FLIP ROBO

Car Price Prediction

Submitted by:

Bhumik P Shah

ACKNOWLEDGMENT

While preparing this case study I had gone through a couple of videos on You tube for reference purpose to understand in depth about the car prices fluctuating based on demand and supply.

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, there has been an uncertainty in the car market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases.

  ## Conceptual Background of the Domain Problem

  Determining whether the listed price of a used car is a challenging task, due to many factors that drive a used vehicle's price on the market. The prices of new cars in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But due to the increased price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase. It is important to know their actual market value while both buying and selling.

- ## Review of Literature

  Because I truly think that sharing sources and knowledges allow to help others but also ourselves, the sources of the project are available at the following link:

  https://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26612934.pdf

  ## Motivation for the Problem Undertaken

  Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a dataset, including its size, initial patterns in the data and other attributes.

  There are no null values present in any of the features in this dataset.

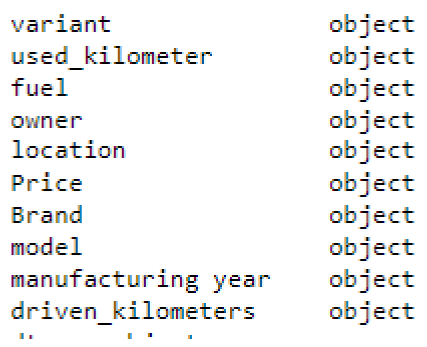
  The outliers are there in one of the feature named as driven kilometres and the same has been confirmed using boxplot and distribution plot. The outliers are treated using IQR method by replacing the same with the median of the respective feature. As all the numerical features i.e. independent features in this dataset are having skewness less than 0.5 there is no need to remove skewness from those features. The correlation of every numerical feature with label/target is checked and one of the feature driven kilometres is having very less correlation with the label/target. It can be dropped from the dataset but it is a wise decision not to drop the feature based on correlation as it will result into data loss. The scaled data is given to the model for learning and testing purpose

- Data Sources and their formats

  The data is collected from cardekho.com for different locations using web scrapping tool selenium.

  The dataset contains 5091 records (rows) and 10 features (columns) with car_ price feature being the target attribute.

  The details of 10 features(columns) are attached below:

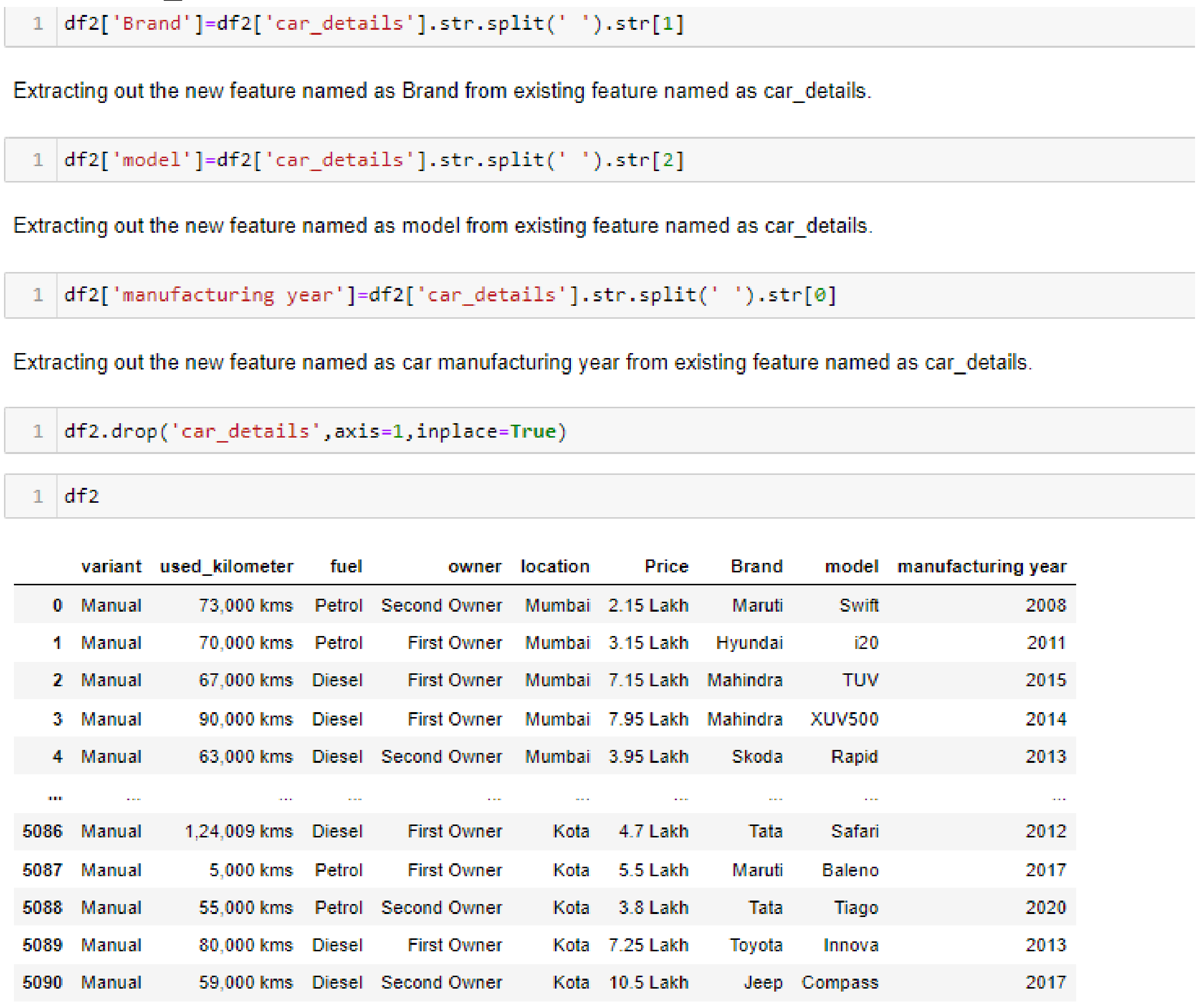


```
variant               object
used_kilometer        object
fuel                  object
owner                 object
location              object
Price                 object
Brand                 object
model                 object
manufacturing year    object
driven_kilometers     object
```

All the features in this dataset are of object datatype.

- ## Data Pre-processing Done

Extracting out new features named as Brand, model and manufacturing year from the existing feature car_details and finally dropping the feature car_details as all the useful information is extracted.



```
df2['Brand']=df2['car_details'].str.split(' ').str[1]
```

Extracting out the new feature named as Brand from existing feature named as car_details.

```
df2['model']=df2['car_details'].str.split(' ').str[2]
```

Extracting out the new feature named as model from existing feature named as car_details.

```
df2['manufacturing year']=df2['car_details'].str.split(' ').str[0]
```

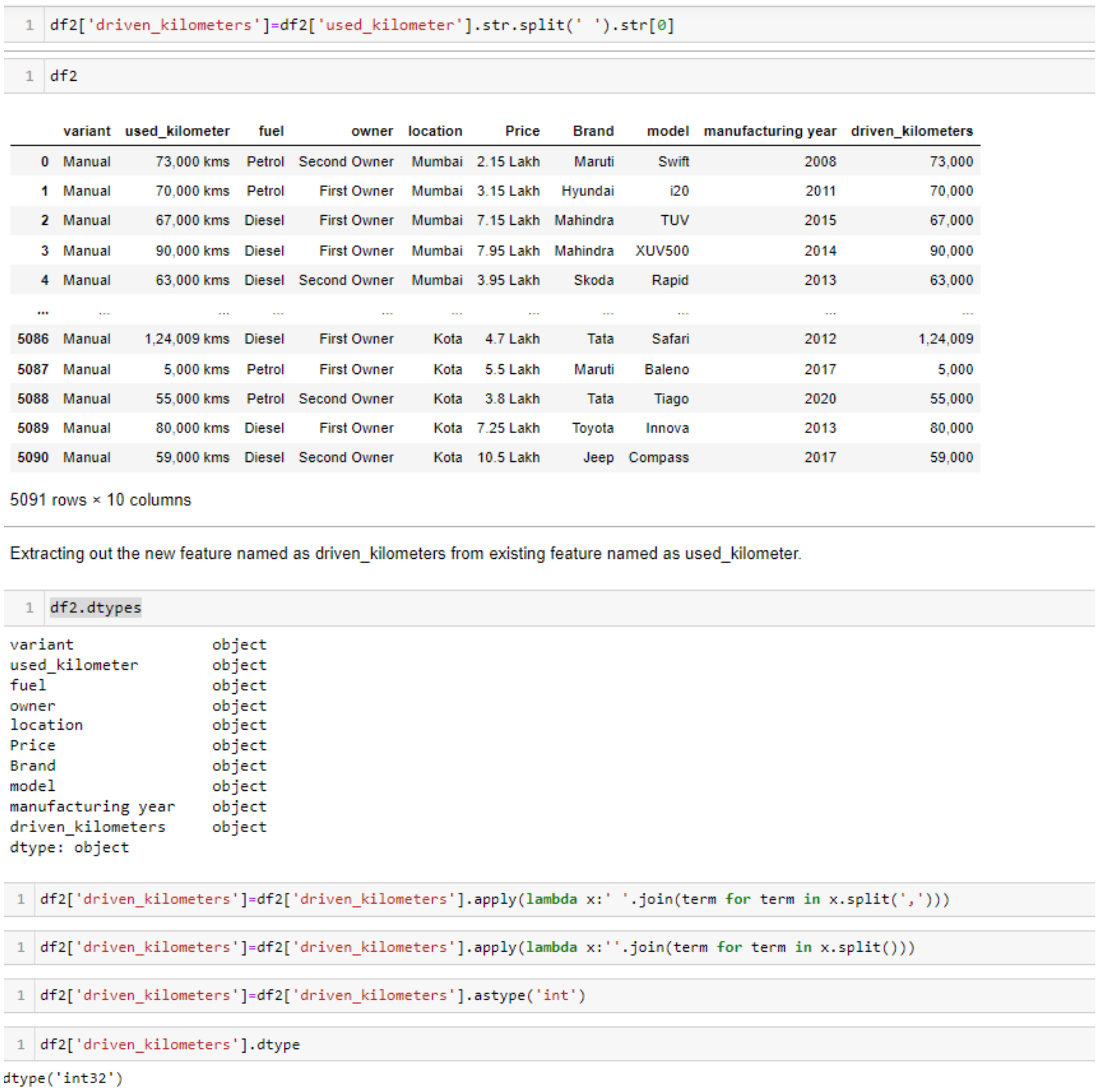
Extracting out the new feature named as car manufacturing year from existing feature named as car_details.

```
df2.drop('car_details',axis=1,inplace=True)
```

```
df2
```

	variant	used_kilometer	fuel	owner	location	Price	Brand	model	manufacturing year
0	Manual	73,000 kms	Petrol	Second Owner	Mumbai	2.15 Lakh	Maruti	Swift	2008
1	Manual	70,000 kms	Petrol	First Owner	Mumbai	3.15 Lakh	Hyundai	i20	2011
2	Manual	67,000 kms	Diesel	First Owner	Mumbai	7.15 Lakh	Mahindra	TUV	2015
3	Manual	90,000 kms	Diesel	First Owner	Mumbai	7.95 Lakh	Mahindra	XUV500	2014
4	Manual	63,000 kms	Diesel	Second Owner	Mumbai	3.95 Lakh	Skoda	Rapid	2013
...	...	...	...	...	...	...	...	...	...
5086	Manual	1,24,009 kms	Diesel	First Owner	Kota	4.7 Lakh	Tata	Safari	2012
5087	Manual	5,000 kms	Petrol	First Owner	Kota	5.5 Lakh	Maruti	Baleno	2017
5088	Manual	55,000 kms	Petrol	Second Owner	Kota	3.8 Lakh	Tata	Tiago	2020
5089	Manual	80,000 kms	Diesel	First Owner	Kota	7.25 Lakh	Toyota	Innova	2013
5090	Manual	59,000 kms	Diesel	Second Owner	Kota	10.5 Lakh	Jeep	Compass	2017

On the similar lines new feature driven_kilometer is extracted from used_kilometer . Further, the new feature driven_kilometer is having object datatype it is to be converted into float datatype as kilometer is always a numerical value.



```
df2['driven_kilometers']=df2['used_kilometer'].str.split(' ').str[0]
```

```
df2
```

	variant	used_kilometer	fuel	owner	location	Price	Brand	model	manufacturing year	driven_kilometers
0	Manual	73,000 kms	Petrol	Second Owner	Mumbai	2.15 Lakh	Maruti	Swift	2008	73,000
1	Manual	70,000 kms	Petrol	First Owner	Mumbai	3.15 Lakh	Hyundai	i20	2011	70,000
2	Manual	67,000 kms	Diesel	First Owner	Mumbai	7.15 Lakh	Mahindra	TUV	2015	67,000
3	Manual	90,000 kms	Diesel	First Owner	Mumbai	7.95 Lakh	Mahindra	XUV500	2014	90,000
4	Manual	63,000 kms	Diesel	Second Owner	Mumbai	3.95 Lakh	Skoda	Rapid	2013	63,000
...	...	...	...	...	...	...	...	...	...	...
5086	Manual	1,24,009 kms	Diesel	First Owner	Kota	4.7 Lakh	Tata	Safari	2012	1,24,009
5087	Manual	5,000 kms	Petrol	First Owner	Kota	5.5 Lakh	Maruti	Baleno	2017	5,000
5088	Manual	55,000 kms	Petrol	Second Owner	Kota	3.8 Lakh	Tata	Tiago	2020	55,000
5089	Manual	80,000 kms	Diesel	First Owner	Kota	7.25 Lakh	Toyota	Innova	2013	80,000
5090	Manual	59,000 kms	Diesel	Second Owner	Kota	10.5 Lakh	Jeep	Compass	2017	59,000

5091 rows × 10 columns

Extracting out the new feature named as driven_kilometers from existing feature named as used_kilometer.

```
df2.dtypes
```

```
variant               object
used_kilometer        object
fuel                  object
owner                 object
location              object
Price                 object
Brand                 object
model                 object
manufacturing year    object
driven_kilometers     object
dtype: object
```

```
df2['driven_kilometers']=df2['driven_kilometers'].apply(lambda x:' '.join(term for term in x.split(',')))
```

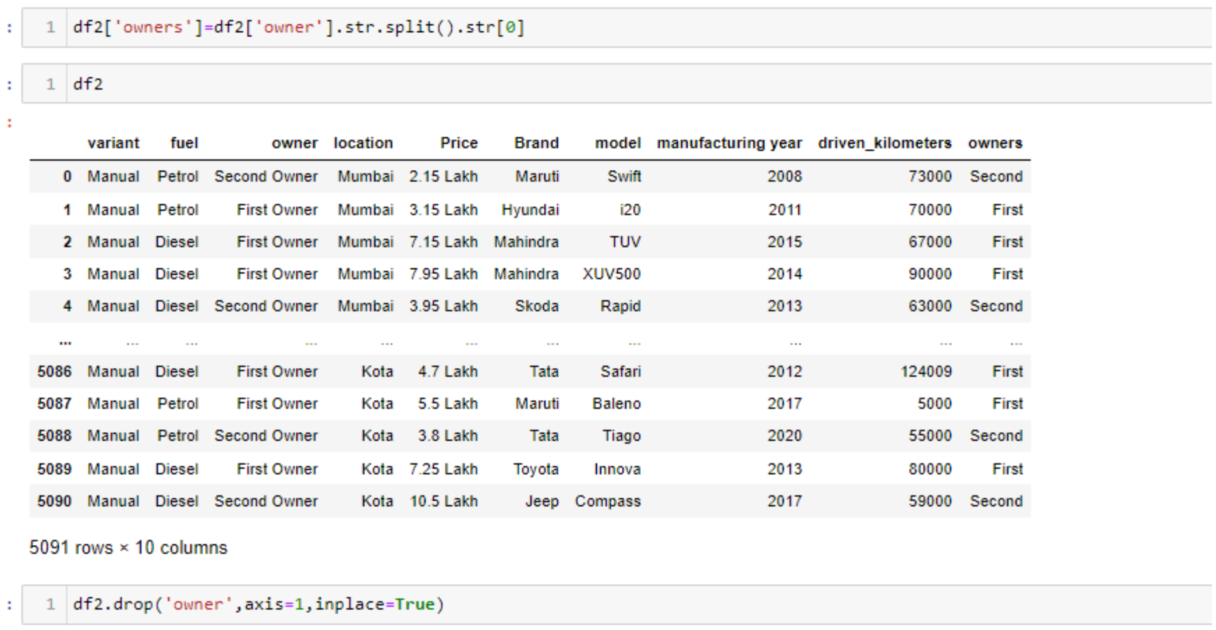
```
df2['driven_kilometers']=df2['driven_kilometers'].apply(lambda x:''.join(term for term in x.split()))
```

```
df2['driven_kilometers']=df2['driven_kilometers'].astype('int')
```

```
df2['driven_kilometers'].dtype
```

```
dtype('int32')
```

Similarly new features named as owners and car price are extracted from the existing feature owner and price and dropping the existing feature after extracting the useful information from it.



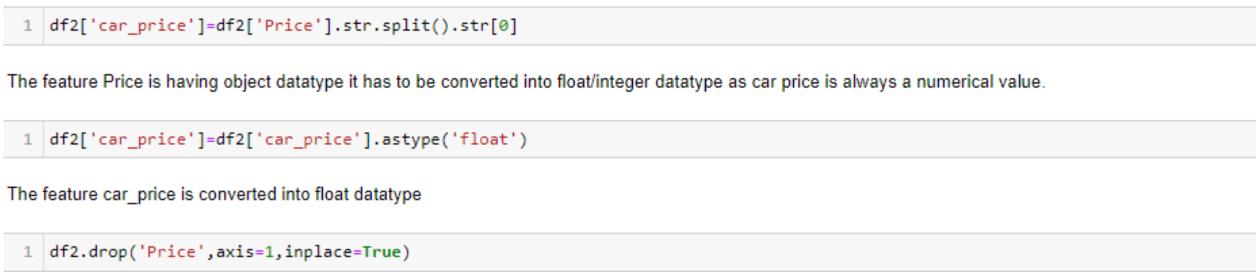
```python
df2['owners']=df2['owner'].str.split().str[0]
```

```python
df2
```

	variant	fuel	owner	location	Price	Brand	model	manufacturing year	driven_kilometers	owners
0	Manual	Petrol	Second Owner	Mumbai	2.15 Lakh	Maruti	Swift	2008	73000	Second
1	Manual	Petrol	First Owner	Mumbai	3.15 Lakh	Hyundai	i20	2011	70000	First
2	Manual	Diesel	First Owner	Mumbai	7.15 Lakh	Mahindra	TUV	2015	67000	First
3	Manual	Diesel	First Owner	Mumbai	7.95 Lakh	Mahindra	XUV500	2014	90000	First
4	Manual	Diesel	Second Owner	Mumbai	3.95 Lakh	Skoda	Rapid	2013	63000	Second
...	...	...	...	...	...	...	...	...	...	...
5086	Manual	Diesel	First Owner	Kota	4.7 Lakh	Tata	Safari	2012	124009	First
5087	Manual	Petrol	First Owner	Kota	5.5 Lakh	Maruti	Baleno	2017	5000	First
5088	Manual	Petrol	Second Owner	Kota	3.8 Lakh	Tata	Tiago	2020	55000	Second
5089	Manual	Diesel	First Owner	Kota	7.25 Lakh	Toyota	Innova	2013	80000	First
5090	Manual	Diesel	Second Owner	Kota	10.5 Lakh	Jeep	Compass	2017	59000	Second

5091 rows × 10 columns

```python
df2.drop('owner',axis=1,inplace=True)
```



```python
df2['car_price']=df2['Price'].str.split().str[0]
```

The feature Price is having object datatype it has to be converted into float/integer datatype as car price is always a numerical value.

```python
df2['car_price']=df2['car_price'].astype('float')
```

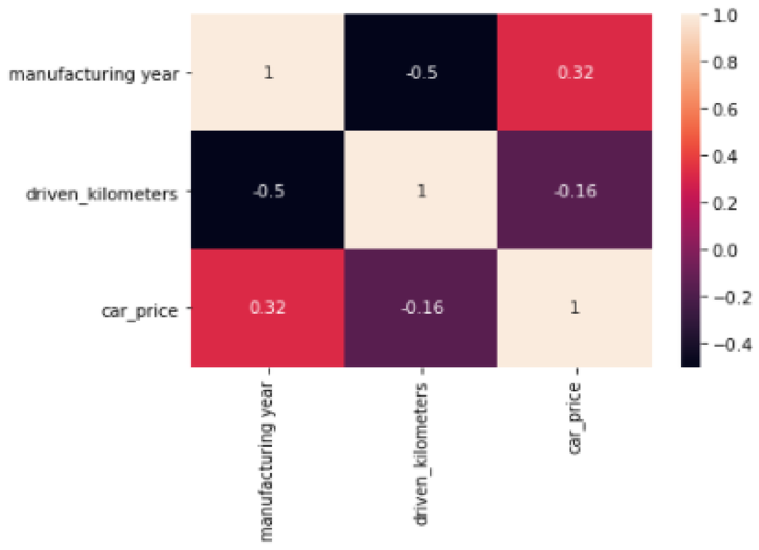
The feature car_price is converted into float datatype

```python
df2.drop('Price',axis=1,inplace=True)
```

The new feature car_price is having object datatype is converted into float datatype as a car price is always a numerical value.

- ## Data Inputs- Logic- Output Relationships

As per the correlation details attached below the target/output attribut e Car price is having a positive linear correlation with feature manufact uring year. The feature driven_kilometers is having very less correlatio n with the target attribute it can be dropped from the dataset but I am not dropping it as data is vital.



- Hardware and Software Requirements and Tools Used

  I have used Jupyter Notebook for writing python codes. Further, libraries such as Pandas, Seaborn, Matplotlib, SKlearn, NumPy are used.

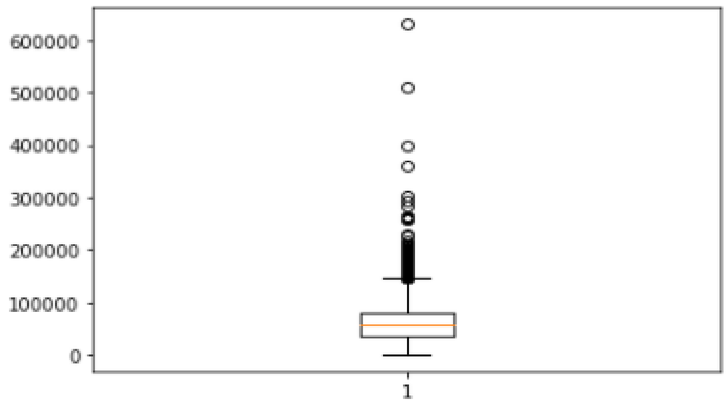
Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  The shape of the dataset, datatypes, null values present in the dataset or not are amongst few things that are first to be known. There are no null values present in this dataset. Further, next step is to go for outlier detection if any using box plot and distribution plot. The feature driven_kilometers is having outliers as confirmed from the boxplot.

```
plt.boxplot(df2['driven_kilometers'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x14401ac7790>,
  <matplotlib.lines.Line2D at 0x14401ac7af0>],
 'caps': [<matplotlib.lines.Line2D at 0x14401ac7e50>,
  <matplotlib.lines.Line2D at 0x14401ad31f0>],
 'boxes': [<matplotlib.lines.Line2D at 0x14401ac7430>],
 'medians': [<matplotlib.lines.Line2D at 0x14401ad3550>],
 'fliers': [<matplotlib.lines.Line2D at 0x14401ad38b0>],
 'means': []}
```



Further, the outliers are treated using IQR method and replacing the same with the median of the feature driven_kilometers.

```
Q1=df3['driven_kilometers'].quantile(0.25)
Q3=df3['driven_kilometers'].quantile(0.75)
IQR=Q3-Q1
lower_boundary=Q1-(1.5*IQR)
upper_boundary=Q3+(1.5*IQR)
print(Q1,Q3,IQR)
print(upper_boundary)
df3['driven_kilometers']=df3['driven_kilometers'].astype('float')
df3['driven_kilometers'].values[df3['driven_kilometers']>upper_boundary]=np.nan
```

```
5849.0 80000.0 44151.0
46226.5
```

```
df3['driven_kilometers']=df3['driven_kilometers'].fillna(df3['driven_kilometers'].median())
```

After treating outliers next step is to go for skewness check. All the numerical features i.e independent features in this dataset are having skewness less than 0.5 so there is no need to treat it .The categorical features are converted into numeric type using label encoder as machine learning models only accepts numerical values.

```python
from sklearn.preprocessing import LabelEncoder
for i in df3.columns:
    if df3[i].dtypes=='object':
        le=LabelEncoder()
        df3[i]=le.fit_transform(df3[i].values.reshape(-1,1))
```

After completing all the above mentioned steps dataset is splitted into two variables x and y.

```python
x=df3.drop('car_price',axis=1)
y=df3['car_price']
```

Standard Scaling is to be applied first before splitting the data into train_test_split.

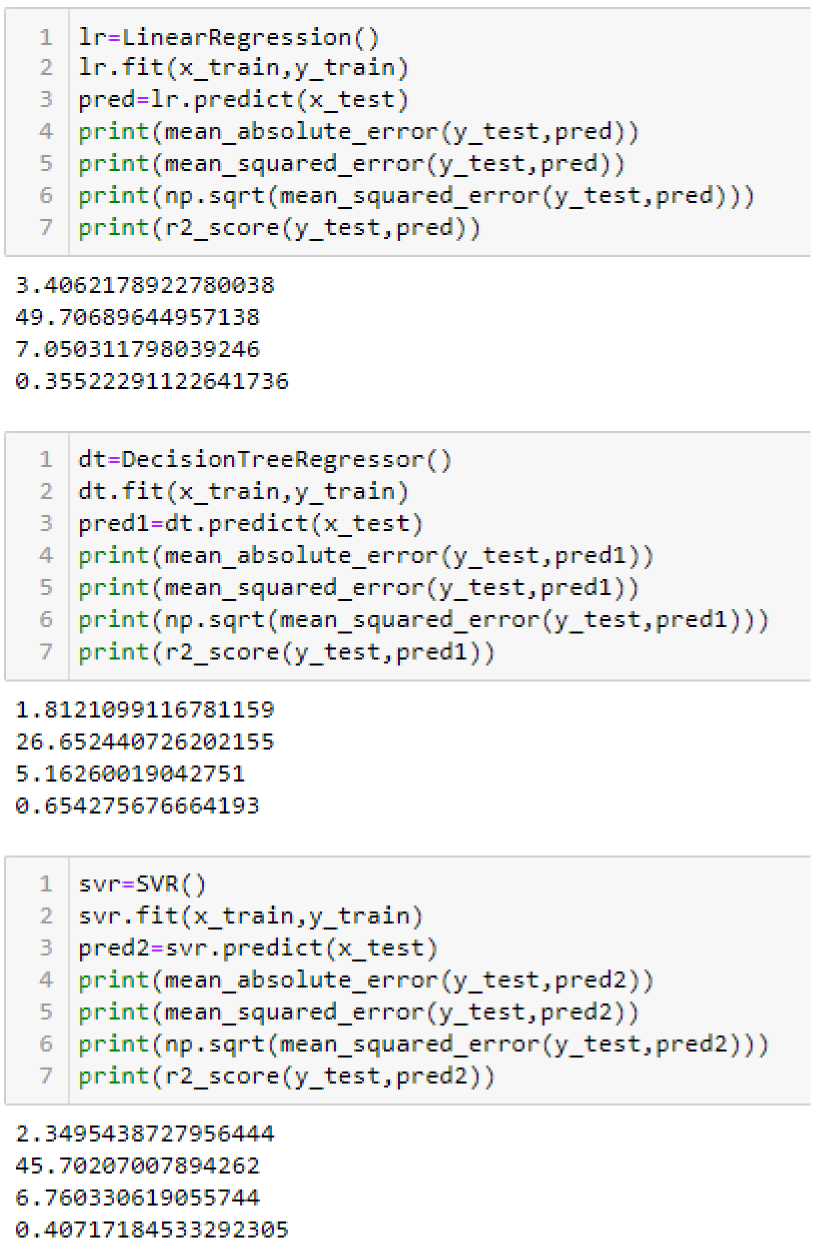
```python
se=StandardScaler()
x=se.fit_transform(x)
```

- ## Testing of Identified Approaches (Algorithms)

  The various algorithms like Linear Regression, Decision Tree Regressor, SVR, KNeighborsRegressor, Ada Boost Regressor, Gradient Boosting Regressor, Random Forest Regressor, Lasso , Ridge are used. First, approach is to find r2 score using all mentioned algorithms. Secondly, cross validation score of all the above mentioned algorithms are checked .The difference between r2 score and cross validation score is minimum for Linear Regression, Gradient Boosting Regressor, Random Forest Regressor, Lasso and ridge. To find the best models amongst these can be done using Hyper tuning where the parameters are tuned to get better r2_score as compared to the one’s without tuning.

- ## Run and Evaluate selected models

  Kindly find attached below the snapshots of various algorithms used



```python
lr=LinearRegression()
lr.fit(x_train,y_train)
pred=lr.predict(x_test)
print(mean_absolute_error(y_test,pred))
print(mean_squared_error(y_test,pred))
print(np.sqrt(mean_squared_error(y_test,pred)))
print(r2_score(y_test,pred))
```

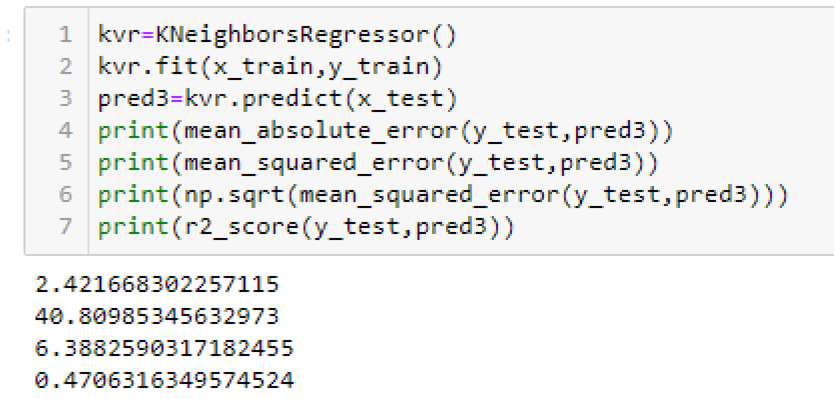
```
3.4062178922780038
49.70689644957138
7.050311798039246
0.35522291122641736
```

```python
dt=DecisionTreeRegressor()
dt.fit(x_train,y_train)
pred1=dt.predict(x_test)
print(mean_absolute_error(y_test,pred1))
print(mean_squared_error(y_test,pred1))
print(np.sqrt(mean_squared_error(y_test,pred1)))
print(r2_score(y_test,pred1))
```

```
1.8121099116781159
26.652440726202155
5.16260019042751
0.654275676664193
```

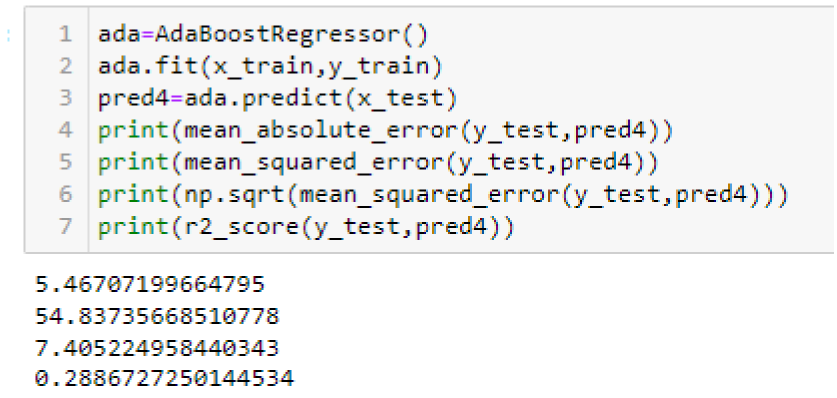
```python
svr=SVR()
svr.fit(x_train,y_train)
pred2=svr.predict(x_test)
print(mean_absolute_error(y_test,pred2))
print(mean_squared_error(y_test,pred2))
print(np.sqrt(mean_squared_error(y_test,pred2)))
print(r2_score(y_test,pred2))
```

```
2.3495438727956444
45.70207007894262
6.760330619055744
0.40717184533292305
```



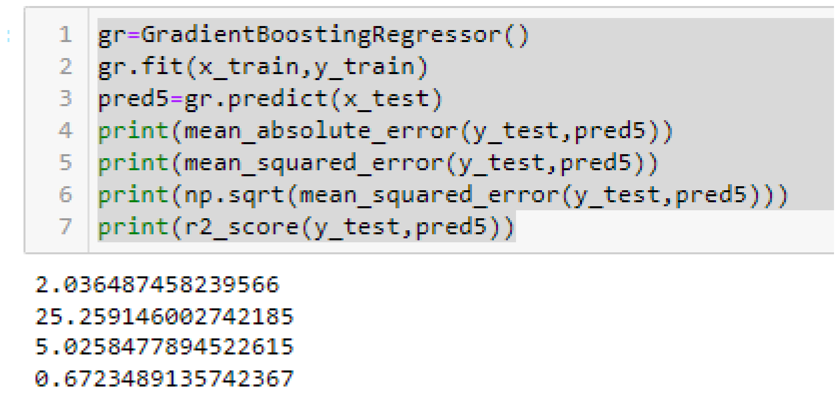
```python
kvr=KNeighborsRegressor()
kvr.fit(x_train,y_train)
pred3=kvr.predict(x_test)
print(mean_absolute_error(y_test,pred3))
print(mean_squared_error(y_test,pred3))
print(np.sqrt(mean_squared_error(y_test,pred3)))
print(r2_score(y_test,pred3))
```

```
2.421668302257115
40.80985345632973
6.3882590317182455
0.4706316349574524
```



```python
ada=AdaBoostRegressor()
ada.fit(x_train,y_train)
pred4=ada.predict(x_test)
print(mean_absolute_error(y_test,pred4))
print(mean_squared_error(y_test,pred4))
print(np.sqrt(mean_squared_error(y_test,pred4)))
print(r2_score(y_test,pred4))
```

```
5.46707199664795
54.83735668510778
7.405224958440343
0.2886727250144534
```



```python
gr=GradientBoostingRegressor()
gr.fit(x_train,y_train)
pred5=gr.predict(x_test)
print(mean_absolute_error(y_test,pred5))
print(mean_squared_error(y_test,pred5))
print(np.sqrt(mean_squared_error(y_test,pred5)))
print(r2_score(y_test,pred5))
```

```
2.036487458239566
25.259146002742185
5.0258477894522615
0.6723489135742367
```

```python
rfc=RandomForestRegressor()
rfc.fit(x_train,y_train)
pred6=rfc.predict(x_test)
print(mean_absolute_error(y_test,pred6))
print(mean_squared_error(y_test,pred6))
print(np.sqrt(mean_squared_error(y_test,pred6)))
print(r2_score(y_test,pred6))
```

```
1.5591617295200713
23.00509413093242
4.796362593771703
0.7015875325116432
```

```python
from sklearn.linear_model import Lasso, Ridge
ls=Lasso(alpha=0.001)
ls.fit(x_train,y_train)
pred7=ls.predict(x_test)
print(mean_absolute_error(y_test,pred7))
print(mean_squared_error(y_test,pred7))
print(np.sqrt(mean_squared_error(y_test,pred7)))
print(r2_score(y_test,pred7))
```

```
3.405340123566653
49.7075950676387
7.050361343054602
0.35521384904479303
```

```python
rd=Ridge(alpha=0.001)
rd.fit(x_train,y_train)
pred8=rd.predict(x_test)
print(mean_absolute_error(y_test,pred8))
print(mean_squared_error(y_test,pred8))
print(np.sqrt(mean_squared_error(y_test,pred8)))
print(r2_score(y_test,pred8))
```

```
3.4062175662085505
49.706896943699114
7.050311833082215
0.35522290481679897
```

```python
score=cross_val_score(lr,x,y,cv=5)
print(score.mean())
```

```
0.3613570203934705
```

```python
score1=cross_val_score(dt,x,y,cv=5)
print(score1.mean())
```

```
0.4637798656520797
```

```python
score2=cross_val_score(svr,x,y,cv=5)
print(score2.mean())
```

```
0.49237115165496964
```

```python
score3=cross_val_score(kvr,x,y,cv=5)
print(score3.mean())
```

```
0.48677647619780917
```

```python
score4=cross_val_score(ada,x,y,cv=5)
print(score4.mean())
```

```
0.21438146703845212
```

```python
score5=cross_val_score(gr,x,y,cv=5)
print(score5.mean())
```

```
0.679297616318818
```

```python
score6=cross_val_score(rfc,x,y,cv=5)
print(score6.mean())
```

```
0.7046873990771002
```

```python
score7=cross_val_score(ls,x,y,cv=5)
print(score7.mean())
```

```
0.36140998906472904
```

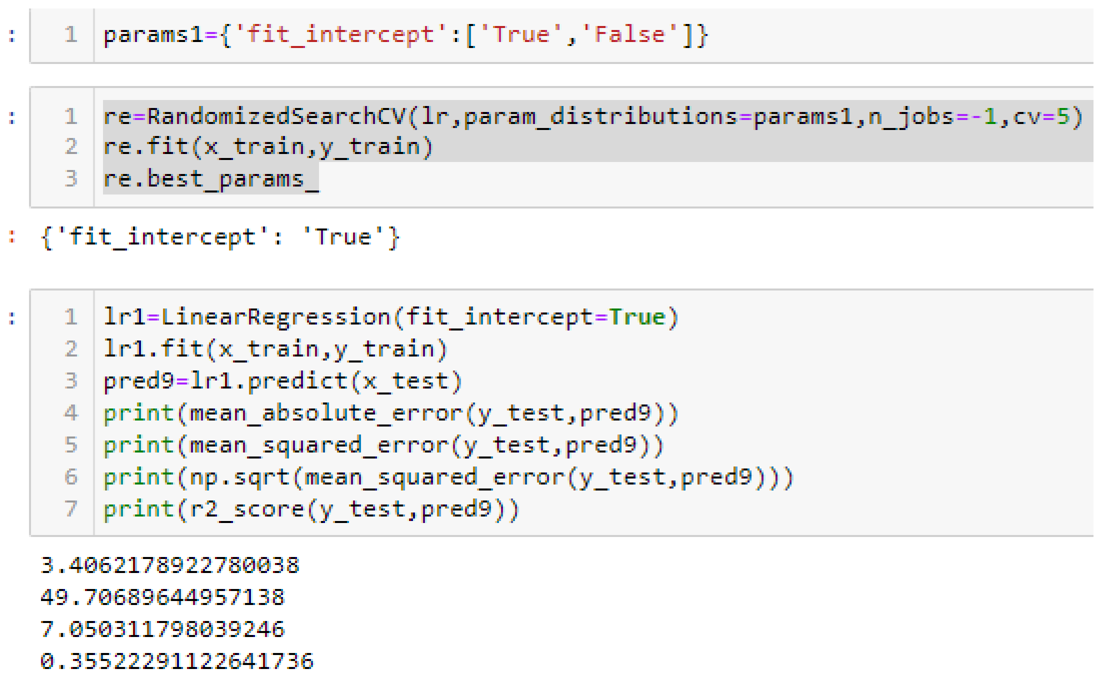
```python
score8=cross_val_score(rd,x,y,cv=5)
print(score8.mean())
```

```
0.36135705170530097
```

```python
score8=cross_val_score(rd,x,y,cv=5)
print(score8.mean())
```

```
0.36135705170530097
```

Hyper tuning of Linear Regression, Gradient Boosting Regressor, Random Forest Regressor, Lasso and ridge. is carried out to find out if the r2_score can be improved or not.



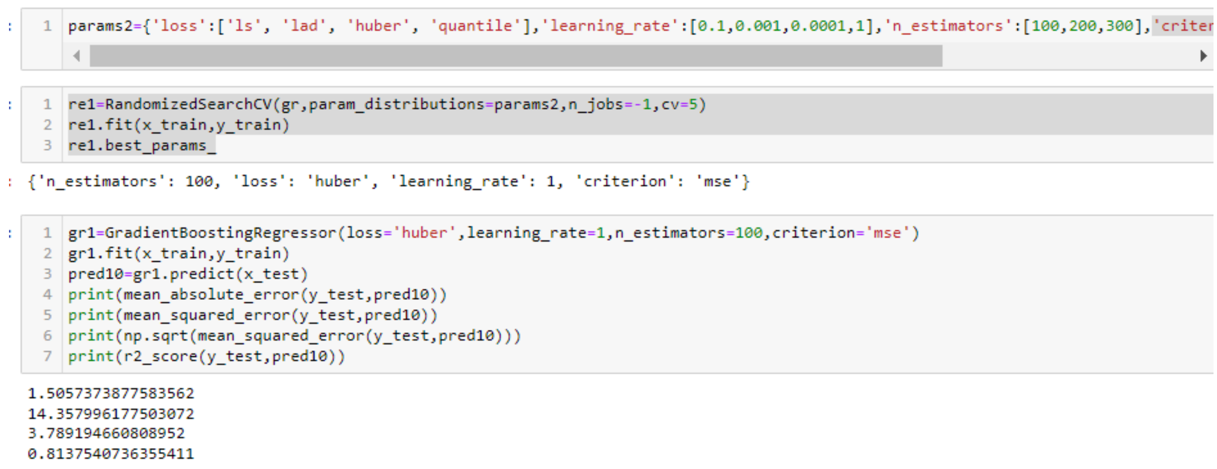
```
params1={'fit_intercept':['True','False']}
```

```
re=RandomizedSearchCV(lr,param_distributions=params1,n_jobs=-1,cv=5)
re.fit(x_train,y_train)
re.best_params_
```

```
{'fit_intercept': 'True'}
```

```
lr1=LinearRegression(fit_intercept=True)
lr1.fit(x_train,y_train)
pred9=lr1.predict(x_test)
print(mean_absolute_error(y_test,pred9))
print(mean_squared_error(y_test,pred9))
print(np.sqrt(mean_squared_error(y_test,pred9)))
print(r2_score(y_test,pred9))
```

```
3.4062178922780038
49.70689644957138
7.050311798039246
0.35522291122641736
```



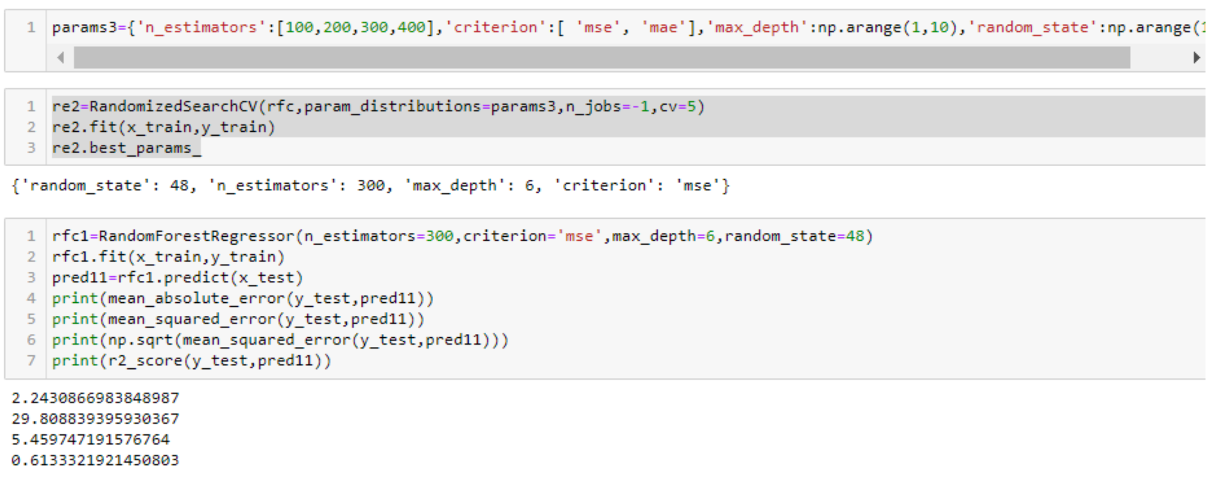
```
params2={'loss':['ls', 'lad', 'huber', 'quantile'],'learning_rate':[0.1,0.001,0.0001,1],'n_estimators':[100,200,300],'criter
```

```
re1=RandomizedSearchCV(gr,param_distributions=params2,n_jobs=-1,cv=5)
re1.fit(x_train,y_train)
re1.best_params_
```

```
{'n_estimators': 100, 'loss': 'huber', 'learning_rate': 1, 'criterion': 'mse'}
```

```
gr1=GradientBoostingRegressor(loss='huber',learning_rate=1,n_estimators=100,criterion='mse')
gr1.fit(x_train,y_train)
pred10=gr1.predict(x_test)
print(mean_absolute_error(y_test,pred10))
print(mean_squared_error(y_test,pred10))
print(np.sqrt(mean_squared_error(y_test,pred10)))
print(r2_score(y_test,pred10))
```

```
1.5057373877583562
14.357996177503072
3.789194660808952
0.8137540736355411
```



```
params3={'n_estimators':[100,200,300,400],'criterion':[ 'mse', 'mae'],'max_depth':np.arange(1,10),'random_state':np.arange(1
```

```
re2=RandomizedSearchCV(rfc,param_distributions=params3,n_jobs=-1,cv=5)
re2.fit(x_train,y_train)
re2.best_params_
```

```
{'random_state': 48, 'n_estimators': 300, 'max_depth': 6, 'criterion': 'mse'}
```

```
rfc1=RandomForestRegressor(n_estimators=300,criterion='mse',max_depth=6,random_state=48)
rfc1.fit(x_train,y_train)
pred11=rfc1.predict(x_test)
print(mean_absolute_error(y_test,pred11))
print(mean_squared_error(y_test,pred11))
print(np.sqrt(mean_squared_error(y_test,pred11)))
print(r2_score(y_test,pred11))
```

```
2.2430866983848987
29.808839395930367
5.459747191576764
0.6133321921450803
```

```
params4={'fit_intercept':['True','False',],'alpha':[0.1,0.001,0.001,1],'random_state':np.arange(1,100)}
```

```
re3=RandomizedSearchCV(ls,param_distributions=params4,n_jobs=-1,cv=5)
re3.fit(x_train,y_train)
re3.best_params_
```

```
{'random_state': 69, 'fit_intercept': 'False', 'alpha': 0.001}
```

```
ls1=Lasso(alpha=0.001,fit_intercept='false',random_state=69)
ls1.fit(x_train,y_train)
pred12=ls1.predict(x_test)
print(mean_absolute_error(y_test,pred12))
print(mean_squared_error(y_test,pred12))
print(np.sqrt(mean_squared_error(y_test,pred12)))
print(r2_score(y_test,pred12))
```

```
3.405340123566653
49.7075950676387
7.050361343054602
0.35521384904479303
```

```
params5={'fit_intercept':['True','False',],'alpha':[0.1,0.001,0.001,1],'random_state':np.arange(1,100)}
```

```
re4=RandomizedSearchCV(rd,param_distributions=params5,n_jobs=-1,cv=5)
re4.fit(x_train,y_train)
re4.best_params_
```

```
{'random_state': 95, 'fit_intercept': 'False', 'alpha': 1}
```

```
rd1=Ridge(alpha=1,fit_intercept='false',random_state=95)
rd1.fit(x_train,y_train)
pred13=rd1.predict(x_test)
print(mean_absolute_error(y_test,pred13))
print(mean_squared_error(y_test,pred13))
print(np.sqrt(mean_squared_error(y_test,pred13)))
print(r2_score(y_test,pred13))
```

```
3.4058919047696015
49.70739159787322
7.0503469132996015
0.355216488369563
```

Based on Hypertuning Gradient Boosting Regressor is my best model with r2_score of 81%.

Key Metrics for success in solving problem under consideration

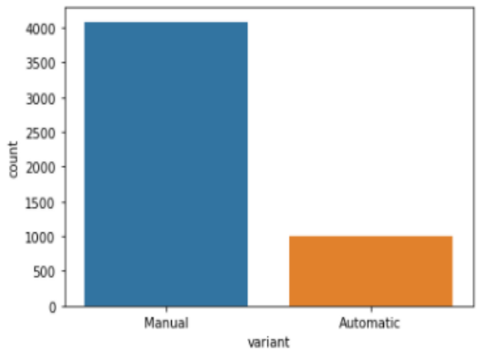
IQR method is used to treat outliers . If I would have used zscore instead of IQR there would have been more data loss ,so by using IQR method data loss is avoided as data is very vital and costly.

- ### Visualizations
  Data Visualization is the graphical representation of the information and data. By using visual elements like charts, graphs and maps data

visualization tools provide an accessible way to see and understand trends, outliers and patterns in data.

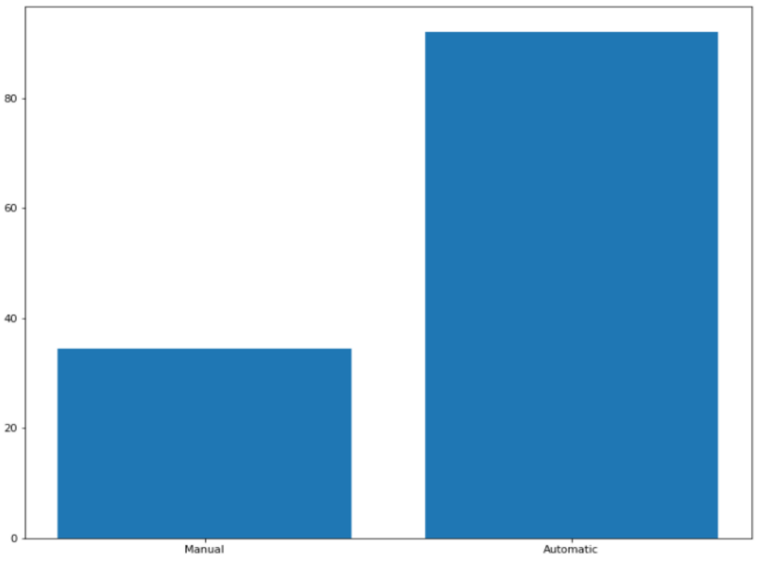
```
sns.countplot(x='variant',data=df2)
```

```
<AxesSubplot:xlabel='variant', ylabel='count'>
```



```
As per the dataset 80%(4085 out of 5091) of the records are having manual transmission pertaining to used cars available
for sale.
```

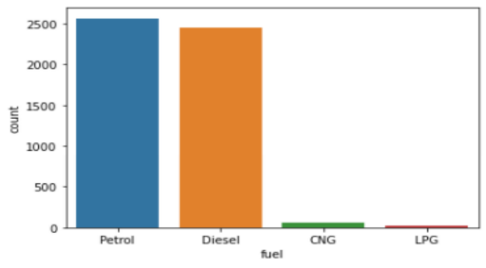
```
plt.figure(figsize=[12,10])
plt.bar(df2['variant'],df2['car_price'])
plt.show()
```



From the above it can be said the cars having automatic transmission are costly as compared to the one's having manual transmission

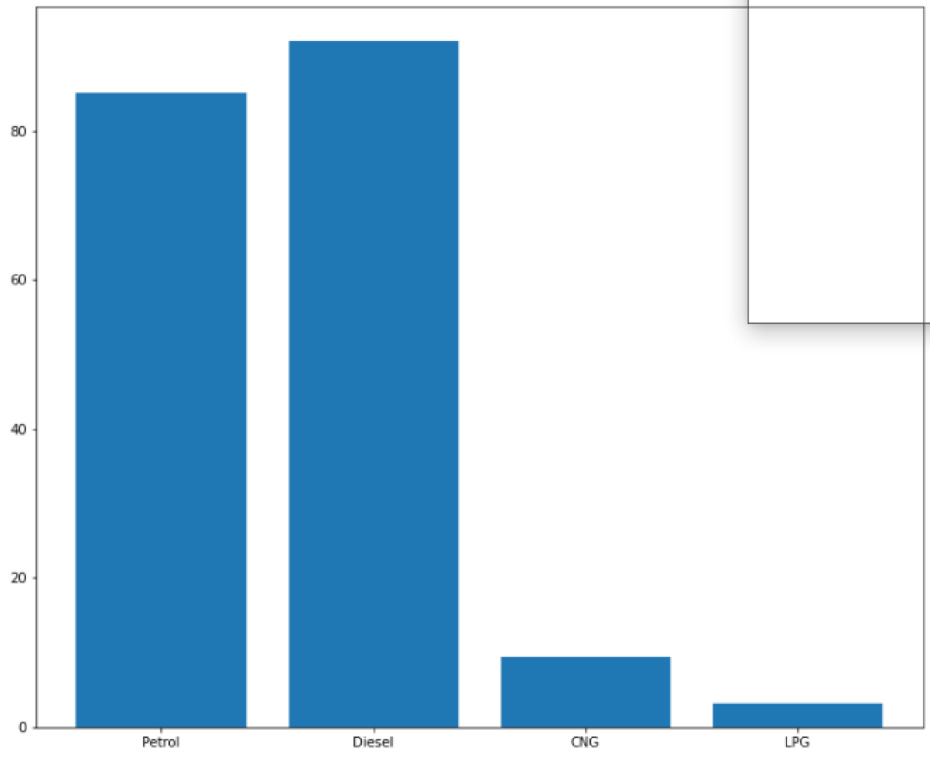
```
sns.countplot(x='fuel',data=df2)
```

```
<AxesSubplot:xlabel='fuel', ylabel='count'>
```



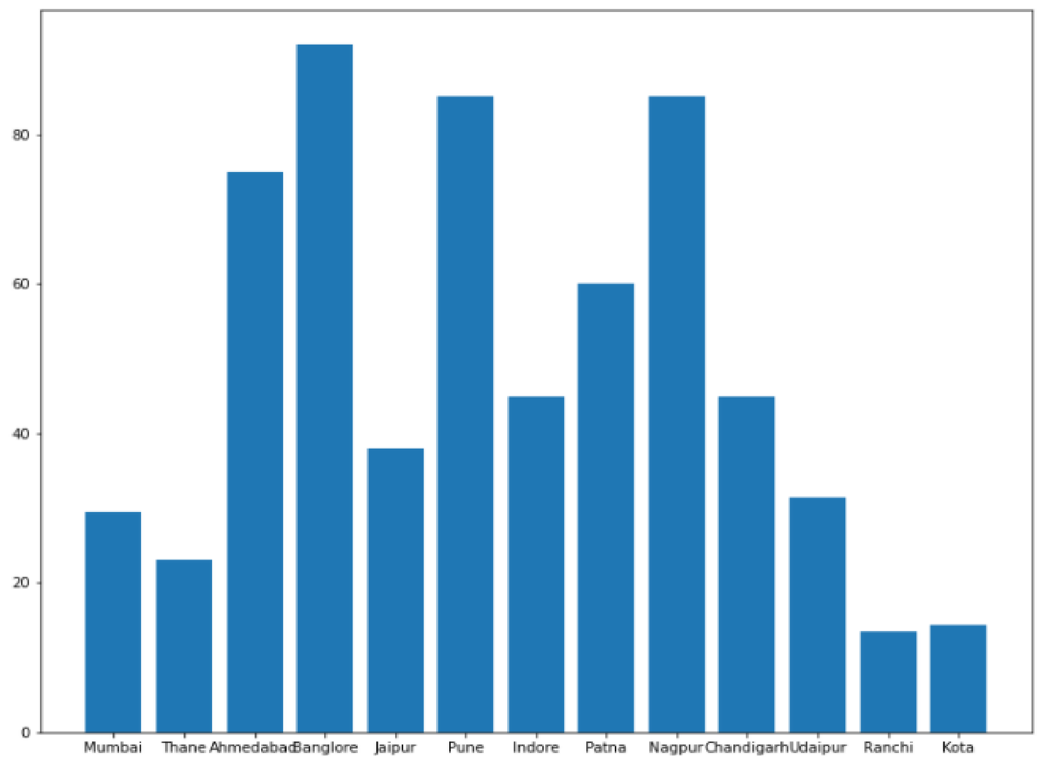
```
Most the used cars available for sale are running on petrol and diesel.Further,0.01%(52 out of 5091)cars are running on CNG
and 0.0003%(22 out of 5091) are running on LPG.
```

```
plt.figure(figsize=[12,10])
plt.bar(df2['fuel'],df2['car_price'])
plt.show()
```



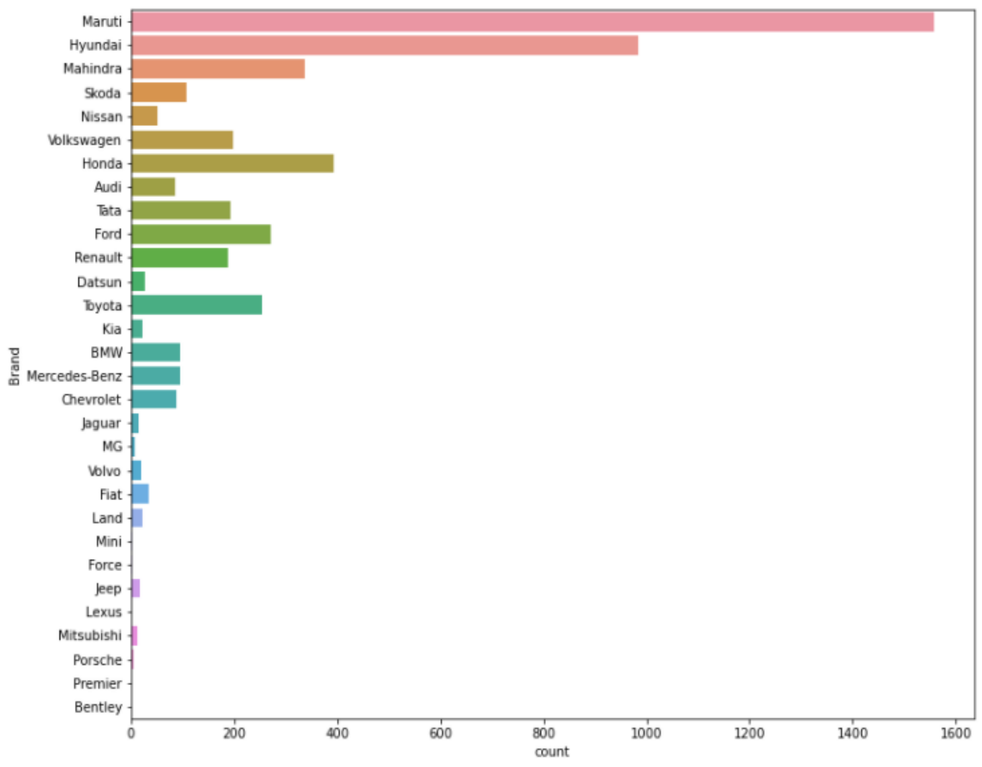
The cars running on diesel are found to be costlier as compared to cars running on petrol.

```
plt.figure(figsize=[12,10])
plt.bar(df2['location'],df2['car_price'])
plt.show()
```



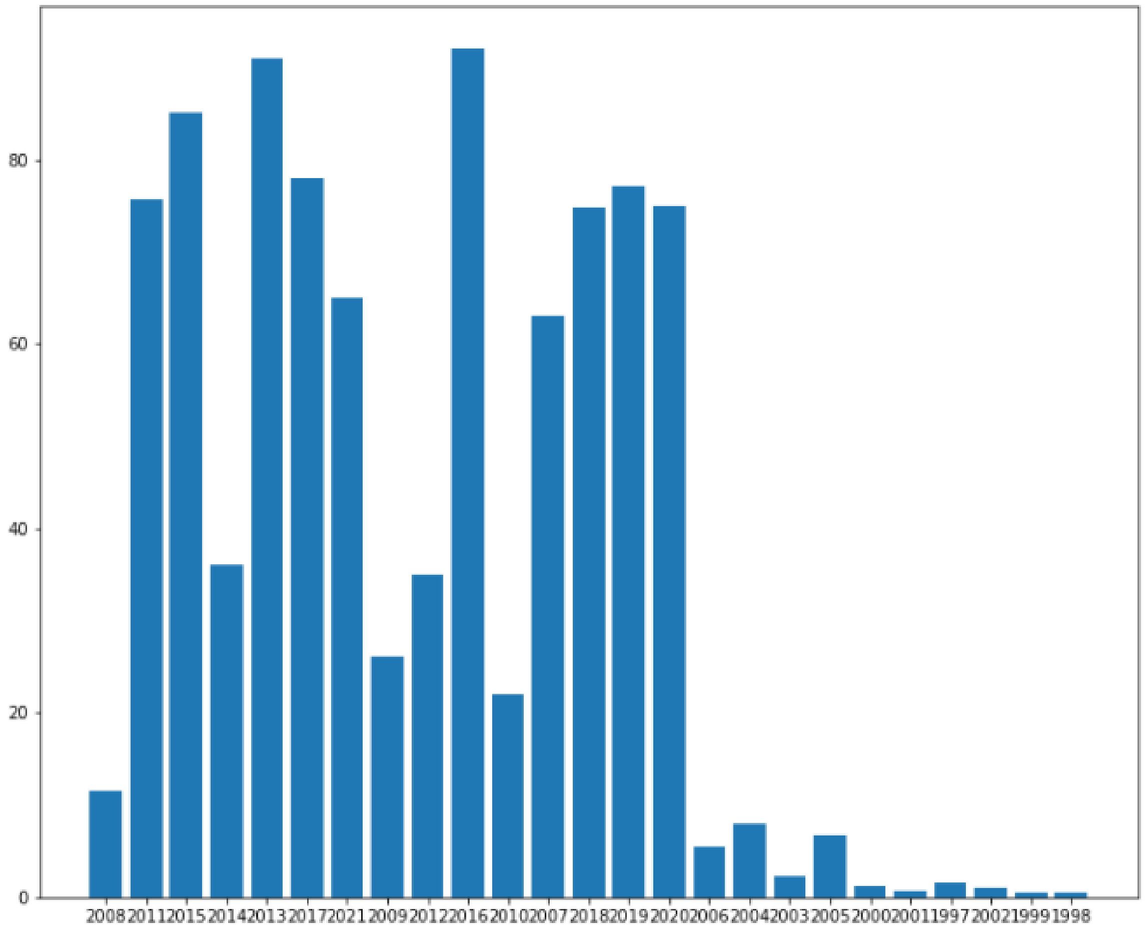
Out of all the locations,Bangalore is costly when it comes to used cars available for sale

```python
plt.figure(figsize=[12,10])
sns.countplot(y='Brand',data=df2)
plt.show()
```



```
Most of the used cars available for sale are for Maruti brand and the least are for Lexus, Bentley etc. Further, the
premium Brand cars like Lexus, Bentley are costlier and it is not afforable by many to buy the same.
The brands like Maruti, Hyundai etc cars are afforable by many as they are cheaper compared to other premium brands and
hence there are many buyers for this cars.
```

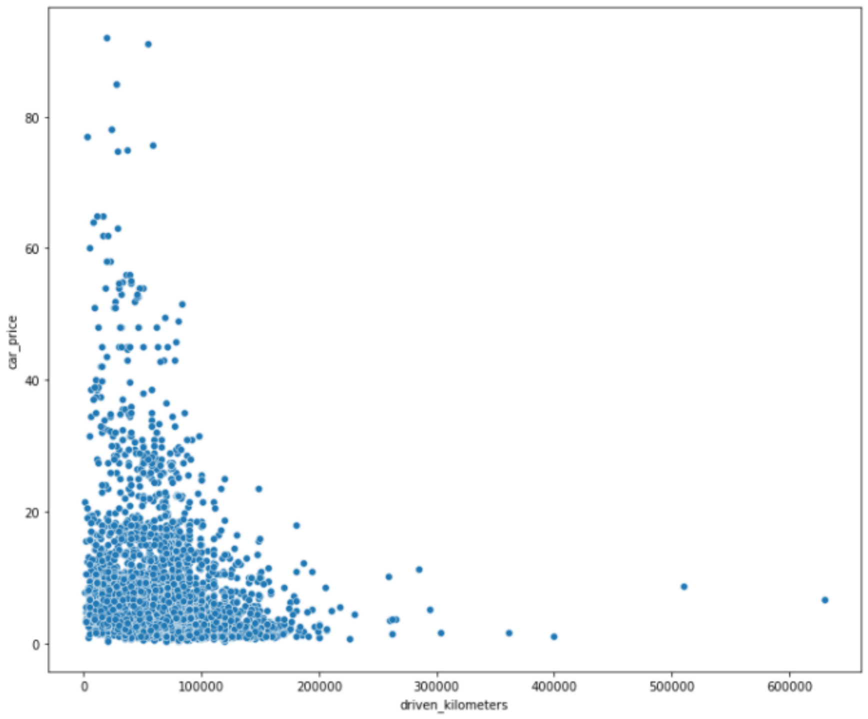
```python
plt.figure(figsize=[12,10])
plt.bar(df2['manufacturing year'],df2['car_price'])
plt.show()
```



The cars which are maufactured in the year from 1997 to 2004 and available for selling are cheaper as compared to those cars manufactured in the year from 2017 to 2020.

The car price keeps on depreciating with time older the car manufacturing year cheaper it is depending on the condition of car.

```python
plt.figure(figsize=[12,10])
sns.scatterplot(x='driven_kilometers',y='car_price',data=df2)
plt.show()
```



From the scatterplot it can be said for most of the records the driven kilometer lies in the range of 30000 to 100000 kms. There are only few records where the driven kilometer is more than 300000 kms. As such no relationship can be established between features driven_kilometers and car_price.

- ## Interpretation of the Results

  From visualization the key findings like skewness, outliers present in the dataset were known. Further, the heat map was drawn showing the correlation with target feature and based on it the one's having less correlation with the target feature can be known and if the need arises can be dropped.

  Various other details can be drawn out from visualization like say for suppose I am taking feature Brand which is categorical in nature. After applying visualization techniques i.e. sns.countplot on the feature Brand I can find out the maximum records are for which brand.

  From the scatterplot one can visualize the relationship between two features i.e. whether they are linearly positively correlated or linearly negatively correlated or the case may be there exist no relation between those features.

CONCLUSION

- Key Findings and Conclusions of the Study

  Gradient Boosting Regressor is my best model with r2 score of 81%
  .

- Learning Outcomes of the Study in respect of Data Science

  This case study helped me learn how to treat outliers without removing them using IQR method, how to hyper tune the parameters using RandomizedSeachCV to get the best accuracy from the model.

- Limitations of this work and Scope for Future Work

  This study used different models in order to predict used car prices. However, there was a relatively small dataset for making a strong inference because number of observations were only 5091. Gathering more data can yield more robust predictions.

  As suggestion for further studies, while pre-processing data, instead of using label encoder, one hot encoder method can be used. Thus, all non-numeric features can be converted to nominal data instead of ordinal data. This may cause a serious change in performance of predictive models. Different scalers can be checked whether there is an improvement in prediction power of models or not.