Machine Learning Case Study
Car Prediction Price Analysis

Submitted by,
M.Kavitha.

ACKNOWLEDGMENT

This presentation includes the Car price prediction done by myself with reference to the data analysis prepared by me using web scraping from the website Cars24 .Also referred to Google for some detailed learning in the analysis report writing for the completion of the 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. 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

In this section,We scraped the data of used cars from websites (Cars24).We used web scraping for this. We fetched data for different locations. The number of columns for data depends on the proper scraping we are doing and also the website which we are scraping .

The dataset contains the data of the used car.On the basis of the data we have to predict the price of the car.The dataset contains the data like 'name', 'year', 'transmission', 'mileage', 'owner', 'fuel', 'price', 'model', 'location' details of the used car.

Review of Literature:

I started the Research by first reading and analyzing the data collected by myself . SalePrice is the target column here. All the features are then analyzed, missing data handling, outlier detection, data cleaning are done. New features are extracted, redundant features dropped and categorical features are encoded accordingly. Then the data in split into train and test data and feature scaling is performed.

· **Motivation for the Problem Undertaken**

The main Objective behind the project is to perform the given task successfully and analyze the dataset thoroughly, learn the objective concepts and perform the prediction according to the provided dataset.

Analytical Problem Framing

· Mathematical/ Analytical Modeling of the Problem

1.Importing modules, Reading the data
2.Analyzing Numerical Features
   Checking Statistical summary
   Checking Distribution of numerical features .
   Inspecting Correlation
   Missing Value Handling
   Encoding Categorical Features
   Correcting data type
   Univariate and Bivariate analysis,
   DataVisualization.
3.Analyzing Categorical Features

Missing Value Handling
Encoding Categorical Features
Data Visualization
Dropping Redundant Features
4.Splitting data into Train and Test data
5.Comparing model coefficients
6.Model Evaluation
7.Choosing the final model and most significant features.
8.Evaluation of Regressor Models
9.HyperParameter Tuning.
10.Conclusion.

Data Sources and their formats

Data contains 5984 entries each having 9 variables.

• Data does not have Null values.
• Extensive EDA is performed to gain relationships of important variables and price.
• Data contains numerical as well as categorical variables.
• We have to build Machine Learning models, apply Regressor model coefficients and determine the optimal values of Hyper Parameters.

• We need to find important features which affect the price.

Data Description:

First, we will import the required libraries:

Importing Dataset

Importing the Libraries

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

# for model building
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import RFE
import statsmodels.api as sm
# for model evaluation
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
# for suppressing warnings
import warnings
warnings.filterwarnings("ignore")
```

Importing the Dataset

```
]: df=pd.read_csv('final_car.csv',delimiter='\t')
df
```

]:

	name	year	transmission	mileage	owner	fuel	price	model	location
0	Maruti Alto 800	2017	Manual	11,658 km	1st Owner	Petrol	₹2,93,899	LXI Manual	ahmedabad
1	Maruti Alto K10	2016	Manual	10,179 km	1st Owner	Petrol	₹3,31,599	VXI Manual	ahmedabad
2	Maruti Swift	2015	Manual	31,933 km	1st Owner	Diesel	₹4,59,199	VDI ABS Manual	ahmedabad
3	Maruti Baleno	2020	Manual	4,560 km	1st Owner	Petrol	₹6,75,699	DELTA 1.2 K12 Manual	ahmedabad
4	Maruti Celerio	2018	Manual	8,663 km	1st Owner	Petrol	₹4,60,199	ZXI Manual	ahmedabad
...	...	...	...	...	...	...	...	...	...
5979	Hyundai i20	2012	Manual	1,06,458 km	1st Owner	Petrol	₹3,24,599	SPORTZ 1.2 O Manual	Noida
5980	Hyundai Verna	2014	Manual	1,29,478 km	2nd Owner	Diesel	₹5,58,199	FLUIDIC 1.6 SX CRDI Manual	Noida
5981	Maruti Ertiga	2015	Manual	1,23,016 km	1st Owner	Diesel	₹5,41,399	VDI ABS Manual	Noida
5982	Ford Ecosport	2013	Manual	1,55,432 km	1st Owner	Diesel	₹4,06,699	1.5TITANIUM TDCI Manual	Noida
5983	Maruti Swift	2015	Manual	1,11,801 km	1st Owner	Diesel	₹4,49,299	VDI ABS Manual	Noida

Once the data is collected ,we perform several steps to explore the data.The Aim of this step is to get the better understanding of the data structure,do initial preprocessing,clean the data,check for skewness,outliers,missing values,do encoding ,standard scale the dataset and finally build the model.

Understanding the data:

In the first part of the dataframe is evaluated for structure,columns,data types.we use basic pandas functions to perform these steps.

We start by analyzing the pricing of the car and find out which variable is important in selecting a used car for the best driving and significant in predicting a price of a new car.
The data provided consists of car model ,types and price,Also it tells about the fuel type whether it is petrol,diesel.
In the collected dataset there were no null values and duplicate records .

Exploratory Data Analysis

In Data Analysis We will Analyze To Find out the below stuff.

1.Missing Values

2.All The Numerical Variables

3.Distribution of the Numerical Variables

4.Categorical Variables

5.Cardinality of Categorical Variables

6.Outliers

7.Relationship between independent and dependent feature(SalePrice)

```
df.shape
```

```
(5984, 9)
```

Missing Values

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

```
name            0
year            0
transmission    0
mileage         0
owner           0
fuel            0
price           0
model           0
location        0
dtype: int64
```

```
df.columns
```

```
Index(['name', 'year', 'transmission', 'mileage', 'owner', 'fuel', 'price',
       'model', 'location'],
      dtype='object')
```

Checking the datatypes of the columns

```
df.dtypes
```

```
name            object
year             int64
transmission    object
mileage         object
owner           object
fuel            object
price           object
model           object
location        object
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5984 entries, 0 to 5983
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5984 entries, 0 to 5983
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   name          5984 non-null   object
 1   year          5984 non-null   int64 
 2   transmission  5984 non-null   object
 3   mileage       5984 non-null   object
 4   owner         5984 non-null   object
 5   fuel          5984 non-null   object
 6   price         5984 non-null   object
 7   model         5984 non-null   object
 8   location      5984 non-null   object
dtypes: int64(1), object(8)
memory usage: 420.9+ KB
```

]:

```
print(df['owner'].unique())
print(df['fuel'].unique())
print(df['transmission'].unique())
print(df['location'].unique())
```

```
['1st Owner' '3rd Owner' '2nd Owner' '4th Owner']
['Petrol' 'Diesel' 'Petrol + CNG' 'Petrol + LPG' 'Electric']
['Manual' 'Automatic']
['ahmedabad' 'bengaluru' 'delhi' 'mumbai' 'hyderabad' 'Noida']
```

Describe the dataset

```
df.describe()
```

	year
count	5984.000000
mean	2014.936330
std	2.915454
min	2007.000000
25%	2013.000000
50%	2015.000000
75%	2017.000000
max	2021.000000

Data Preprocessing

```
df['price']=df['price'].str.replace("₹"," ").str.replace(",","").astype("float32")
```

```
df
```

	name	year	transmission	mileage	owner	fuel	price	model	location
0	Maruti Alto 800	2017	Manual	11,658 km	1st Owner	Petrol	293899.0	LXI Manual	ahmedabad
1	Maruti Alto K10	2016	Manual	10,179 km	1st Owner	Petrol	331599.0	VXI Manual	ahmedabad
2	Maruti Swift	2015	Manual	31,933 km	1st Owner	Diesel	459199.0	VDI ABS Manual	ahmedabad
3	Maruti Baleno	2020	Manual	4,560 km	1st Owner	Petrol	675699.0	DELTA 1.2 K12 Manual	ahmedabad
4	Maruti Celerio	2018	Manual	8,663 km	1st Owner	Petrol	460199.0	ZXI Manual	ahmedabad
...	...	...	...	...	...	...	...	...	...
5979	Hyundai i20	2012	Manual	1,06,458 km	1st Owner	Petrol	324599.0	SPORTZ 1.2 O Manual	Noida
5980	Hyundai Verna	2014	Manual	1,29,478 km	2nd Owner	Diesel	558199.0	FLUIDIC 1.6 SX CRDI Manual	Noida
5981	Maruti Ertiga	2015	Manual	1,23,016 km	1st Owner	Diesel	541399.0	VDI ABS Manual	Noida
5982	Ford Ecosport	2013	Manual	1,55,432 km	1st Owner	Diesel	406699.0	1.5TITANIUM TDCI Manual	Noida

```
df['mileage']=df['mileage'].str.replace("km","").str.replace(",","").astype("float32")
```

```
df
```

	name	year	transmission	mileage	owner	fuel	price	model	location
0	Maruti Alto 800	2017	Manual	11658.0	1st Owner	Petrol	293899.0	LXI Manual	ahmedabad
1	Maruti Alto K10	2016	Manual	10179.0	1st Owner	Petrol	331599.0	VXI Manual	ahmedabad
2	Maruti Swift	2015	Manual	31933.0	1st Owner	Diesel	459199.0	VDI ABS Manual	ahmedabad
3	Maruti Baleno	2020	Manual	4560.0	1st Owner	Petrol	675699.0	DELTA 1.2 K12 Manual	ahmedabad
4	Maruti Celerio	2018	Manual	8663.0	1st Owner	Petrol	460199.0	ZXI Manual	ahmedabad

```
df.describe()
```

	year	mileage	price
count	5984.000000	5984.000000	5.984000e+03
mean	2014.936330	56539.781250	5.103656e+05
std	2.915454	41845.160156	3.202498e+05
min	2007.000000	23.000000	8.919900e+04
25%	2013.000000	28543.750000	3.103990e+05
50%	2015.000000	50166.000000	4.305990e+05
75%	2017.000000	76553.000000	6.080490e+05
max	2021.000000	969664.000000	4.725000e+06

```
df['Current Year']=2020
```

```
df
```

	name	year	transmission	mileage	owner	fuel	price	model	location	Current Year
0	Maruti Alto 800	2017	Manual	11658.0	1st Owner	Petrol	293899.0	LXI Manual	ahmedabad	2020
1	Maruti Alto K10	2016	Manual	10179.0	1st Owner	Petrol	331599.0	VXI Manual	ahmedabad	2020
2	Maruti Swift	2015	Manual	31933.0	1st Owner	Diesel	459199.0	VDI ABS Manual	ahmedabad	2020
3	Maruti Baleno	2020	Manual	4560.0	1st Owner	Petrol	675699.0	DELTA 1.2 K12 Manual	ahmedabad	2020
4	Maruti Celerio	2018	Manual	8663.0	1st Owner	Petrol	460199.0	ZXI Manual	ahmedabad	2020

```
df['no_year']=df['Current Year']- df['year']
```

```
df
```

	name	year	transmission	mileage	owner	fuel	price	model	location	Current Year	no_year
0	Maruti Alto 800	2017	Manual	11658.0	1st Owner	Petrol	293899.0	LXI Manual	ahmedabad	2020	3
1	Maruti Alto K10	2016	Manual	10179.0	1st Owner	Petrol	331599.0	VXI Manual	ahmedabad	2020	4
2	Maruti Swift	2015	Manual	31933.0	1st Owner	Diesel	459199.0	VDI ABS Manual	ahmedabad	2020	5
3	Maruti Baleno	2020	Manual	4560.0	1st Owner	Petrol	675699.0	DELTA 1.2 K12 Manual	ahmedabad	2020	0
4	Maruti Celerio	2018	Manual	8663.0	1st Owner	Petrol	460199.0	ZXI Manual	ahmedabad	2020	2
...	...	...	...	...	...	...	...	...	...	...	...
5979	Hyundai i20	2012	Manual	106458.0	1st Owner	Petrol	324599.0	SPORTZ 1.2 O Manual	Noida	2020	8
5980	Hyundai Verna	2014	Manual	129478.0	2nd Owner	Diesel	558199.0	FLUIDIC 1.6 SX CRDI Manual	Noida	2020	6
5981	Maruti Ertiga	2015	Manual	123016.0	1st Owner	Diesel	541399.0	VDI ABS Manual	Noida	2020	5
5982	Ford Ecosport	2013	Manual	155432.0	1st Owner	Diesel	406699.0	1.5TITANIUM TDCI Manual	Noida	2020	7
5983	Maruti Swift	2015	Manual	111801.0	1st Owner	Diesel	449299.0	VDI ABS Manual	Noida	2020	5

```
df.drop(['year'],axis=1,inplace=True)
```

```
df
```

	name	transmission	mileage	owner	fuel	price	model	location	Current Year	no_year
0	Maruti Alto 800	Manual	11658.0	1st Owner	Petrol	293899.0	LXI Manual	ahmedabad	2020	3
1	Maruti Alto K10	Manual	10179.0	1st Owner	Petrol	331599.0	VXI Manual	ahmedabad	2020	4
2	Maruti Swift	Manual	31933.0	1st Owner	Diesel	459199.0	VDI ABS Manual	ahmedabad	2020	5
3	Maruti Baleno	Manual	4560.0	1st Owner	Petrol	675699.0	DELTA 1.2 K12 Manual	ahmedabad	2020	0
4	Maruti Celerio	Manual	8663.0	1st Owner	Petrol	460199.0	ZXI Manual	ahmedabad	2020	2
...	...	...	...	...	...	...	...	...	...	...
5979	Hyundai i20	Manual	106458.0	1st Owner	Petrol	324599.0	SPORTZ 1.2 O Manual	Noida	2020	8
5980	Hyundai Verna	Manual	129478.0	2nd Owner	Diesel	558199.0	FLUIDIC 1.6 SX CRDI Manual	Noida	2020	6
5981	Maruti Ertiga	Manual	123016.0	1st Owner	Diesel	541399.0	VDI ABS Manual	Noida	2020	5
5982	Ford Ecosport	Manual	155432.0	1st Owner	Diesel	406699.0	1.5TITANIUM TDCI Manual	Noida	2020	7
5983	Maruti Swift	Manual	111801.0	1st Owner	Diesel	449299.0	VDI ABS Manual	Noida	2020	5

```python
for col in df.columns:
    print(col,': ',len(df[col].unique()), 'labels')
```

```
name :  157 labels
transmission :  2 labels
mileage :  4787 labels
owner :  4 labels
fuel :  5 labels
price :  3547 labels
model :  847 labels
location :  6 labels
Current Year :  1 labels
no_year :  15 labels
```

```python
import seaborn as sns
```

```python
ax = sns.barplot(x="price",data=df)
print(df["model"].value_counts())
```

```
VXI Manual                     568
LXI Manual                     483
VDI Manual                     342
VDI BS IV Manual               152
VDI ABS Manual                 123
                              ...
Trendline 1.0 L Petrol           1
SLE BS IV Manual                 1
1.8 Z3 Automatic                 1
1.6 TDI MT AMBITION Manual       1
ASTA 1.2 KAPPA2 Manual           1
Name: model, Length: 847, dtype: int64
```

Made all these observations from the exploratory data analysis.

Encoding of DataFrame

```
category_vars = ['transmission','owner','fuel','location']
```

```
df_dummied = pd.get_dummies(df[category_vars], drop_first = True)
```

```
df.drop(['name','transmission','owner','fuel','model','location','Current Year'], axis = 1, inplace = True)
```

```
result = pd.concat([df, df_dummied], axis=1)
```

```
result.head()
```

	mileage	price	no_year	transmission_Manual	owner_2nd Owner	owner_3rd Owner	owner_4th Owner	fuel_Electric	fuel_Petrol	fuel_Petrol + CNG	fuel_Petrol + LPG	location_ahmedabad
0	11658.0	293899.0	3	1	0	0	0	0	1	0	0	1
1	10179.0	331599.0	4	1	0	0	0	0	1	0	0	1
2	31933.0	459199.0	5	1	0	0	0	0	0	0	0	1
3	4560.0	675699.0	0	1	0	0	0	0	1	0	0	1

Correlation

```
result.corr()
```

	mileage	price	no_year	transmission_Manual	owner_2nd Owner	owner_3rd Owner	owner_4th Owner	fuel_Electric	fuel_Petrol	fuel_Petrol + CNG	fuel_Petrol + LPG
mileage	1.000000	-0.118474	0.424592	0.091699	0.094773	0.056336	0.015161	0.003371	-0.386300	0.052569	0.043767
price	-0.118474	1.000000	-0.490028	-0.380753	-0.076393	-0.061737	0.040755	-0.011045	-0.275710	-0.048488	-0.044080
no_year	0.424592	-0.490028	1.000000	0.168554	0.208834	0.128216	0.034193	0.004152	-0.002172	-0.011102	0.085166
transmission_Manual	0.091699	-0.380753	0.168554	1.000000	-0.012411	0.000497	-0.066495	0.005184	-0.044267	0.024761	0.018712
owner_2nd Owner	0.094773	-0.076393	0.208834	-0.012411	1.000000	-0.080713	-0.019923	-0.006296	0.000535	-0.007302	-0.004485
owner_3rd Owner	0.056336	-0.061737	0.128216	0.000497	-0.080713	1.000000	-0.006781	-0.002143	0.017367	0.044885	0.036766
owner_4th Owner	0.015161	0.040755	0.034193	-0.066495	-0.019923	-0.006781	1.000000	-0.000529	-0.002601	0.031370	-0.001909
fuel_Electric	0.003371	-0.011045	0.004152	0.005184	-0.006296	-0.002143	-0.000529	1.000000	-0.016894	-0.001486	-0.000603
fuel_Petrol	-0.386300	-0.275710	-0.002172	-0.044267	0.000535	0.017367	-0.002601	-0.016894	1.000000	-0.150178	-0.060975
fuel_Petrol + CNG	0.052569	-0.048488	-0.011102	0.024761	-0.007302	0.044885	0.031370	-0.001486	-0.150178	1.000000	-0.005362

Getting Statistical Analysis for the numerical features:

Describe the dataset

```
result.describe()
```

	mileage	price	no_year	transmission_Manual	owner_2nd Owner	owner_3rd Owner	owner_4th Owner	fuel_Electric	fuel_Petrol	fuel_Petrol + CNG	fuel_P +
count	5984.000000	5.984000e+03	5984.000000	5984.000000	5984.000000	5984.000000	5984.000000	5984.000000	5984.000000	5984.000000	5984.00
mean	56539.781250	5.103656e+05	5.063670	0.861464	0.191678	0.026738	0.001671	0.000167	0.630682	0.013035	0.00
std	41845.160156	3.202498e+05	2.915454	0.345491	0.393654	0.161330	0.040849	0.012927	0.482661	0.113433	0.04
min	23.000000	8.919900e+04	-1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	28543.750000	3.103990e+05	3.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	50166.000000	4.305990e+05	5.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.00
75%	76553.000000	6.080490e+05	7.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.00
max	969664.000000	4.725000e+06	13.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

Comments:

Derived the new column current year ,so that i can create a new column number of years the car was used in accordance with dataset created and dropped the current year column and the year column.,I make a note of

Derived new variables as and when necessary.

Also converted the mileage column from km to m conversion as string type.

Converted the price column from rupees to float32 for the ease of calculation.

derive the variables as and when required and convert the variables to the correct data type.

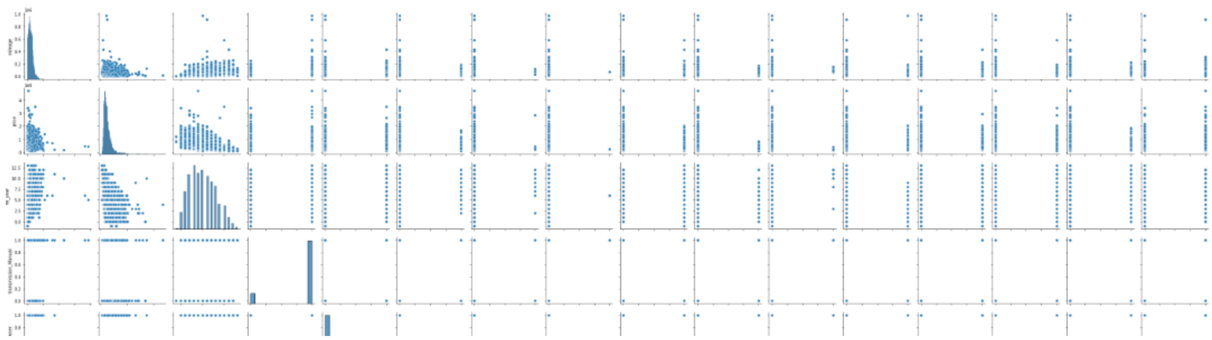
Also checked the company name and model name of the car which varies according to the price of the car.

Visualization of the Data

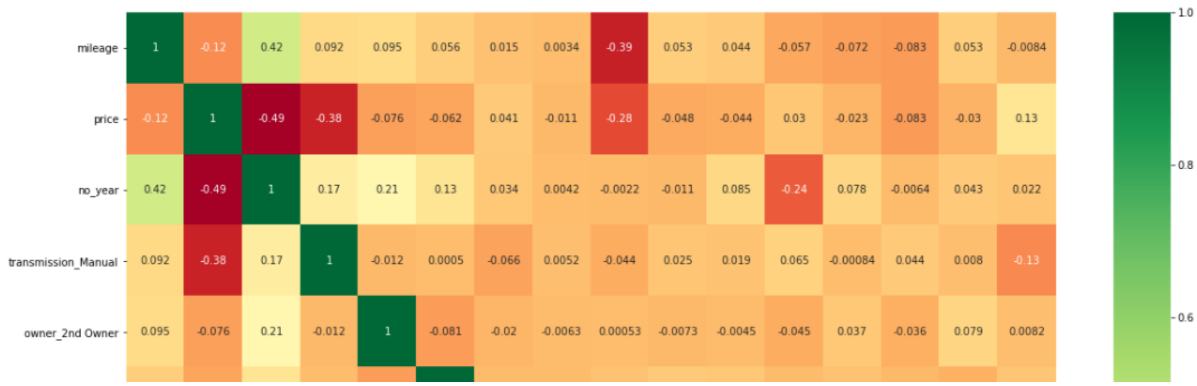
```
import seaborn as sns
```

```
sns.pairplot(result)
```

```
<seaborn.axisgrid.PairGrid at 0xca54b50>
```



```
import seaborn as sns
#get correlations of each features in dataset
corrmat = result.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(result[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

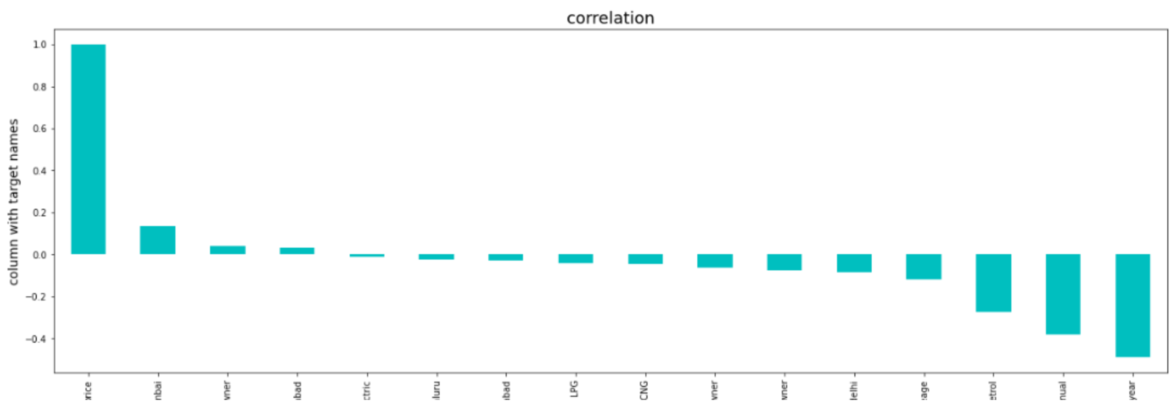


I checked the summary statistics of numeric variables.
Did the correlation plot of all the numeric variables.to understand which are positively correlated and which variables are negatively correlated.
The price column is highly correlated,and the features with dark green color are highly positively

correlated. The columns with red color are negatively correlated. The column fuel type petrol is negatively correlated.

Data Visualization:

```
plt.figure(figsize=(22,7))
result.corr()["price"].sort_values(ascending=False).plot(kind='bar',color='c')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```



Splitting of the X and Y datasets

```
X=result.iloc[:,2:]
y=result.iloc[:,1]
```

```
X.head()
```

	no_year	transmission_Manual	owner_2nd Owner	owner_3rd Owner	owner_4th Owner	fuel_Electric	fuel_Petrol	fuel_Petrol + CNG	fuel_Petrol + LPG	location_ahmedabad	location_bengaluru
0	3	1	0	0	0	0	1	0	0	1	0
1	4	1	0	0	0	0	1	0	0	1	0
2	5	1	0	0	0	0	0	0	0	1	0
3	0	1	0	0	0	0	1	0	0	1	0
4	2	1	0	0	0	0	1	0	0	1	0

Model Development and Evaluation

]:
```
y.head()
```

]:
```
0    293899.0
1    331599.0
2    459199.0
3    675699.0
4    460199.0
Name: price, dtype: float32
```

]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.linear_model import LinearRegression
```

]:
```
lr=LinearRegression(normalize=True)
lr.fit(X_train,y_train)
```

]:
```
LinearRegression(normalize=True)
```

:
```
lr=LinearRegression(normalize=True)
lr.fit(X_train,y_train)
```

:
```
LinearRegression(normalize=True)
```


:
```
lr_pred=lr.predict(X_test)
```

:
```
lr_pred
```

:
```
array([424973.92452638, 678489.00403757, 557302.69399649, ...,
       810817.77350767, 709411.7417032 , 656193.99923    ])
```

:
```
lr_accuracy=round(lr.score(X_train,y_train)*100)
lr_accuracy
```

:
```
46
```

:
```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, lr_pred)
```

:
```
150815.9183452683
```

Run and Evaluate selected models

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, lr_pred)
```

```
150815.9183452683
```

```
print("RMSE VALUE = ",mean_squared_error(y_test, lr_pred,squared = False))
```

```
RMSE VALUE =  249905.99631845043
```

```
import numpy as np
o = np.array(y_test)
```

Original and Predicted Price

```
print("original price  is " ,o[0])
print("predicted average price is " , lr_pred[0])
```

```
original price  is  303499.0
predicted average price is  424973.9245263827
```

Regressor

```
### Feature Importance

from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
```

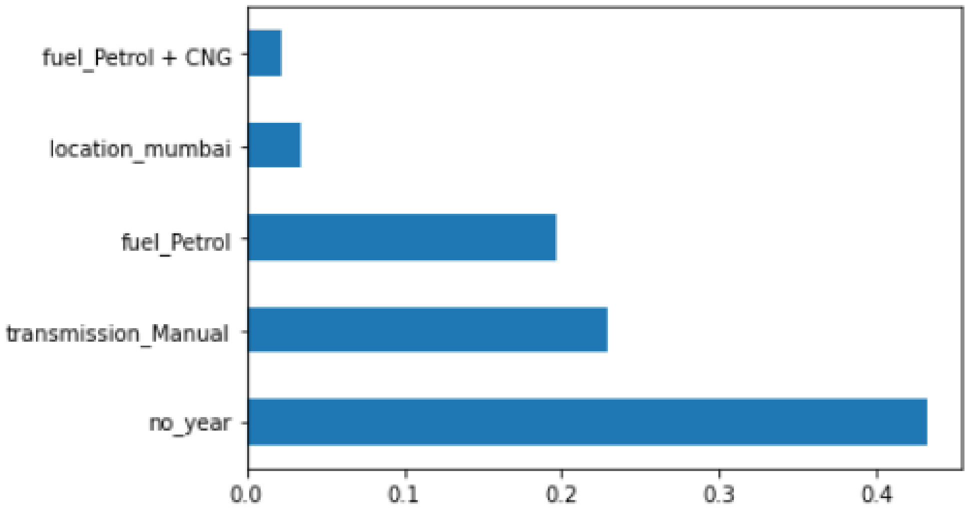
```
ExtraTreesRegressor()
```

```
print(model.feature_importances_)
```

```
[4.31744956e-01 2.29391385e-01 2.04736644e-02 4.21668136e-03
 1.52176249e-02 3.31663954e-04 1.96537735e-01 2.23675006e-02
 1.51947365e-03 1.07930181e-02 1.89922698e-02 4.81088692e-03
 9.77997420e-03 3.38231656e-02]
```

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



KNeighborsRegressor,SVR,DecisionTreeRegressor,RandomForestRegressor

]:
```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

#for RMSLE we will create  own scorer
from sklearn.metrics import make_scorer
```

]:
```

def score(y_pred,y):
    y_pred = np.log(y_pred)
    y = np.log(y)
    return 1 - ((np.sum((y_pred-y)**2))/len(y))**1/2    # 1-RMSLE

# make  own scorer
scorer = make_scorer(score,greater_is_better=True, needs_proba=False)
```

]:
```
knn_reg = KNeighborsRegressor()
svm_reg = SVR(gamma='scale')
dt_reg = DecisionTreeRegressor()
rf_reg = RandomForestRegressor()
```

```
knn_reg = KNeighborsRegressor()
svm_reg = SVR(gamma='scale')
dt_reg = DecisionTreeRegressor()
rf_reg = RandomForestRegressor()
```

```
#Training,Testing
for reg in (knn_reg, svm_reg, dt_reg, rf_reg):
    reg.fit(X_train, y_train)

    y_pred = reg.predict(X_test)

    print(reg, score(y_pred,y_test))
```

```
KNeighborsRegressor() 0.935662454379959
SVR() 0.8665403417654844
DecisionTreeRegressor() 0.9408112935872491
RandomForestRegressor() 0.9434264969944597
```

HyperParameter Tuning

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

```

regressor=RandomForestRegressor()
```

```

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
print(n_estimators)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

```
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
# Minimum number of samples required to split a node
```

```python
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

```python
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features': ['auto', 'sqrt'], 'max_dept
h': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}
```

:
```python
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
```

:
```python
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 10,
```

:
```python
rf_random.fit(X_train,y_train)
```

```
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.2s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.4s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.2s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.3s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.2s

[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  2.1min finished
```

:
```python
rf_random.fit(X_train,y_train)
```

```
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.2s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.3s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=   3.2s

[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  2.1min finished
```

:
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
                                                         900, 1000, 1100,
                                                         1200]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

```
rf_random.best_params_
```

```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

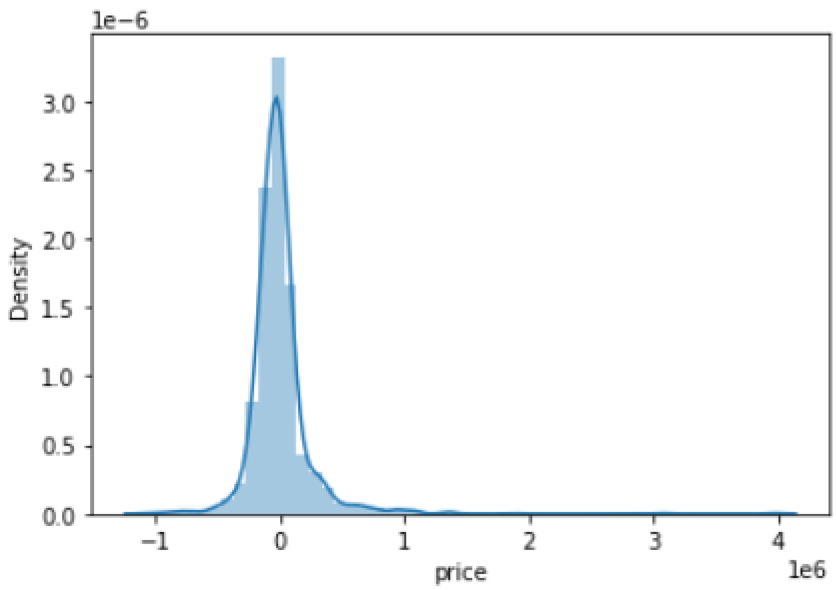
```
rf_random.best_score_
```

```
-47133600203.525894
```

```
predictions=rf_random.predict(X_test)
```

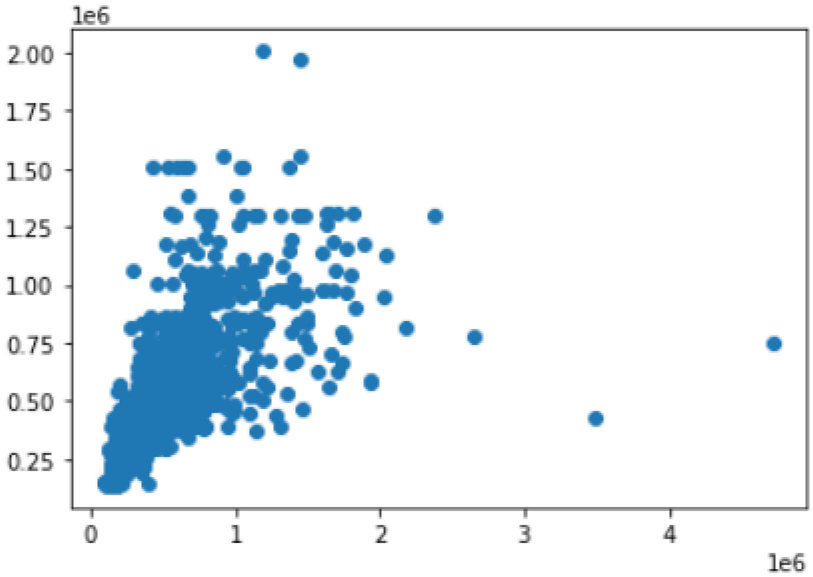
```
sns.distplot(y_test-predictions)
```

<AxesSubplot:xlabel='price', ylabel='Density'>



```
plt.scatter(y_test,predictions)
```

<matplotlib.collections.PathCollection at 0xbebc6d0>



```python
from sklearn import metrics
```

```python
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 136862.197740179
MSE: 59798179809.10714
RMSE: 244536.66352738833
```

Saving the model

```python
import pickle
# open a file, where you ant to store the data
file = open('random_forest_regression_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file)
```

```python
model = open('random_forest_regression_model.pkl','rb')
forest = pickle.load(model)
```

CONCLUSION

```python
model = open('random_forest_regression_model.pkl','rb')
forest = pickle.load(model)
```

```python
y_prediction = forest.predict(X_test)
```

```python
metrics.r2_score(y_test, y_prediction)
```

```
0.4345259609817921
```

```python
conclusion=pd.DataFrame([forest.predict(X_test)

[:],y_prediction[:]],index=["Predicted","Orginal"])
```

```python
conclusion
```

	0	1	2	3	4	5	6	7	8
Predicted	315393.561694	732365.188862	439660.318199	729860.045713	439660.318199	417508.172323	490656.089363	866175.442155	417508.172323	518682.55
Orginal	315393.561694	732365.188862	439660.318199	729860.045713	439660.318199	417508.172323	490656.089363	866175.442155	417508.172323	518682.55

· Interpretation of the Results Summary:

First the Car data is read and analyzed reading the features we analyze ,Price is the target column here. All the features are then analyzed, missing data

handling, outlier detection, data cleaning are done. Trend of SalePrice is observed for change in individual features. New features are extracted, redundant features dropped and categorical features are encoded accordingly.
I take a look at the first few rows using the data dictionary provided.I get a sense of what each column represents,I identify the predictors and response variables.I check if there is any unique identifier for each column.
made a note of all the observations as a part of data understanding,in the data there are 5984 variables out of which price is the responsible variable and all the predictors.
Name of the car and model is the unique identifier for each record as a part of data preparation. I start with data cleaning,i start with null values and data types of each column,i bring all the string values together.
Creating dummy variables increases the number of features greatly, highly imbalanced columns are dropped.
created dummy variables to convert categorical variables to numeric variables.

We need this because linear regression can only be done with numeric variables.
Before modelling divided the model into training and testing data.
70% random samples for training data and remaining 30% for testing data.
I began data modelling by creating the first model as Linear regression with all the variables of training data and noted the results of the first model.
Then calculated the Original price and Predicted price from the values observed.
Plotted the graph of feature importances for better visualization and the residual of the final to ensure that normal distribution.
Using the Regressor models calculated the KNeighborsRegressor,SVR,DecisionTreeRegressor, RandomForestRegressor, and observed that the RandomForestRegressor is performing the good precision and calculated the Hyper parameter tuning for RandomizedSearchCV.Finally calculated the rf_random best score.
Plotted the distplot and scatter plot of Y test predictions,and saved the model.