Car Price Detection

Submitted by:

NITIN SINGH TATRARI

Batch: 1826

ACKNOWLEDGMENT

.

I have studied on the business prospect of used car sector in India. This sector wills growth at a rapid rate in coming years. I have read articles online on aftermath of pandemic on this sector. I have read article and blog on factors upon which the parameter which affect the pricing of used car. I have visited different used car selling websites like OLX, cardekho and cars24. I have also studied different models from Kaggle on used car prediction model.

INTRODUCTION

- ## Business Problem Framing

  Since the pandemic and economic slowdown, trend of buying used car has accelerated. Although sales of used car have overtook those of new cars some years ago, but current economic scenario have further boast the trend that there demand-supply gap in the market.

  Due to this demand-supply gap, our client is facing problems with their previous car price valuation machine learning models. Even with the launch of many new companies in India and various new models, it's hard to evaluate the price.

- ## Conceptual Background of the Domain Problem

  A study conducted by CarDekho said that the used car segment bounced back up to 99% in the aftermath of the lockdowns, as opposed to the new car segment which recovered only up to 77%. Social distancing measures and need for isolation in the pandemic-frenzied world has been a convenient motivation for people to invest in lower-priced cars in place of availing ride-hailing alternatives that compel shared mobility.

  The price of car depends on its actual condition, which will be a challenge to have realistic expectations and set an appropriate price. Earlier people try to sell their cars to family or friend or someone known, which made it difficult to find the actual real sale value. But, now selling used car have become more professional thing, with many websites providing these services, then the correct evaluations of the car is very important for its selling.

- ## Review of Literature

  The prices of used car depends on many parameters such as brand, model, variant, manufacturing year, driven kilometres, number of owners, location, fuel type, transmission type and many other parameters. Since many states have their own rule on the road licence of a car, thus the prices is directly depend on year of manufacturing. Many states also follow stringent paperwork and authentication process, that the selling of used car is a challenge there, which affect the price. Fuel affects the running cost of vehicles, as diesel is cheaper than petrol, the cost per Kilometre decide the price. The driven kilometre tells the story of depreciation of the vehicle. Since depreciation cost is deducted every year, which in return results in the final sale price of the car. Even there is a perception that with the increase of no. of owners, the value of car reduces which have to be considering in pricing the car. One drawback is considering the real expected condition of the car, as a small deviation could affect the price. Thus, verified car plays a key role in pricing as people are willing to pay more for a verified one.

- ## Motivation for the Problem Undertaken

  Used car segment is one of the growing sector of automobile, even it have more growth rate the new car sector. It is estimated to sales 8.2 million cars in FY25, double than new cars. Almost half of all used car sales will come from organized dealers in FY25. US and UK had a ratio of used car to new car sales of 2.8:1 and 4.1:1, respectively, in 2020 and India had a ratio of 1.5:1. Thus, a large potential of growth is available in India where organized dealer will play an important role in it. Today, the organized sector only accounts for 17% of the market. Thus machine learning will play a vital role in aiding in price prediction and making strategy of used car sale for the growing organized sector.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

1) Inter quartile Range(IQR)
   We apply IQR on continuous variable – YEAR, KM_Travelled & PRICE, thus to remove outliers.

2) The correlation of both continuous variables was studied with respect to Target variable. YEAR shows a correlation of 64% and KM_Travelled shows a correlation of 15% with the target variable.

3) We replace the values in BRAND & MODEL with the number of its frequency count. As it will give more weightage to the value which has more frequent count.

```
df_frequency_model=df.MODEL.value_counts().to_dict()
```

```
df.MODEL=df.MODEL.map(df_frequency_model)
```

```
df_frequency_brand=df.BRAND.value_counts().to_dict()
```

```
df.BRAND=df.BRAND.map(df_frequency_brand)
```

4) We have changed all values in categorical variables into numerical as show below:-

```
from sklearn import preprocessing
```

```
le=preprocessing.LabelEncoder()
```

```
for i in ['FUEL','TRANSMISSION','No._of_owners']:
    df[i]=le.fit_transform(df[i])
```

```
df.head()
```

	BRAND	MODEL	YEAR	FUEL	TRANSMISSION	KM_Travelled	No._of_owners	PRICE
0	689	125	2016.0	3	1	57000.0	0	420000.0
1	1446	37	2016.0	3	0	11956.0	0	775000.0
3	1446	32	2016.0	3	1	32000.0	1	347000.0
4	358	82	2015.0	1	1	72000.0	1	500000.0
5	131	33	2017.0	1	0	148000.0	1	160000.0

5) We have standardised the independent variables as show below:-

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
x=sc.fit_transform(x)
```

- Data Sources and their formats

1) Data Sourcing

The data have been extracted from ‘https: //www.olx.in/' using Selenium. The steps followed are as follows:-

a) Importing Selenium and connecting the web driver to open the websites

```
import selenium
from selenium import webdriver
import time
from selenium.common.exceptions import NoSuchElementException,TimeoutException
```

```
#First we will connect to webdriver
driver=webdriver.Chrome(r'C:\Users\Nitin Singh Tatrari\Downloads\chromedriver_win32\chromedriver.exe')
```

```
#Open the webpage with webdriver
driver.get('https://www.olx.in/')
```

b) Loading the pages and extracting the Urls

```
#Clicking at 'Maharashtra'
click2=driver.find_element_by_xpath("//ul[@class='_3eZBP']/li[1]").click()
```

```
#Laoding data
for i in range(0,50):
    try:
        click1=driver.find_element_by_xpath("//button[@class='rui-39-wj rui-3evoE rui-1JPTg']").click()
        time.sleep(5)
    except  NoSuchElementException:
        print("Cannot load more")
        break
```

```
Cannot load more
```

```
#Storing URLs
URL=[]
```

```
#Extracting the URLs
urls=driver.find_elements_by_xpath("//li[@class='EIR5N']/a")
for url in urls:
    URL.append(url.get_attribute('href'))
```

Here, we have extracted the data of the state of ‘Maharashtra’. Similarly we perform for other state till we can scrap more than 5000 datas.

c) Scrapping the data from the Urls and storing it in respective lists

```
# Extracting data from all the URls.
for url in URL[0:5360]:
    try:
        driver.get(url)
        time.sleep(2)
    except TimeoutException:
        print('TimeOut Exception')
    try:
        brand=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[1]/div/span[2]")
        BRAND.append(brand.text)
    except NoSuchElementException:
        BRAND.append('-')
    try:
        model=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[2]/div/span[2]")
        MODEL.append(model.text)
    except NoSuchElementException:
        MODEL.append('-')
    try:
        vari=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[3]/div/span[2]")
        VARI.append(vari.text)
    except NoSuchElementException:
        VARI.append('-')
    try:
        yr=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[4]/div/span[2]")
        YR.append(yr.text)
    except NoSuchElementException:
        YR.append('-')
```

```
    try:
        fuel=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[5]/div/span[2]")
        FUEL.append(fuel.text)
    except NoSuchElementException:
        FUEL.append('-')
    try:
        trans=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[6]/div/span[2]")
        TRANS.append(trans.text)
    except NoSuchElementException:
        TRANS.append('-')
    try:
        km=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[7]/div/span[2]")
        KM.append(km.text)
    except NoSuchElementException:
        KM.append('-')
    try:
        noo=driver.find_element_by_xpath("//div[@class='_3JPEe']/div[8]/div/span[2]")
        NOO.append(noo.text)
    except NoSuchElementException:
        NOO.append('-')
    try:
        loc=driver.find_element_by_xpath("//span[@class='_2FRXm']")
        LOC.append(loc.text)
    except NoSuchElementException:
        LOC.append('-')
    try:
        price=driver.find_element_by_xpath("//span[@class='_2xKfz']")
        PRICE.append(price.text)
    except NoSuchElementException:
        PRICE.append('-')
```

d) Loading the data into Data Frame, reshuffling it and saving it into ‘csv’ file.

```
import pandas as pd
```

```
car=pd.DataFrame()
```

```
car['BRAND']=BRAND[:]
car['MODEL']=MODEL[:]
car['VARIANT']=VARI[:]
car['YEAR']=YR[:]
car['FUEL']=FUEL[:]
car['TRANSMISSION']=TRANS[:]
car['KM_Travelled']=KM[:]
car['No._of_owners']=NOO[:]
car['LOCATION']=LOC[:]
car['PRICE']=PRICE[:]
```

```
#Shuffling the data in the data frame
car_shuffled=car.sample(frac=1).reset_index(drop=True)
```

```
car_shuffled.to_csv('cars.csv')
```

2) Data loading and format

```
#Loading data set
data=pd.read_csv('cars.csv')
data
```

```
# Uploading data in Data frame
df=pd.DataFrame(data)
df.head()
```

	Unnamed: 0	BRAND	MODEL	VARIANT	YEAR	FUEL	TRANSMISSION	KM_Travelled	No._of_owners	LOCATION	PRICE
0	0	Hyundai	i10	1.2 Kappa Magna	2016	Petrol	Manual	57,000 km	1st	Pandiyan, Madurai, Tamil Nadu	₹ 4,20,000
1	1	Maruti Suzuki	Ciaz	2014-2017 AT ZXi	2016	Petrol	Automatic	11,956 km	1st	Mulund East, Mumbai, Maharashtra	₹ 7,75,000
2	2	Audi	Q5	2.0 TDI quattro	2017	Diesel	Automatic	63,000 km	2nd	Pykara, Madurai, Tamil Nadu	₹ 28,50,000
3	3	Maruti Suzuki	Celerio	2014-2017 VXI	2016	Petrol	Manual	32,000 km	2nd	Gokuldham, Mumbai, Maharashtra	₹ 3,47,000
4	4	Mahindra	Bolero	ZLX	2015	Diesel	Manual	72,000 km	2nd	Krishna Nagar, Saharanpur, Uttar Pradesh	₹ 5,00,000

a) The data have 11 columns and 5360 rows.
b) Data is loaded into Data Frame ‘df’

```
#Checking data type
df.dtypes
```

```
Unnamed: 0          int64
BRAND              object
MODEL              object
VARIANT            object
YEAR               object
FUEL               object
TRANSMISSION       object
KM_Travelled       object
No._of_owners      object
LOCATION           object
PRICE              object
dtype: object
```

a) ‘Unnamed: 0’ variable is integer type and all other variable is string type.

- Data Pre-processing Done

1) Checking unique values, its counts and null value.

```
#Checking unique values, value counts & null values
for i in df.columns:
    print('\n')
    print(i)
    print(df[i].unique())
    print('The Total no. of unique value:',len(df[i].unique()))
    print(df[i].value_counts())
    print(df[i].isnull().sum())
```

The following cleaning process was done after studying above results.

a) The data have no Nan values.

b) Since ‘Unmaed: 0’ variable have all unique values as it is just an index number, we can drop it.

```
# Dropping 'Unnamed: 0' as it show index value
df.drop('Unnamed: 0',axis=1,inplace=True)
df.head()
```

c) In ‘FUEL’ variable, we will keep the rows having the values -: 'Petrol','Diesel','CNG & Hybrids','LPG','Electric' and drop all other rows having different values.

```
# Keeping correct data
df=df[df['FUEL'].isin(['Petrol','Diesel','CNG & Hybrids','LPG','Electric'])]
```

d) In ‘TRANSMISSION’ variable, we will keep the rows having the values -: 'Manual','Automatic' and drop all other rows having different values.

```
#  Keeping correct data
df = df[df['TRANSMISSION'].isin(['Manual','Automatic'])]
```

e) Removing rows having ‘-‘as value.

```
#Removing '-' from the data frame
for i in df.columns:
    df = df[df[i] != '-']
```

f) Renaming few values in ‘YEAR’ variable to correct it.

```
# Renaming data
df['YEAR']=df['YEAR'].replace({'2011.0':'2011','2015.0':'2015','2012.0':'2012','2014.0':'2014'})
```

g) In ‘BRAND’ variable, we keep only those values which have frequency greater than 20.

```
s=df.BRAND.value_counts().gt(20)
df=df.loc[df.BRAND.isin(s[s].index)]
```

h) In ‘MODEL’ variable, we keep those values which have frequency greater than 10.

```
s=df.MODEL.value_counts().gt(10)
df=df.loc[df.MODEL.isin(s[s].index)]
```

i) Since ‘VARAINT’ variable have 1650 unique values, we have to drop it.

```
df.drop('VARIANT', axis=1, inplace=True)
```

j) Removing '₹' and ',' from ‘PRICE’ variable.

```
df['PRICE'] = df['PRICE'].str.replace('₹','').str.replace(',','')
```

k) Removing ',' and 'km' from 'KM_Travelled' variable.

```
df['KM_Travelled'] = df['KM_Travelled'].str.replace(',','').str.replace('km','')
```

l) Splitting the 'LOCATION' variable and keep only state name.

```
df2=df['LOCATION'].str.split(',',n=2,expand=True)
```

```
df2.head()
```

	0	1	2
0	Pandiyan	Madurai	Tamil Nadu
1	Mulund East	Mumbai	Maharashtra
3	Gokuldham	Mumbai	Maharashtra
4	Krishna Nagar	Saharanpur	Uttar Pradesh
5	Jal Mahal	Jaipur	Rajasthan

```
df2.drop({0,1},axis=1,inplace=True)
```

```
df2.head()
```

	STATE
0	Tamil Nadu
1	Maharashtra
3	Maharashtra
4	Uttar Pradesh
5	Rajasthan

```
df2.rename(columns={2:'STATE'},inplace=True)
```

m) Removing Outlier by IQR method

```
for i in ['YEAR','KM_Travelled','PRICE']:
    Q1 = df[i].quantile(0.25)
    Q3 = df[i].quantile(0.75)
    IQR = Q3 - Q1
    df = df[((df[i] > (Q1 - 1.5 * IQR)) & (df[i] < (Q3 + 1.5 * IQR)))]
```

- Data Inputs- Logic- Output Relationships

1) Data before and after outlier removal

```
1 df.describe()
```

	YEAR	KM_Travelled	PRICE
count	4175.000000	4175.000000	4.175000e+03
mean	2012.856287	74660.381317	4.990036e+05
std	4.638276	60847.153169	4.773023e+05
min	1979.000000	0.000000	1.500000e+04
25%	2010.000000	45000.000000	2.350000e+05
50%	2013.000000	69000.000000	3.850000e+05
75%	2016.000000	91000.000000	6.250000e+05
max	2021.000000	985000.000000	6.500000e+06

Before Outliers removal

```
1 df.describe()
```

	YEAR	KM_Travelled	PRICE
count	3586.000000	3586.000000	3.586000e+03
mean	2013.233129	66716.029559	4.210766e+05
std	3.909189	34693.331401	2.409952e+05
min	2002.000000	0.000000	1.500000e+04
25%	2011.000000	44000.000000	2.350000e+05
50%	2013.500000	67000.000000	3.750000e+05
75%	2016.000000	88000.000000	5.750000e+05
max	2021.000000	159979.000000	1.200000e+06

After Outliers removal

a) Minimum value in 'YEAR' changed from 1979 to 2002.
b) Maximum value in 'KM_Travelled' changed from 985000 to 159979.
c) Maximum value in 'KM_Travelled' changed from 6500000 to 1200000.

2) Studying Correlation with Target variable

```
1 df.corr()
```

	YEAR	KM_Travelled	PRICE
YEAR	1.000000	-0.235009	0.426681
KM_Travelled	-0.235009	1.000000	-0.086782
PRICE	0.426681	-0.086782	1.000000

Before Outlier removal

```
1 df.corr()
```

	YEAR	KM_Travelled	PRICE
YEAR	1.000000	-0.418953	0.642625
KM_Travelled	-0.418953	1.000000	-0.150110
PRICE	0.642625	-0.150110	1.000000

After Outlier removal

a) Correlation of YEAR with PRICE improves from 42.68% to 64.26%.
b) Correlation of KM_Travlled improves from 8.67% to 15%

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

The assumptions considered from the project are:-

1) The data was scrapped state wise. The states were Maharashtra, Karnataka, Rajasthan, Kerala, Tamil Nadu and Uttar Pradesh.
2) Cars with Electricity as FUEL was not consider due to low frequency count of it.
3) Brand and model of car is selected on frequency count basic. Brand with more than 20 frequency count and model with more than 10 frequency count is only considered.
4) Car Variant is not considered in the model.

- Hardware and Software Requirements and Tools Used

The libraries used are: pandas, numpy, matplotlib.pyplot, seaborn and scikit_learn and Selenium. The laptop used is with Intel I5 10^{th} generation, 4GB RAM, 4GB GPU.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

1) I have cleaned the data by dropping all empty and wrong scrapped data and dropping all columns which have huge percentage of unique values('Unnamed: 0 & Variant). I have also dropped those values which have very less frequency count (Electric in FUEL).
2) I have graphically analysis the data and thus removed STATE as it shows no significance on target variable.
3) I have replaced BRAND AND MODEL name with their frequency count respectively. I have kept values greater than 20 in BRAND and greater than 10 in MODEL.
4) I have categorically encoded FUEL, TRANSMISSION & No._of_owners.
5) I have standardised the data with Standard Scaler function.

- Testing of Identified Approaches (Algorithms)

The algorithms used for training and testing are

a) Linear Regressor
b) Lasso Regressor
c) Decision Tree Regressor
d) Support Vector Regressor
e) K-Nearest Neighbour Regressor
f) Random Forest Regressor

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

```
from sklearn.ensemble import RandomForestRegressor
```

- Run and Evaluate selected models

A) MODEL SELECTION

1) Linear Regression

```
lg=LinearRegression()
lg.fit(x_train,y_train)
pred=lg.predict(x_test)
print("R2_score:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print('Root_mean_square_error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_score: 0.49202342630882
mean_squared_error: 26708027041.050014
mean_absolute_error: 125560.6718635326
Root_mean_square_error: 163425.90688458795
```

2) Decision Tree Regression

```
dtr=DecisionTreeRegressor()
dtr.fit(x_train,y_train)
pred=dtr.predict(x_test)
print("R2_score:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print('Root_mean_square_error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_score: 0.6601557988138298
mean_squared_error: 17868084051.73089
mean_absolute_error: 87468.71355617455
Root_mean_square_error: 133671.55288890336
```

3) Lasso Regression

```
ls=Lasso(alpha=0.0001)
ls.fit(x_train,y_train)
```

```
Lasso(alpha=0.0001)
```

```
ls.score(x_train,y_train)
```

```
0.4797951387839703
```

```
lg.score(x_test,y_test)
```

```
0.49202342630882
```

4) Support Vector Regression

```
svr=SVR()
svr.fit(x_train,y_train)
pred=svr.predict(x_test)
print("R2_score:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print('Root_mean_square_error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_score: -0.03303022031272751
mean_squared_error: 54313920143.70215
mean_absolute_error: 186423.8129421633
Root_mean_square_error: 233053.4705678123
```

5) K-Nearest Kneighbors regression

```
kn=KNeighborsRegressor()
kn.fit(x_train,y_train)
pred=kn.predict(x_test)
print("R2_score:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print('Root_mean_square_error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_score: 0.5729362194913042
mean_squared_error: 22453852379.841225
mean_absolute_error: 102670.68746518106
Root_mean_square_error: 149846.09564430173
```

B) ENSEMBLE TECHNIQUE

```
from sklearn.ensemble import RandomForestRegressor
```

```
rfr=KNeighborsRegressor()
rfr.fit(x_train,y_train)
pred=rfr.predict(x_test)
print("R2_score:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print('Root_mean_square_error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_score: 0.5729362194913042
mean_squared_error: 22453852379.841225
mean_absolute_error: 102670.68746518106
Root_mean_square_error: 149846.09564430173
```

C) HYPERPARAMETER TUNNING

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

```
crit={'criterion':['mse','friedman_mse','mae','poisson']}
#alphavalue={'alpha':[1,0.1,0.01,0.001,0.001,0.0001,0]}
grid=GridSearchCV(estimator=dtr,param_grid=crit)
grid.fit(x,y)
print(grid)
print(grid.best_score_)
print(grid.best_estimator_.criterion)
print(grid.best_params_)
```

```
GridSearchCV(estimator=DecisionTreeRegressor(),
             param_grid={'criterion': ['mse', 'friedman_mse', 'mae',
                                       'poisson']})
0.6304173680203665
mse
{'criterion': 'mse'}
```

- Key Metrics for success in solving problem under consideration

  The Key metrics used are:-

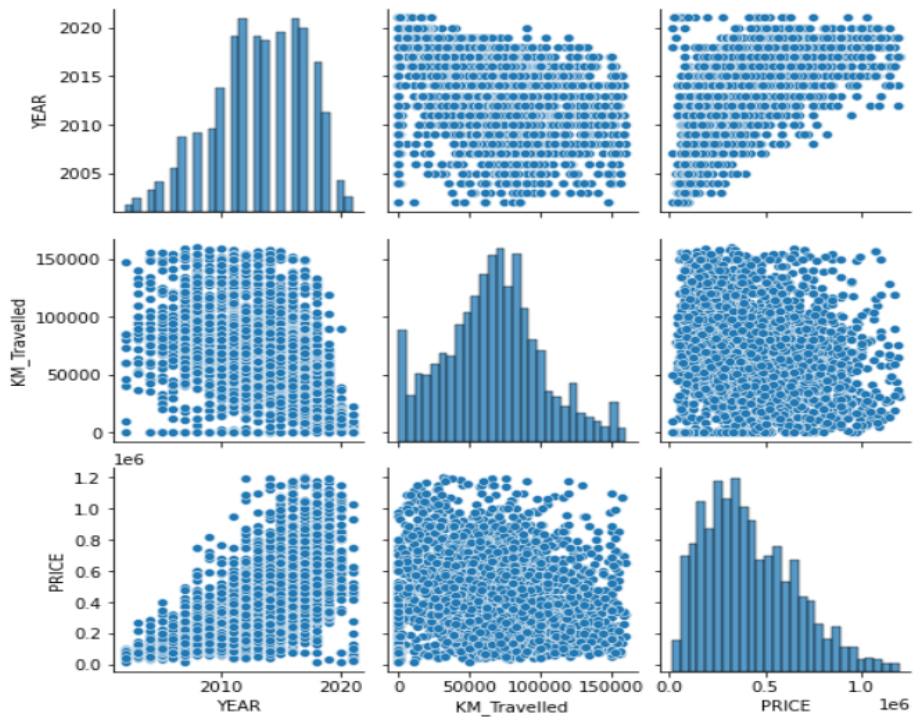
  1) R2 Score: It gives the variance of predicted value from the original value.
  2) Mean Absolute error: It gives the mean difference of predicted value from the original value
  3) Mean Square error: It gives the squared difference of predicted value from the original value.
  4) Root mean square error: It the square roots of mean square error. Thus gives the difference of predicted value from the original value.

- Visualizations

  1) Pair plot:

```
1 sns.pairplot(df)
```

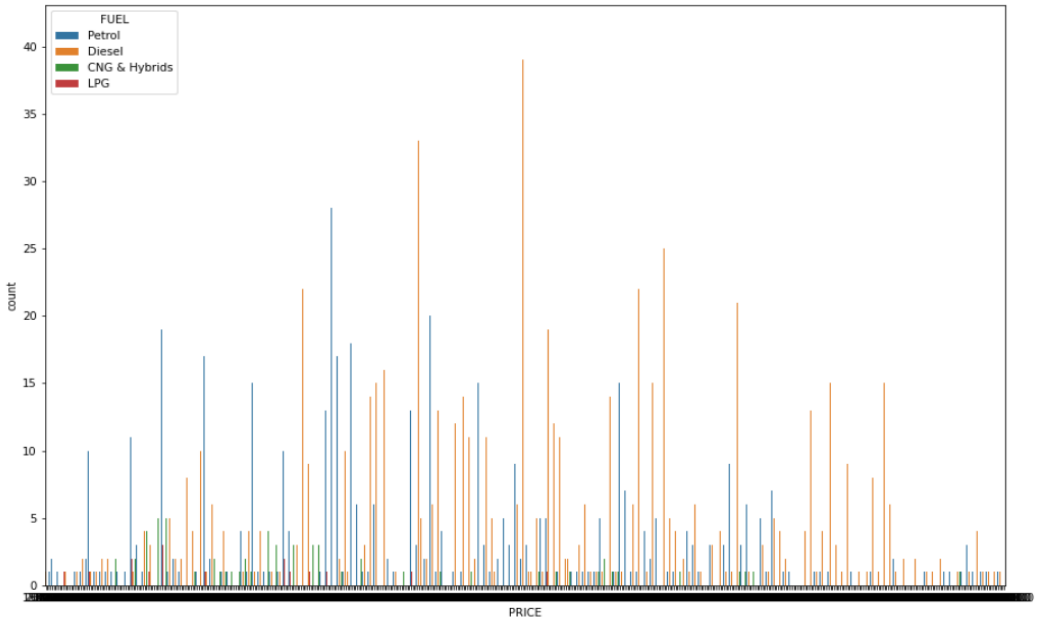
```
<seaborn.axisgrid.PairGrid at 0x147b098de20>
```



Here, we can observe KM_Travelled and YEAR does not able to form linear relation with Target variable PRICE.

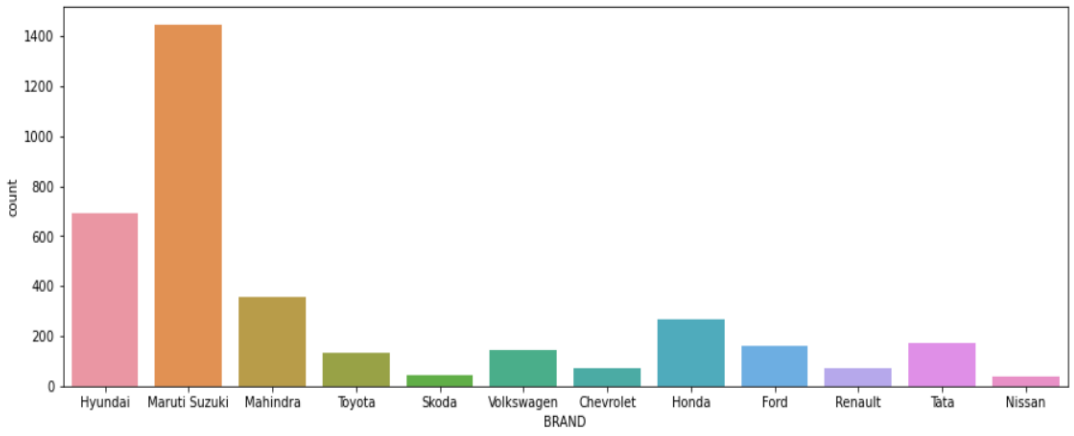
2) Count plot

```
plt.figure(figsize=(15,10))
sns.countplot(x='PRICE',hue='FUEL',data=df)
plt.show()
```



Here we can observe that Diesel run used car frequency count increase as compare to petrol run used car with increase in price

```
plt.figure(figsize=(15,5))
sns.countplot(df['BRAND'])
plt.show()
```



Here, we can observe Maruti Suzuki Brand car are more available for resale in market, followed by Hyundai.

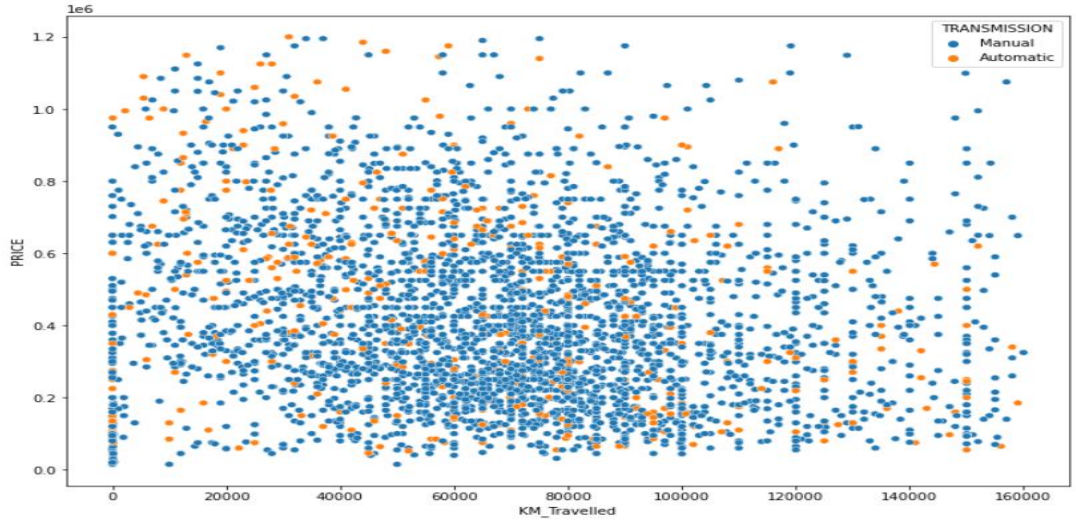
3) Scatter plot

```
plt.figure(figsize=(12,9))
sns.scatterplot(x='KM_Travelled',y='PRICE',hue='FUEL',data=df)
plt.show()
```

Here we have observed:

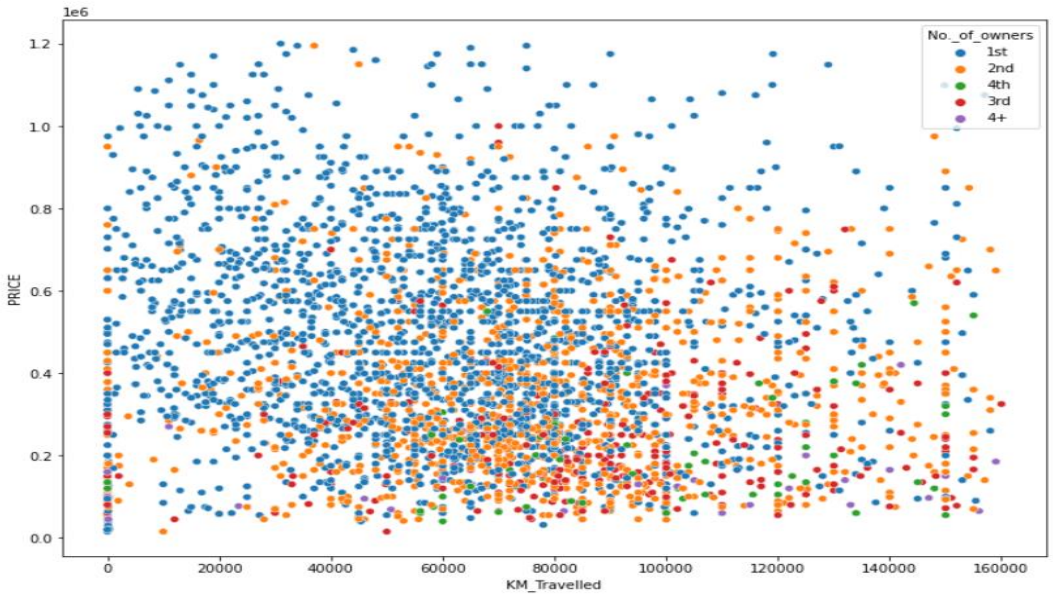
a) Diesel run car have travelled more kilometres than petrol run ones
b) High priced and more kilometres travelled car are mostly diesel run only.
c) High priced car density decreases with increase in kilometres travelled.

```
plt.figure(figsize=(12,9))
sns.scatterplot(x='KM_Travelled',y='PRICE',hue='TRANSMISSION',data=df)
plt.show()
```



We observe cars with Automatic transmission show a faint inverse linear relation with price and kilometre travelled

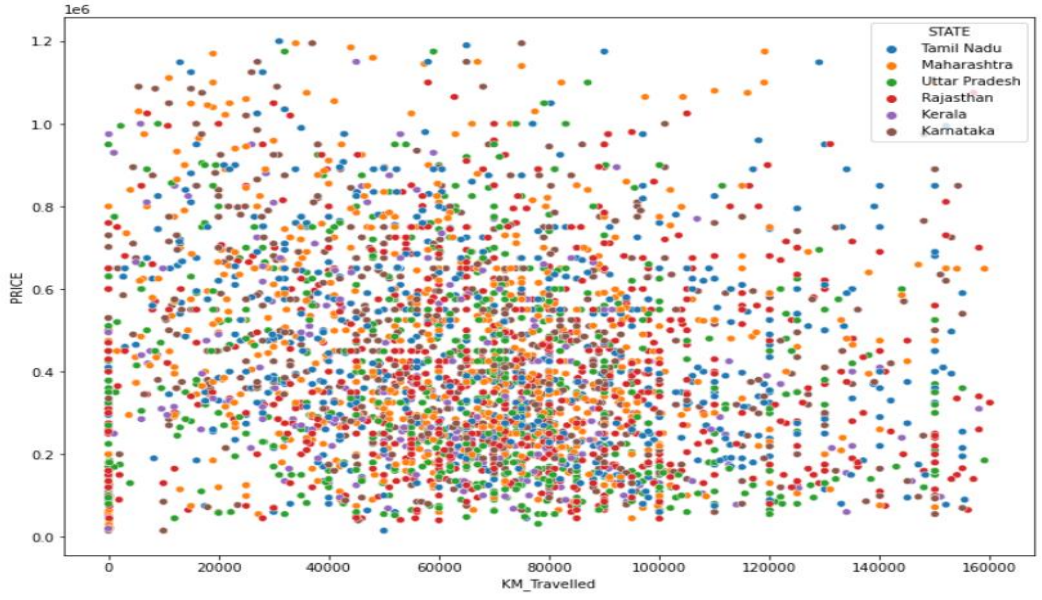
```
plt.figure(figsize=(12,9))
sns.scatterplot(x='KM_Travelled',y='PRICE',hue='No._of_owners',data=df)
plt.show()
```



Here, we observer that:-

a) With increase of no. of owners, the distance travelled also increase .Thus; they have linear relationship between them.
b) First owned used car are at higher price than others.

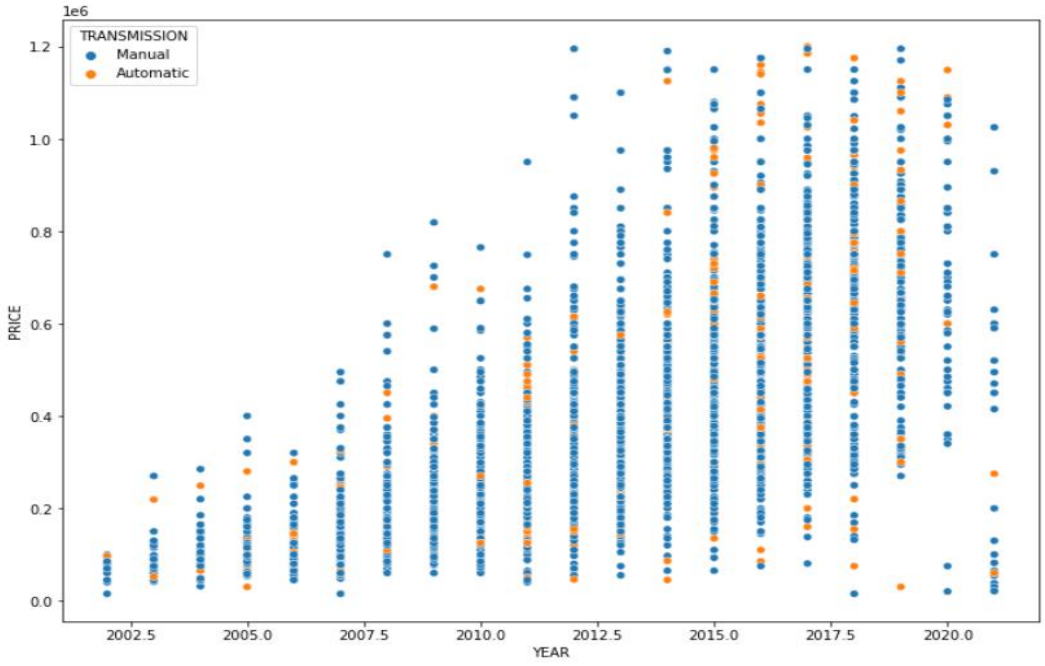
```
plt.figure(figsize=(12,9))
sns.scatterplot(x='KM_Travelled',y='PRICE',hue='STATE',data=df)
plt.show()
```



STATE shows no relation with price of car, thus we can drop the column.

```
df.drop('STATE',axis=1,inplace=True)
```

```
plt.figure(figsize=(12,9))
sns.scatterplot(x='YEAR',y='PRICE',hue='TRANSMISSION',data=df)
plt.show()
```



YEAR shows a slight linear relation with PRICE.

```
plt.figure(figsize=(12,9))
sns.scatterplot(x='YEAR',y='PRICE',hue='No._of_owners',data=df)
plt.show()
```

Here we observe that from 2015, comparatively more first owned used cars are available for resale. Thus we can interpret that people are not interested of selling/buying second owned used cars.

- Interpretation of the Results

1) Diesel run used car are costlier than petrol run used car
2) Maruti Suzuki used cars have higher percentage share in second hand cars market.
3) Diesel car is used for longer distance travel than petrol car.
4) Car prices decrease with increase of no. of users.
5) Older cars have comparatively lesser resale value than newer car.
6) After 2015, there is trend of buying not more than first owned car.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  Describe the key findings, inferences, observations from the whole problem.

  1) Maruti Suzuki is the most available used car for resale in OLX.
  2) Maximum used car run on petrol & diesel only.
  3) Car with Manual transmission are five times more than car with automated transmission.
  4) From 2015, comparatively more first owned used cars are available for resale. Thus we can interpret that people are not interested of selling/buying second owned used cars.
  5) Cost of car does not depend on state.

- ## Learning Outcomes of the Study in respect of Data Science

  The best accuracy was given by Decision Tree regressor of R2 score 66%. Columns – FUEL, TRANSMISSION & No._of_owners have categorical type data. Even BRAND & YEAR have limited number of unique values (12 & 20 respectively). Only KM_Travelled is a proper continuous variable. Thus Decision Tree regressor will perform better.

  R2 score tells how good model is fitted to the regression line. Higher the value, better the model is fit. MAE, MSE & RMSE tell us how much the predicted value is deviated from actual value. Lesser the values, better the model.

- ## Limitations of this work and Scope for Future Work

  In the dataset, there were many car brands whose frequency was less. Thus it can impact the model in fitting. Same goes with model and variant. Variant was dropped as it may lead to over-fitting of model because it has more 1500 unique values.

The data was taken for few states only, thus if we could gather the data of almost all states, then we can get clearer picture in respect to state wise.

Here, we have replace value in BRAND & MODEL with frequency count of each value. Thus two values of same frequency count will come under same value. Thus it is one of the drawbacks of this approach.