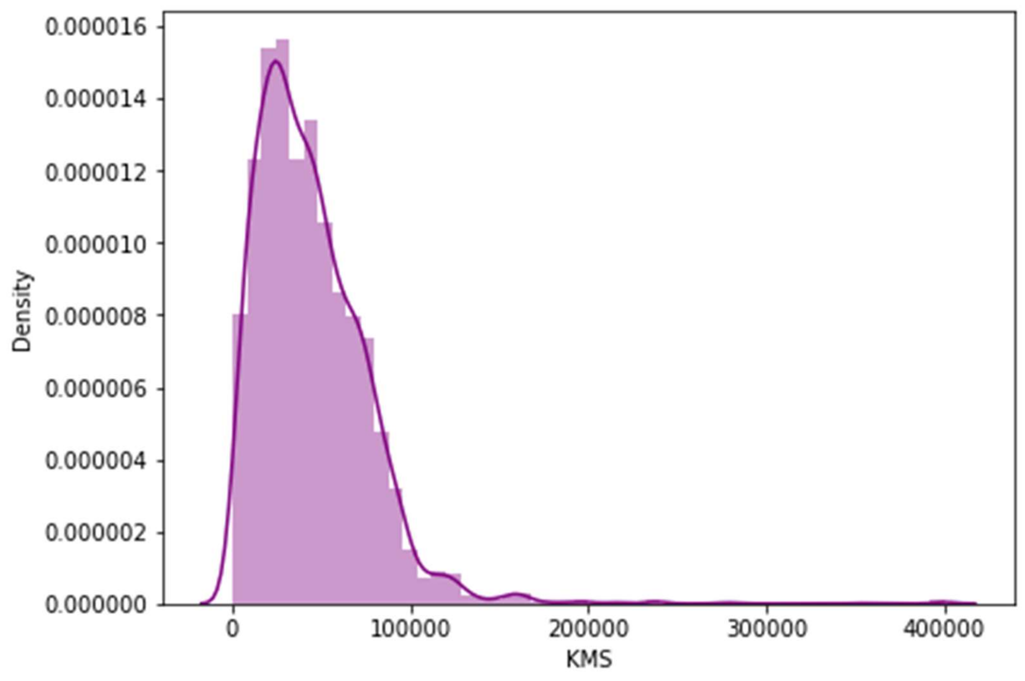
Manual Transmission has more count compared to Automatic Transmission in te used car category.



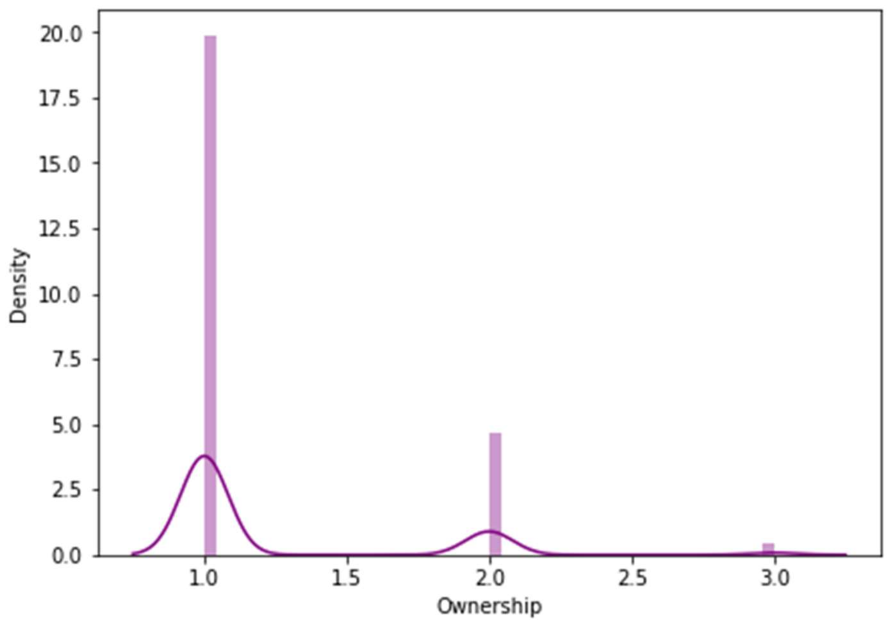
Petrol cars were most in count (above 300), while petrol+LPG were least or may be in the range of 1-2.



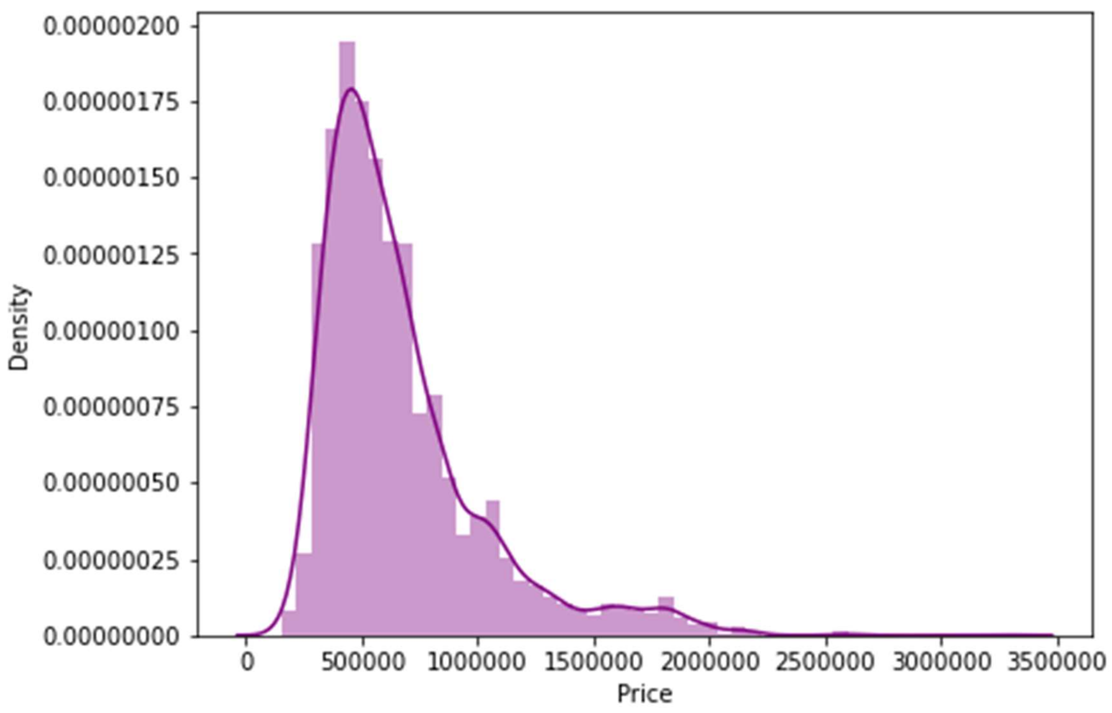
Maximum cars were from the mid-year 2017-2019, while least were from the year mid-2008 and 2022.



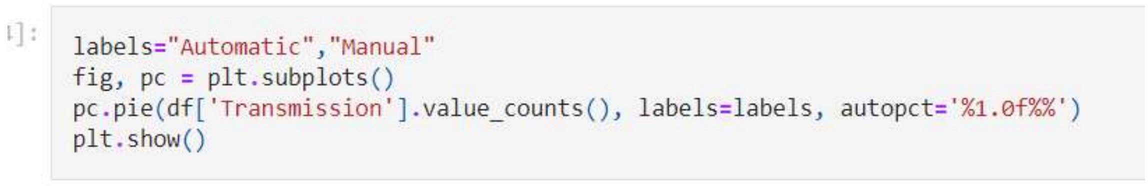
Most cars covered the distance between 20000-80000kms. Least touched the range of 200000kms. The highest density was for the kms 40000-50000.



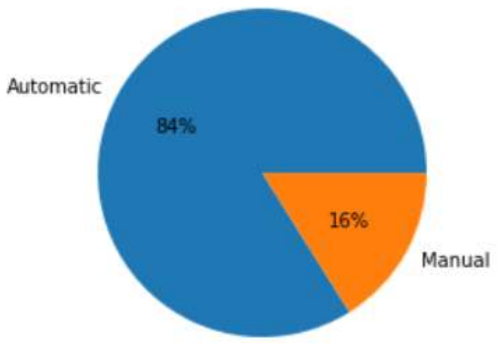
1st ownership cars were maximum in count, while 3rd ownership were the least.



The maximum price was around 2200000, while the maximum cars were having the price between 400000-800000 INR.

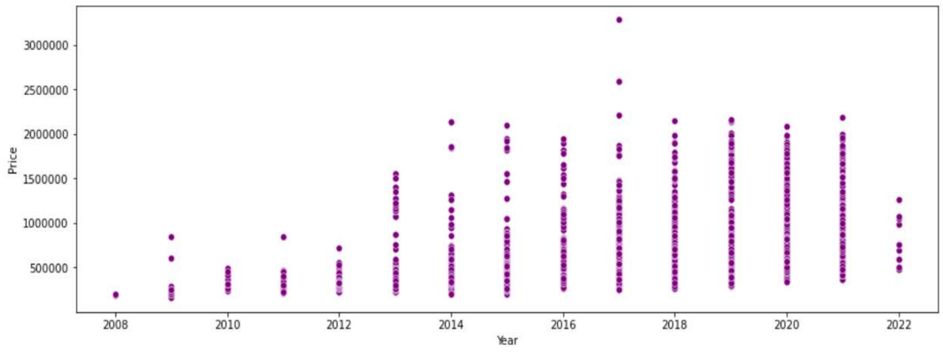


```
labels="Automatic","Manual"
fig, pc = plt.subplots()
pc.pie(df['Transmission'].value_counts(), labels=labels, autopct='%1.0f%%')
plt.show()
```



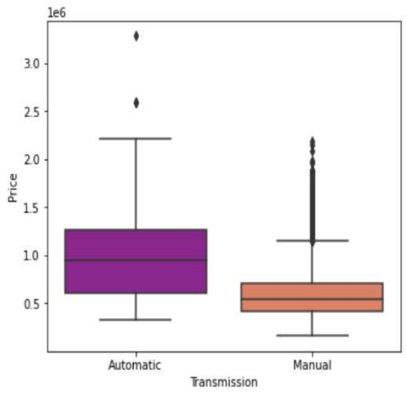
84% used cars were automatic, whereas 16% were manual.

```
plt.figure(figsize=(15,5))
sns.scatterplot(x = 'Year' ,y ='Price', color='purple', data = df)
plt.ticklabel_format(style='plain')
plt.show()
```



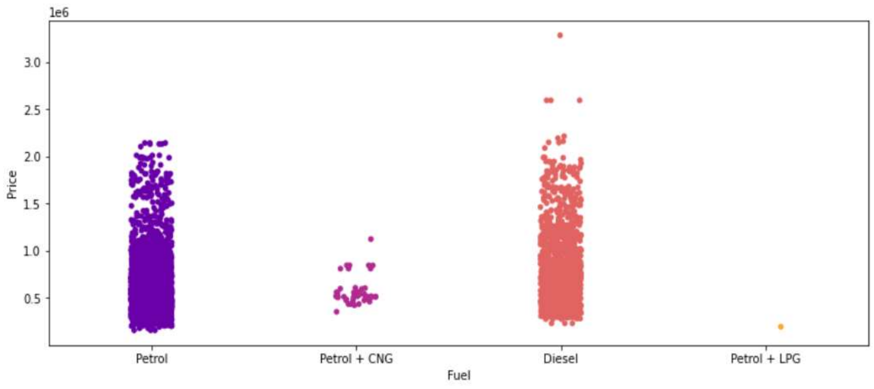
Maximum used cars were from 2019 whose price were above 3000000 INR, and those from the years 2014-2021 were in the range of 2000000-2500000 INR. Least priced were fromthe year 2008 whose price was less than 500000

```
plt.figure(figsize=(6,5))
sns.boxplot(x = 'Transmission' , y ='Price',  data = df,palette='plasma')
plt.show()
```



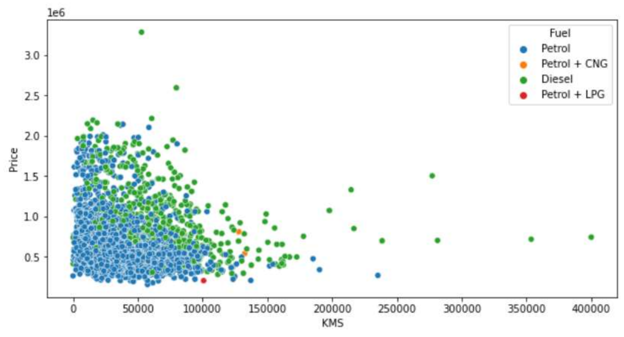
The price of cars with automatic transmission was in the range of 600000-3000000INR and above. While price for manual transmission was from 500000-around 2500000INR.

```
plt.figure(figsize=(13,5))
sns.stripplot(x = 'Fuel' , y ='Price', data = df,palette='plasma')
plt.show()
```



Cars with disel fuel were priced at about 3000000INR and above, but most density was between 400000-2000000INR. PEtrol+LPG were priced at less than 500000INR, while petrol cars were priced between 200000-2300000INR

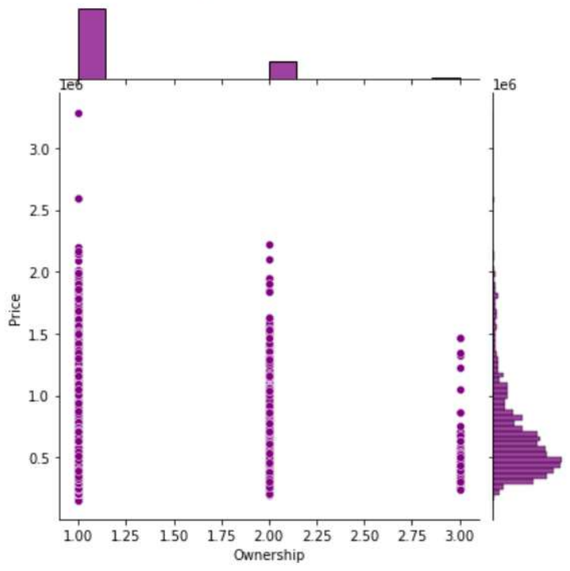
```
plt.figure(figsize=(10,5))
sns.scatterplot(x = 'KMS', y ='Price',hue='Fuel', data = df)
plt.show()
```



- More the KMS, less was the price cost. Disel cars were priced lower and were the cars with maximum kms.
- Petrol+LPG cars were having 100000kms added to them and were in the price range of 200000-300000INR, while petrol+CNG has around 140000kms and had a price range between 500000-900000INR.
- Maximum cars had petrol as fuel and covered the distance of 0-100000kms, with the pricerange from 200000-2000000INR.
- Diesel car with around 50000kms had the price of above 3000000INR.

```
plt.figure(figsize=(5,5))
sns.jointplot(x = 'Ownership' , y ='Price', color='purple', data = df)
plt.show()
```

```
<Figure size 360x360 with 0 Axes>
```



As the ownership of the cars increased the price of the cars decreased.

- Interpretation of the Results

  Visualizations: From the visualizations, it was observed that:

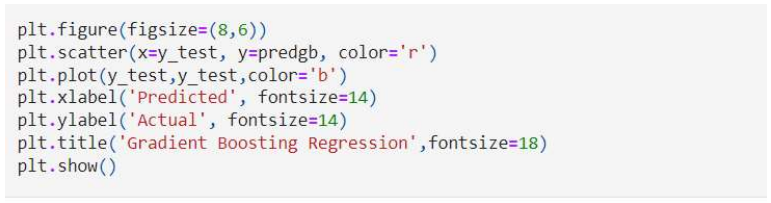
  - Price has the highest positive correlation with the Year.
  - The old the car, the lower was its price range. 3rd owner cars were less in price compared to the 1st owner cars.

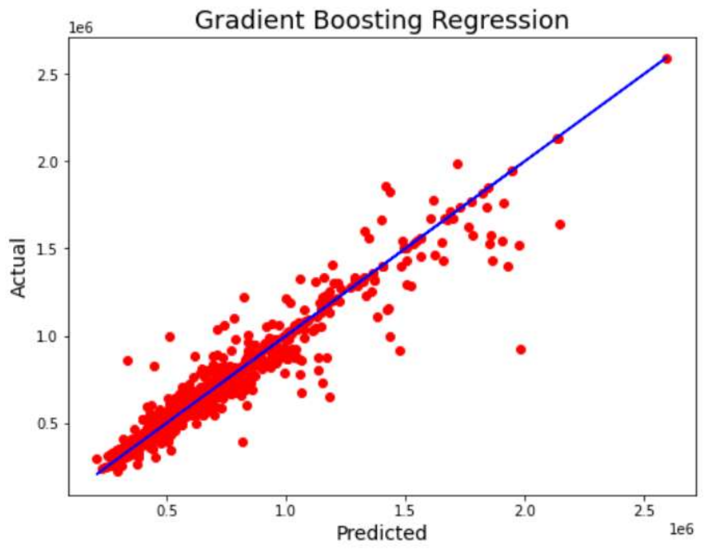
- The more the kms the car has, the less is the price of the car, since the more the car travels, more is the wear- tear and maintenance of the car.
- Automatic cars were priced in higher range than the manual operating cars because of the ease to drive the cars.

Pre-processing: In the pre-processing stage, we found that outliers and skewness were in the KMS driven, Year, and Price.

Modelling: Given the gradient boosting regression algorithm, the model's CV score was increased from 82% to 92% after hyper-tuning the model.

The best fit was plotted and it was observed that the line passed through maximum point equally.


```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=predgb, color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Predicted', fontsize=14)
plt.ylabel('Actual', fontsize=14)
plt.title('Gradient Boosting Regression',fontsize=18)
plt.show()
```

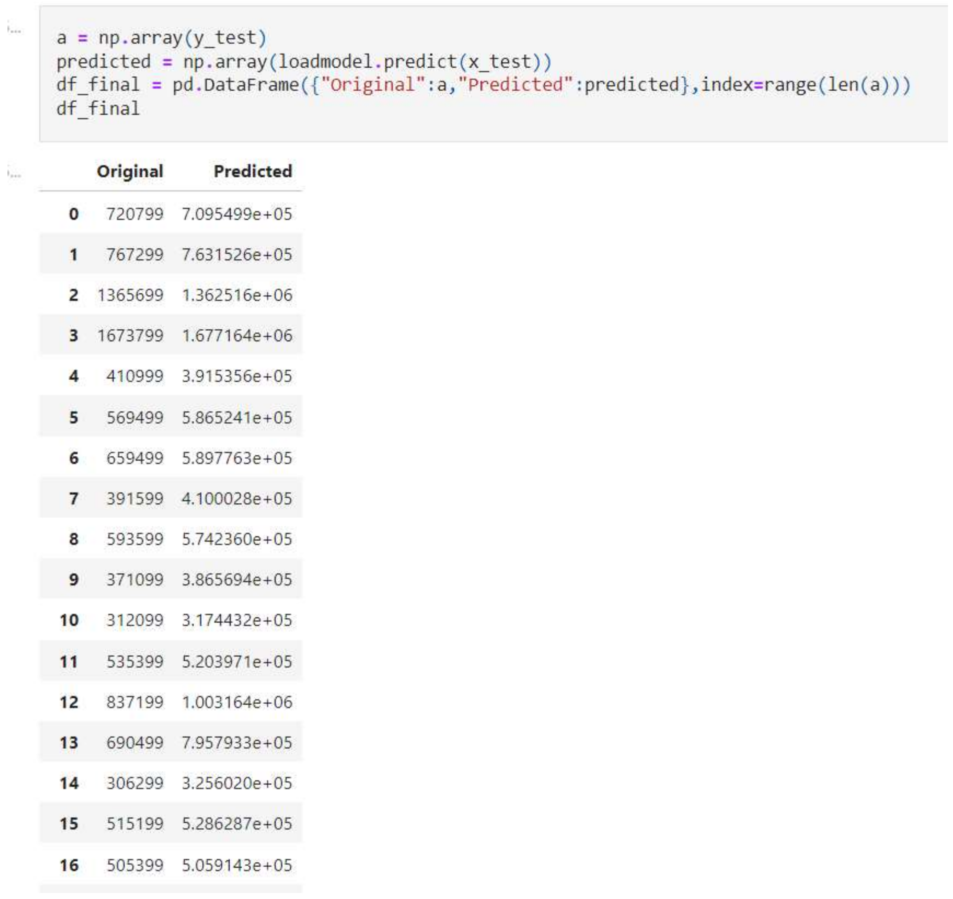


CONCLUSION

- Key Findings and Conclusions of the Study

Following key findings were found from the whole model building:

- The older the car the less will be its price range.
- Petrol and diesel cars were more preferred than the hybrid petrol+CNG, petrol+LPG.
- As the ownership of the cars increases, the price of the car decreases.
- Year of when the car was purchased didn’t matter much as 5-6 years old cars had same value.
- Automatic cars are priced more because of the ease to use it as compared to the manual cars.
- Luxury cars were priced higher, although the number of these cars which were sold were low.
- The price of the used cars was predicted with 92% accuracy after model building.
- The model was saved in the pickle (.pkl) format and the following were the concluded results with the original.



```
a = np.array(y_test)
predicted = np.array(loadmodel.predict(x_test))
df_final = pd.DataFrame({"Original":a,"Predicted":predicted},index=range(len(a)))
df_final
```

	Original	Predicted
0	720799	7.095499e+05
1	767299	7.631526e+05
2	1365699	1.362516e+06
3	1673799	1.677164e+06
4	410999	3.915356e+05
5	569499	5.865241e+05
6	659499	5.897763e+05
7	391599	4.100028e+05
8	593599	5.742360e+05
9	371099	3.865694e+05
10	312099	3.174432e+05
11	535399	5.203971e+05
12	837199	1.003164e+06
13	690499	7.957933e+05
14	306299	3.256020e+05
15	515199	5.286287e+05
16	505399	5.059143e+05

Learning Outcomes of the Study in respect of Data Science

- Obtain, clean/process, and transform data.
- Analyse and interpret data using an ethically responsible approach.
- Use appropriate models of analysis, assess the quality of input, derive insight from results, and investigate potential issues.
- Apply computing theory, languages, and algorithms, as well as mathematical and statistical models, and the principles of optimization to appropriately formulate and use data analysis.
- Formulate and use appropriate models of data analysis to solve hidden solutions to business-related challenges.
- Price Prediction modelling – This allows predicting the prices of used cars & how they are varying considering the different factors affecting the prices in the real time scenarios.
- Prediction of Price – This helps to predict the future price range based on inputs from the past and different types of factors related to used cars. This is best done using predictive data analytics to calculate the future values of the used cars.
- Deployment of ML models – The Machine learning models can also predict the car prices depending upon the needs of the buyers and recommend them, so customers can make final decisions as per the needs.

Limitations of this work and Scope for Future Work

Probably more columns would have made to build a model considering many factors on which the price of the used cars will depend on.