Final Project Report
USED CARS PRICE PREDICTION

VARAPRASAD RAO KURRA

27th April 2020

Business Understanding

The project topic is Used Cars price prediction. This project will help us to know the value of the used cars in the present-day market based upon its descriptive features. We try to build our model that will help us to predict the price of the cars. There is an online site called CARS24, they buy used cars and Guarantee to offer the owner the best price for their car. The motivation for selecting this project is to analyze the factors that affect the price of the used cars and derive a model that will help to gain some understanding how the cost of the used cars can vary.

The dataset consists 13 columns each describing the feature of the car. Like Kilometers_Driven, Fuel_Type, Mileage, Engine and its power will help us to understand the data well. The target variable will be New_Price that is dependent on descriptive features mentioned. The price of the car is always dependent on the Fuel_Type, Mileage, Engine and its Horsepower. Since we have all the features that describe a car in our dataset, we can make the best use of the data.

Business Problem

CARS24 is failing to meet its customer requirements when dealing with the purchase of the used cars. Due that it's facing the customer churns. The main problem the organization facing is in predicting the cost of the used cars. This will be best helping the user and CARS24 to get understanding about how the price of each car can be.

Dataset

The dataset we are dealing with has 14 columns where each attribute describes the car. Dataset speaks about the all the features that we take into consideration whenever we are trying to buy a car. The dataset train-data.csv contains 6019 rows and 14 cols. Our solution is to best use the training dataset and build a model for predicting what would be the cost of a used car in the present-day scenario. Our dataset involves both continuous and categorical data.

Dataset Link : https://www.kaggle.com/avikasliwal/used-cars-price-prediction#train-data.csv

Proposed Analytics Solution

We will be building a model that will best use the dataset to predict the used car price. This will help CARS24 to put the accurate cost of the car by taking its descriptive features into the consideration. We will try to find out the features that plays major role in deciding the price of the car. Running this predictive model everyday will keep the user updated about the new prices of the car.

We will be performing the **Regression analysis** to predict the price of the Used car. We will use Linear Regression, Random Forest Regression, KNN Regression for predicting.

Data Exploration and Preprocessing

In the Data Exploration part, we will be looking into the attributes of our dataset and how we will be modifying our attributes to form our Analytics Base Table. In total used-car-price-prediction has 14 attributes where 13 among them are Descriptive Features and 1 Target Feature. We will now look into the properties of the Attributes and discuss what are all the modifications we will be doing on the attributes to build our Analytics Base Table.

In our dataset we have *14 attributes*. Below is the gist of the Dataset attributes.

```
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         6019 non-null   int64
 1   Name               6019 non-null   object
 2   Location           6019 non-null   object
 3   Year               6019 non-null   int64
 4   Kilometers_Driven  6019 non-null   int64
 5   Fuel_Type          6019 non-null   object
 6   Transmission       6019 non-null   object
 7   Owner_Type         6019 non-null   object
 8   Mileage            6017 non-null   object
 9   Engine             5983 non-null   object
 10  Power              5983 non-null   object
 11  Seats              5977 non-null   float64
 12  New_Price          824 non-null    object
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
```

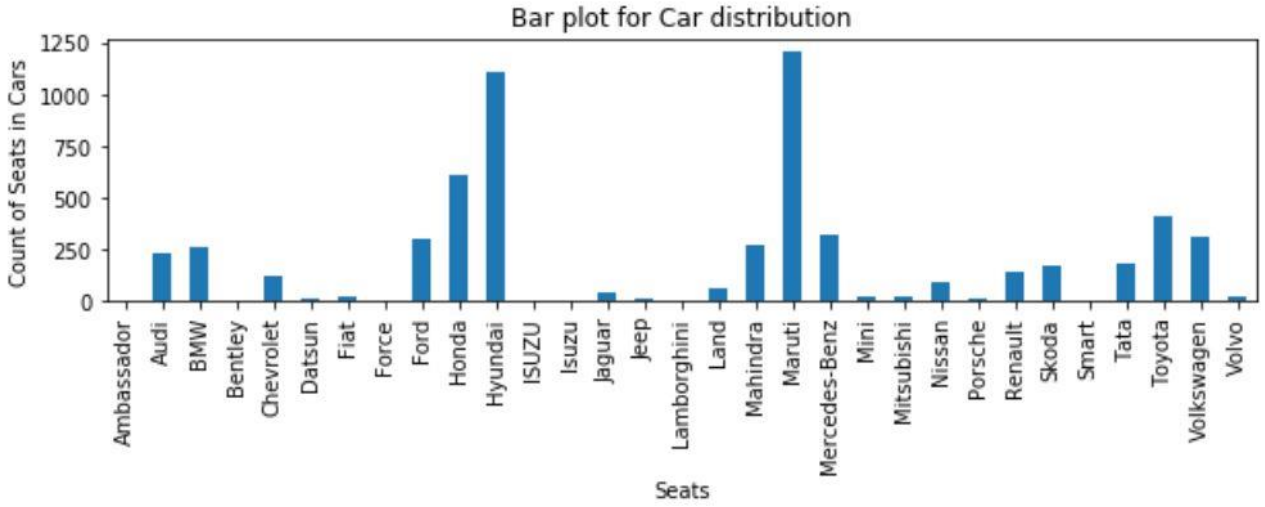
If we look at our dataset there are 2 float64 attributes, 3 integer attributes and 9 object types. Before discussing about our attributes, we will look into a sample of our data. Below is the screenshot of first five records of the dataset.

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN	12.50
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.61 Lakh	4.50
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	NaN	17.74

Above is a simple snippet of the sample records from the dataset.

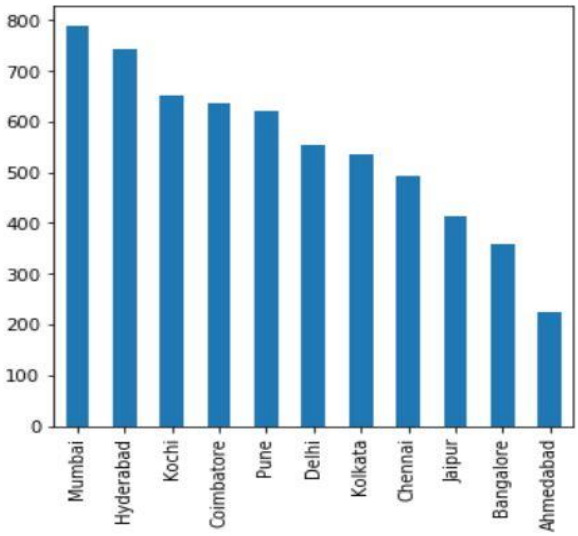
First attribute is Unnamed: 0, this is not a useful attribute because this is nothing but an index column. We already have the default index as displayed therefore this attribute will not affect our results. Hence this attribute can be safely discarded from the dataset.

Second Attribute is Name, this attribute gives the user the Name of the car. This is a typical attribute with most cardinality number about 1876. When handling this attribute, we trimmed the data based on the Brand name of the car. Because the price of the car is dependent on the name of the brand name. Unfortunately, the cardinality is the major drawback of this attribute. So, we tried to modify the attribute values based on the Brand Name (which is nothing, but first occurrence separated by space). So the below graph will demonstrate the work.



We converted this because this should not become a problem during the creation of dummy variables. Creating 1876 dummy columns for applying regression algorithms is of no use.

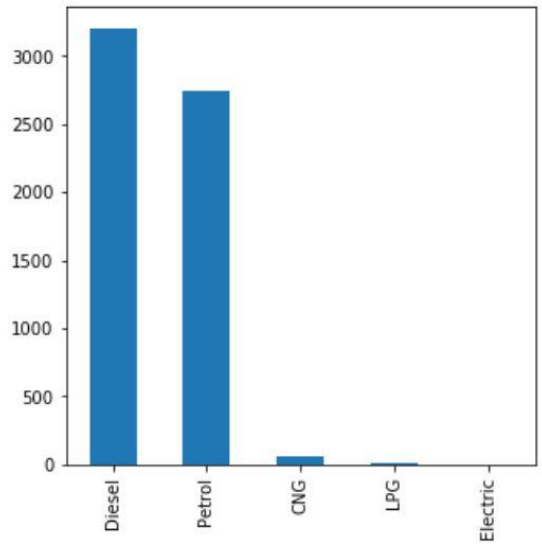
Third column is the location, this is a simple attribute specifying the location of the car. This attribute will also impact the Target Variable because change of location causes change in Car price. Below is the distribution if the Location attribute.



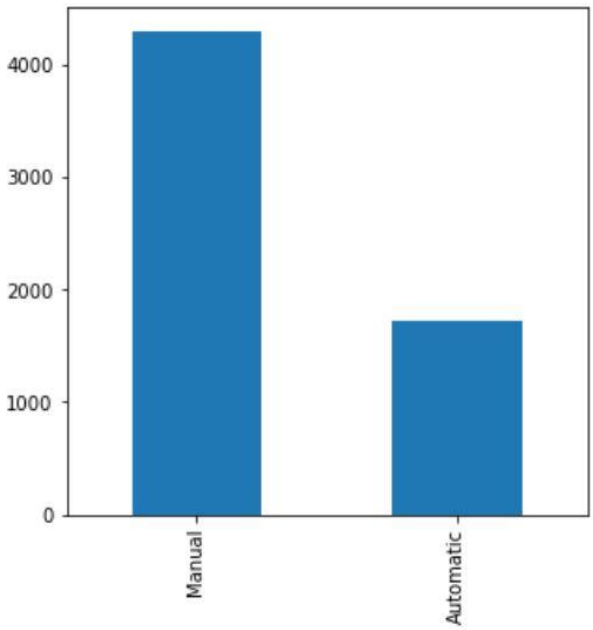
The next is **Kilometer driven** continuous variable. It is also a key attribute to predict the price of the Car. In general, as kilometers driven increase the cost price must decrease. This attribute is having values

in large.

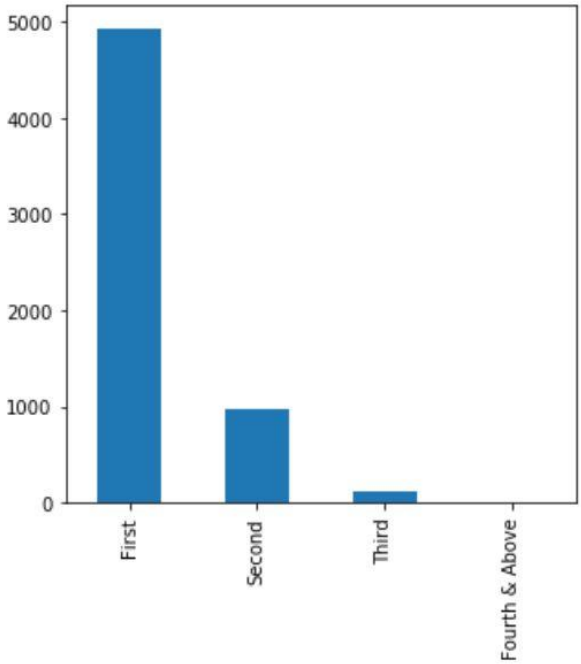
Coming to **Fuel Type**, this is categorical data with 5 entries describing the types describing the type of fuel each car is using. We can encode this during our regression analysis. Below is the gist briefing Fuel Type.



The next attribute is Transmission Type. It is binary attribute it has two attributes namely Automatic and Manual. We can observe that there is a cost variance between these two types of cars. So let's see how these values drafted.



Owner Type: This attribute describes the owner type. Let's see the distribution of Owner Type. This attribute impacts the cost of the car as the owner type changes the cost will also degrade.

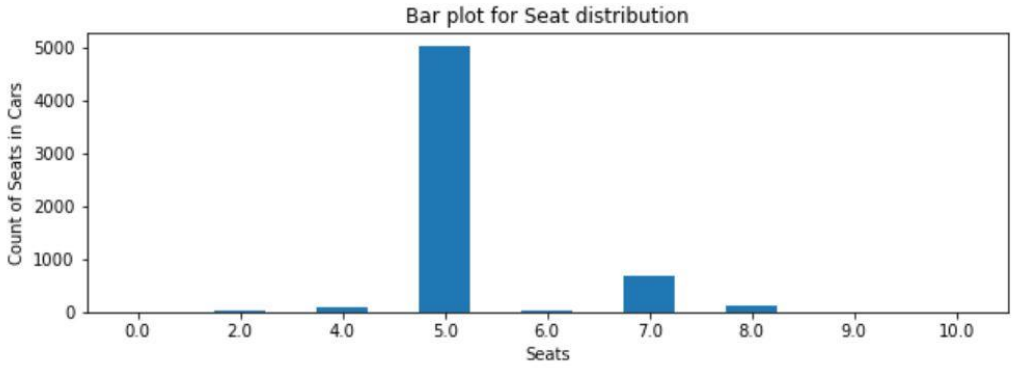


Mileage: This Continuous Data best helps to construct the price of the car. One of the best features that helps to know about the car.

Engine: This attribute specifies the engine of the car measured in CC. It is obvious that as the Engine performance degrades cost also influences by it. This is also a continues variable that changes from car to car.

Power: Another key feature that will have an impact the price of the car. Well this is also a continuous variable.

Seats: This is a kind of ordinal data where we rank the cars based on the number of seats. As the seats increase the car price will also increase. So, this attribute does an impact on the target variable. Let's see the distribution of this feature. The majority



New Price: This is a column that shows price of the current cars price in the market. We will discuss about this attribute in the data cleaning process. This attribute holds the most missing values.

Price: Finally, the target variable Price of the Car. A continuous variable that is dependent on all the attributes discussed.

Data Quality Report

A Data Quality Report is the most important tool of the data exploration process. A data quality report includes tabular reports one for **continuous features** and **one for categorical features** that describe the characteristics of each feature in an Analytical Base Table using standard statistical measures of **central tendency** and **variation**.

Now let's look into Data Quality Reports for both Continuous and Categorical Features.

1. Data Quality Report for Continuous Feature.

	Count	%Miss	Card.	Min	1st_Qrt	Mean	Median	3rd_Qrt	Max	Std.Dev.
Year	6019	0.000	22	1998.00	2011.0	2013.36	2014.00	2016.00	2019.0	3.27
Kilometers_Driven	6019	0.000	3093	171.00	34000.0	58738.38	53000.00	73000.00	6500000.0	91268.84
Mileage	6017	0.033	442	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Engine	5983	0.602	146	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Power	5983	0.602	372	NaN	NaN	NaN	NaN	NaN	NaN	NaN
New_Price	824	630.461	540	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Price	6019	0.000	1373	0.44	3.5	9.48	5.64	9.95	160.0	11.19

If we look at the Data Quality report of the Continuous Data. We can observe that there are some NaN values. This is because the data is preprocessed, and the respective attributes are in String (Object format). These need to be preprocessed and converted into float64 data format, we will look into this data quality report after the preprocessing is done.

2. Data Quality Report Categorical Data

	Count	%Miss	Card.	Mode	Mode_Freq.	Mode_%	2nd_Mode	2nd_Mode_Freq.	2nd_Mode_%
Location	6019	0.000	11	Mumbai	790	13.1251	Hyderabad	742	12.3276
Fuel_Type	6019	0.000	5	Diesel	3205	53.248	Petrol	2746	45.6222
Transmission	6019	0.000	2	Manual	4299	71.4238	Automatic	1720	28.5762
Owner_Type	6019	0.000	4	First	4929	81.8907	Second	968	16.0824
Seats	5977	0.703	9	5	5014	83.8882	7	674	11.2766

Categorical data's attributes in Data Quality Report varies from that of the attribute in the Continuous Data Quality Report. We can observe the Mode and Mode %, 2nd Mode and 2nd Mode % in the above data quality report.

Missing Values and Outliers

Missing Values:

Now its time to the look into the missing values of our dataset. A missing value can be in be any format like the record is completely blank or be placed by a null value or placed by a NULL value or placed by a NaN value. Such kind of data must be explicitly handled by the user before working on the data. So, we will try to replace such kind of data with a mean value or a Median or a Mode value of that attribute.

It is argued that if the % of missing values of an attribute is more than 60% the attribute can be deleted from our dataset. In our dataset if we observe New_Price has 85% of missing values thus this attribute can be discarded from the dataset.

Outliers:

An outlier is a data that varies from other observations. An outlier may be due to variability in the measurement; the latter are sometimes excluded from the data set. Let us investigate the outliers of our dataset. We use boxplots to represent the Outliers of an attribute. These plots use the concept of quantiles. Q1 and Q3 holds the 25^{th} % and 75^{th} % values of the Dataset.

So, Inter Quantile Range is defined as **IQR = Q1 – Q3**.

Lower Outlier Bound is given by equation **Q1 – 1.5*IQR**

Upper Outlier Bound is given by equation **Q3 + 1.5*IQR**

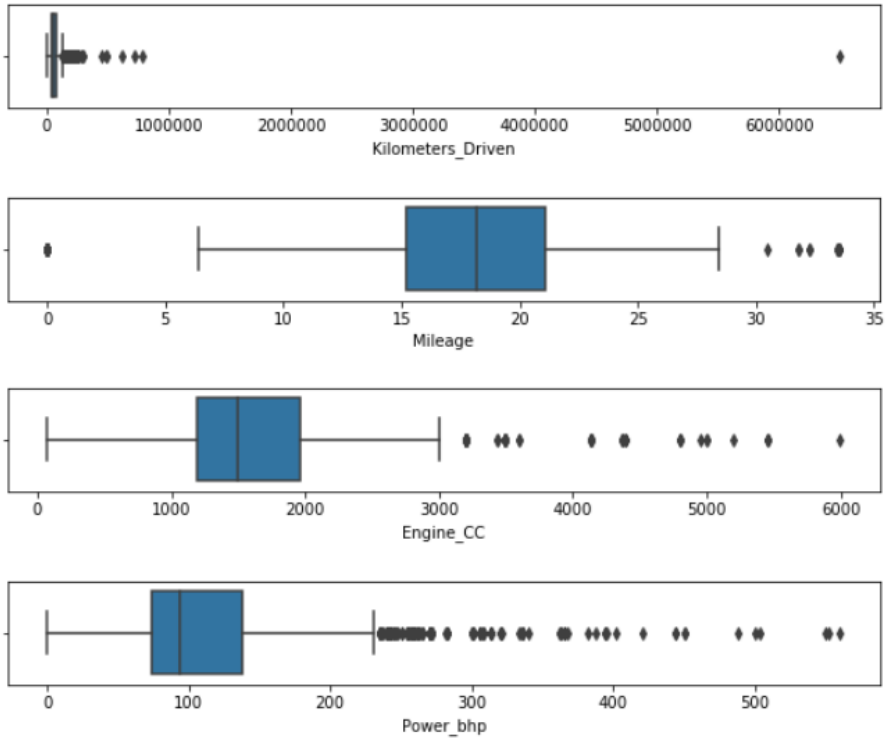
```
First Quantile
Unnamed: 0            1504.5
Year                  2011.0
Kilometers_Driven    34000.0
Seats                    5.0
Price                    3.5
Name: 0.25, dtype: float64
```

```
Third Quantile
Unnamed: 0            4513.50
Year                  2016.00
Kilometers_Driven    73000.00
Seats                    5.00
Price                    9.95
Name: 0.75, dtype: float64
```

IQR Values

```
IQR Values
Unnamed: 0            3009.00
Year                     5.00
Kilometers_Driven    39000.00
Seats                    0.00
Price                    6.45
dtype: float64
```

Hence, the observations that fall below the Lower Outlier Bound and the observations that fall above Upper Outlier Bound are the outliers of the Data. Let us investigate the outlier of the outliers in a graphic al representation of the dataset. We use boxplots to represent the outliers here in this section.



Binning:

```
1 bin_data = data.copy()
2 print(bin_data)
3
4
5 bin_data['Kilometers_Driven_1'] = pd.qcut(bin_data['Kilometers_Driven'], q=4)
6 bin_data['Kilometers_Driven_2'] = pd.qcut(bin_data['Kilometers_Driven'], q=10, precision=0)
7
8 bin_data.head()
```

Results of the Binning:

Kilometers_Driven
72000
41000
46000
87000
40670

Kilometers_Driven_1	Kilometers_Driven_2
(53000.0, 73000.0]	(68390.0, 79000.0]
(34000.0, 53000.0]	(38000.0, 45351.0]
(34000.0, 53000.0]	(45351.0, 53000.0]
(73000.0, 6500000.0]	(79000.0, 97000.0]
(34000.0, 53000.0]	(38000.0, 45351.0]

Normalization

As we detected the Outliers using the quantile concept. We do our data normalization using the same concept. We clip the outliers between the lower and upper bound limits. This will remove the outliers and clip the outlier values within the boundaries.

	Count	%Miss	Card.	Min	1st_Qrt	Mean	Median	3rd_Qrt	Max	Std.Dev.
Kilometers_Driven	6019	0.0	3093	171.00	34000.00	58738.38	53000.00	73000.00	6500000.00	91268.84
Mileage	6019	0.0	431	0.00	15.17	18.13	18.15	21.10	33.54	4.58
Engine_CC	6019	0.0	147	72.00	1198.00	1621.28	1493.00	1969.00	5998.00	599.55
Power_bhp	6019	0.0	371	0.00	74.00	111.23	93.70	138.03	560.00	55.29
Price	6019	0.0	1373	0.44	3.50	9.48	5.64	9.95	160.00	11.19

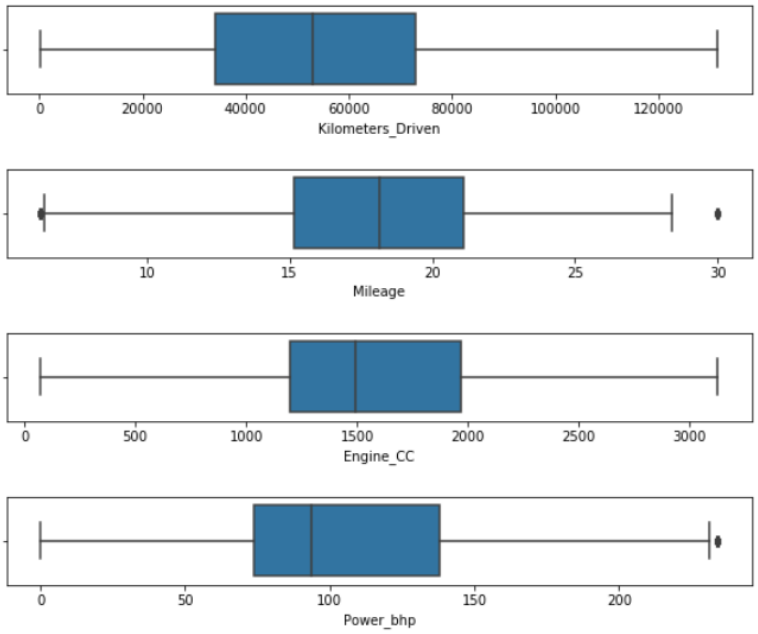
outlier_lower_bound	outlier_upper_bound
-24500.00	131500.00
6.27	30.00
41.50	3125.50
-22.04	234.08
-6.17	19.62

We will see the boxplots after the normalization below. Data after normalization.

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine_CC	Power_bhp	Seats	Price
0	Maruti	Mumbai	2010	72000	CNG	Manual	First	26.60	998.0	58.16	5.0	1.75
1	Hyundai	Pune	2015	41000	Diesel	Manual	First	19.67	1582.0	126.20	5.0	12.50
2	Honda	Chennai	2011	46000	Petrol	Manual	First	18.20	1199.0	88.70	5.0	4.50
3	Maruti	Chennai	2012	87000	Diesel	Manual	First	20.77	1248.0	88.76	7.0	6.00
4	Audi	Coimbatore	2013	40670	Diesel	Automatic	Second	15.20	1968.0	140.80	5.0	17.74
...	...	...	...	...	...	...	...	...	...	...	...	...
6014	Maruti	Delhi	2014	27365	Diesel	Manual	First	28.40	1248.0	74.00	5.0	4.75
6015	Hyundai	Jaipur	2015	100000	Diesel	Manual	First	24.40	1120.0	71.00	5.0	4.00
6016	Mahindra	Jaipur	2012	55000	Diesel	Manual	Second	14.00	2498.0	112.00	8.0	2.90
6017	Maruti	Kolkata	2013	46000	Petrol	Manual	First	18.90	998.0	67.10	5.0	2.65
6018	Chevrolet	Hyderabad	2011	47000	Diesel	Manual	First	25.44	936.0	57.60	5.0	2.50

Boxplots after the normalization.

If we look at the below boxplots the outliers, we saw in the above section have been clipped to within the range. Box plots indicate of the data's symmetry and skewness. Unlike many other digrams of data display, **boxplots** show **outliers**.



Feature Selection and Transformations

Feature selection is one of the important steps in building a Machine Learning model. In this phase we will do our research in which attributes will best help the most useful with respect to the target variable.

With respect to this phase we will identify the attributes that will best support the Target variable. Here +ve correlation coefficient says that that attribute will best help target variable.

	Year	Kilometers_Driven	Mileage	Engine_CC	Power_bhp	Seats	Price
Year	1.000000	-0.173048	0.321534	-0.051712	0.070520	0.015204	0.305327
Kilometers_Driven	-0.173048	1.000000	-0.065253	0.091029	0.024197	0.082782	-0.011493
Mileage	0.321534	-0.065253	1.000000	-0.588354	-0.444878	-0.299631	-0.306588
Engine_CC	-0.051712	0.091029	-0.588354	1.000000	0.840934	0.392982	0.657118
Power_bhp	0.070520	0.024197	-0.444878	0.840934	1.000000	0.109025	0.757711
Seats	0.015204	0.082782	-0.299631	0.392982	0.109025	1.000000	0.052811
Price	0.305327	-0.011493	-0.306588	0.657118	0.757711	0.052811	1.000000

This is one of the techniques to select your features. There is also another method called KBestFeature. This will give us a score and we can select K features from it. We implemented it and lets look into the list of the observations of the model we generated.

```
[ 230.60938695  624.25301842 4572.64572883 8111.59780132]
[[7.200e+04 2.660e+01 9.980e+02 5.816e+01]
 [4.100e+04 1.967e+01 1.582e+03 1.262e+02]
 [4.600e+04 1.820e+01 1.199e+03 8.870e+01]
 [8.700e+04 2.077e+01 1.248e+03 8.876e+01]
 [4.067e+04 1.520e+01 1.968e+03 1.408e+02]]
```

Kilometers_driven has 230.60 and Mileage has 624.25 score, Engine_CC is 4572.64 and Power_bhp is 8111.59; So the continuous variable that has a positive impact on the target variable.

Feature Transformation: Feature Transformation is nothing but obtaining new feature values from that of the existing feature. Our model is to predict the cost of used car. So, we focus more on the regression part. Including the categorical features in our dataset is no longer. We try to convert these categorical variables into the continuous variables.

We performed these in our project as well.

Code Snippet:

```
new_data_dummies = new_data.copy()
new_data_dummies = pd.get_dummies(new_data_dummies)
new_data_dummies.info()

display(new_data_dummies)
```

So, our new dataset will be new_data_dummies. We perform our validations and perform our regression models using this dataset only.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 60 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Year                      6019 non-null   int64
 1   Kilometers_Driven         6019 non-null   int64
 2   Mileage                   6019 non-null   float64
 3   Engine_CC                 6019 non-null   float64
 4   Power_bhp                 6019 non-null   float64
 5   Seats                     6019 non-null   float64
 6   Price                     6019 non-null   float64
 7   Name_Ambassador           6019 non-null   uint8
 8   Name_Audi                 6019 non-null   uint8
 9   Name_BMW                  6019 non-null   uint8
 10  Name_Bentley              6019 non-null   uint8
 11  Name_Chevrolet            6019 non-null   uint8
 12  Name_Datsun               6019 non-null   uint8
 13  Name_Fiat                 6019 non-null   uint8
 14  Name_Force                6019 non-null   uint8
 15  Name_Ford                 6019 non-null   uint8
 16  Name_Honda                6019 non-null   uint8
```

Above is a simple example of few attributes of our new dataset. The Name attribute changes its type from categorical to continuous. Performing feature selection on this dataset will give us an idea to choose the attributes we need to consider during the feature required.

Model Selection and Evaluation

Model Selection: There are two kinds of models we can choose depending upon the business problem. Theses models are of two types.

- Regression Models
- Classification Models

Since, we are performing prediction of the cost of the used car our models will be Regression Models.

We will be performing models like

- Information-based Learning (Decision Tree Model)
- Similarity-based Learning (Nearest Neighbor Model)

- Probability-based (Naïve Bayes Model)
- Error-based Learning (Linear Regression Model)

Evaluation Metrics

Since we are dealing with Regression models. We will be using metrics that regression models will best use. We will look into each of them as we build the model below.

1. Mean Squared Error.
2. Root-Mean-Squared-Error
3. R^2 or Coefficient of Determination.
4. Mean-Absolute-Error.
5. Adjusted R^2.

These Evaluation Metrics will take the predicted value and the original value. It will finally compute the error based on the original value.

Models

1. Linear Regression Model:

This simple linear regression calculator uses the least squares method to find the line of best fit for a set of paired data, allowing you to estimate the value of a dependent variable (*Y*) from a given independent variable (*X*).

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.metrics import confusion_matrix



#Preparing data for the Model
columns1 = new_data_dummies.columns
X_Linear =  new_data_dummies.loc[:, new_data_dummies.columns != 'Price']
y_Linear = new_data_dummies.iloc[:, 6].values

#Splitting the data as Training and Testing parts we take 20% for testing and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X_Linear, y_Linear, test_size = 0.2, random_state = 0)

#Building a Linear Regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)                    #Matrix of feature X_train and Predictor y_train

print('')
#Predicting for Testing Data
y_pred_Linear = regressor.predict(X_test)

print('')
#Displaying the Predicted  Values
array1 = np.concatenate((y_pred_Linear.reshape(len(y_pred),1), (y_test.reshape(len(y_test),1) )),1)
display(array1)
df_Linear = pd.DataFrame(array1,columns=['Predicted_Value','Original_Value'])
display(df_Linear.head(15))
print('')


print('')
print('Final Numbers from the Data')
print('Observations from Linear Regression')
print('')
print('Y intercept is :',regressor.intercept_)
# print('Coeffiecient is :', regressor.coef_)
zipped_coeff = zip(columns1,regressor.coef_)
print(zipped_coeff)                                #This will zip all the attributes with its respective Coeffiecents

print('')
print('Score is',round(regressor.score(X_test,y_test),2))
r2_score_error = r2_score(df_Linear['Predicted_Value'],df_Linear['Original_Value'])
print('Mean absolute error %:',metrics.mean_absolute_error(df_Linear['Predicted_Value'],df_Linear['Original_Value']))
print('Root mean quare error ',round(r2_score_error,2),'%')
accuracy = round(regressor.score(X_test,y_test),2)
print('Accuracy of the model ',accuracy)
print('Accuarcy % of the model ',accuracy*100,'%')
print('Explained Varience Score',round(metrics.explained_variance_score(y_test, y_pred_Linear),4))
```

Predicted vs Actual Results.

	Predicted_Value	Original_Value
0	7.404982	7.25
1	2.781703	4.25
2	4.039142	3.90
3	11.577341	8.41
4	13.936518	13.48
5	5.335121	4.25

Scores (Accuracy)

```
Score is 0.9
Mean absolute error %: 1.3845973940772283
Root mean quare error  0.88 %
Accuracy of the model  0.9
Accuarcy % of the model  90.0 %
Explained Varience Score 0.8991
```

Random forest is a Supervised Learning algorithm which uses ensemble learning method for classification and regression.

Random Forest Regression

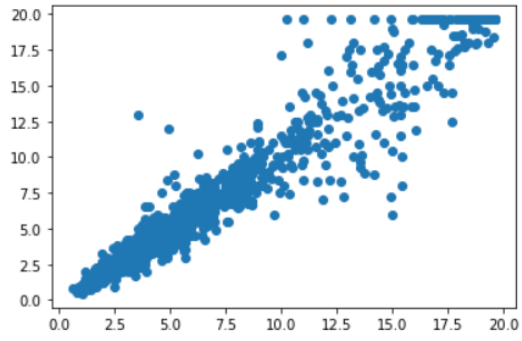
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

#Preparing data for the Model
X_Forest =  new_data_dummies.loc[:, new_data_dummies.columns != 'Price']
y_Forest = new_data_dummies.iloc[:, 6].values

#Splitting the data as Training and Testing parts we take 20% for testing and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X_Forest, y_Forest, test_size = 0.2, random_state = 0)

print('')
#Building a Linear Regression model
regressor_Forest = RandomForestRegressor()
regressor_Forest.fit(X_train, y_train)                    #Matrix of feature X_train and Predictor y_train

print('')
#Predicting for Testing Data
Predicted_Value_Forest = regressor_Forest.predict(X_test)

print('')
#Displaying the Predicted  Values
array1 = np.concatenate((y_pred.reshape(len(Predicted_Value_Forest),1), (y_test.reshape(len(y_test),1) )),1)
display(array1)
df_RnForest = pd.DataFrame(array1,columns=['Predicted_Value_Forest','Original_Value_Forest'])
display(df_RnForest.head(15))
print('Score is',round(regressor_DTree.score(X_test,y_test),4))

accuracy = round(regressor_Forest.score(X_test,y_test),4)
print(accuracy)
print('Accuarcy Percentage is :',accuracy*100,'%')

round(metrics.explained_variance_score(y_test, y_pred),4)
plt.scatter(df_RnForest['Predicted_Value_Forest'],df_RnForest['Original_Value_Forest'])

```

Results of the Random forest Regression are given below:

	Predicted_Value_Forest	Original_Value_Forest
0	5.645281	7.25
1	4.404418	4.25
2	5.097864	3.90
3	6.804852	8.41
4	7.144637	13.48
5	5.820676	4.25

Accuracy Results of Random Forest Regressor.

```
Score is 0.9466
0.9466
Accuarcy Percentage is : 94.66 %
Mean absolute error %: 0.7995263084691308
Root mean quare error  0.94 %
Accuracy of the model  0.9
```



Decision Tree Regressor

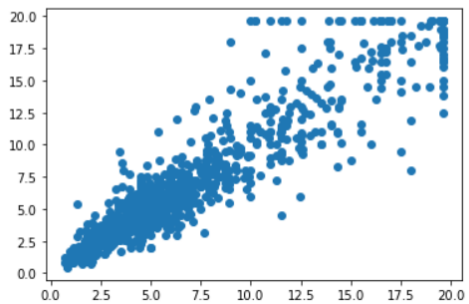
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics

X_Tree =  new_data_dummies.loc[:, new_data_dummies.columns != 'Price']
y_Tree = new_data_dummies.iloc[:, 6].values

X_train, X_test, y_train, y_test = train_test_split(X_Tree, y_Tree, test_size = 0.2, random_state = 0)

regressor_DTree = DecisionTreeRegressor()
regressor_DTree.fit(X_train, y_train)                    #Matrix of feature X_train and Predictor y_train
print(regressor_DTree)


y_pred_DTree = regressor_DTree.predict(X_test)
print(y_pred_DTree)
np.set_printoptions(precision = 2)
array1 = np.concatenate((y_pred.reshape(len(y_pred),1), (y_test.reshape(len(y_test),1) )),1)
df = pd.DataFrame(array1,columns=['Predicted_Value_DTree','Original_Value_DTree'])
display(df.head(15))
plt.scatter(df['Predicted_Value_DTree'],df['Original_Value_DTree'])


print('Score is',round(regressor_DTree.score(X_test,y_test),4))
accuracy = round(regressor_DTree.score(X_test,y_test),4)
print('Accuracy % is ',accuracy*100,'%')
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
[7.77 3.6  3.85 ... 8.25 5.84 2.65]
```

	Predicted_Value_DTree	Original_Value_DTree
0	5.645281	7.25
1	4.404418	4.25
2	5.097864	3.90
3	6.804852	8.41
4	7.144637	13.48
5	5.820676	4.25
6	6.689745	5.35
7	4.422426	1.67
8	5.563214	3.75

Accuracy of the model.

```
Score is 0.92
Accuracy % is  92.0 %
Mean absolute error %: 0.9759966777408675
Root mean quare error  0.92 %
Accuracy of the model  0.9
```



KNN Regression Model

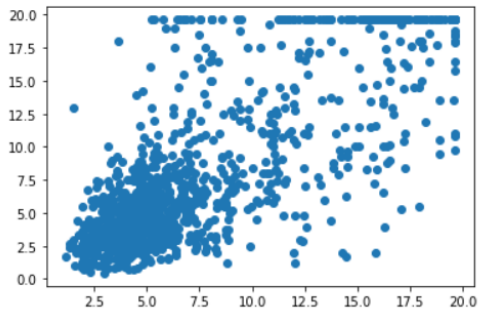
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn import metrics

X_KNN = new_data_dummies.loc[:, new_data_dummies.columns != 'Price']
y_KNN = new_data_dummies.iloc[:, 6].values

X_train, X_test, y_train, y_test = train_test_split(X_KNN, y_KNN, test_size = 0.2, random_state = 0)

regressor_KNN = KNeighborsRegressor(n_neighbors = 3)
regressor_KNN.fit(X_train, y_train)                    #Matrix of feature X_train and Predictor y_train
print(regressor_KNN)


y_pred = regressor_KNN.predict(X_test)
print(y_pred)
array1_regressor_KNN = np.concatenate((y_pred.reshape(len(y_pred),1), (y_test.reshape(len(y_test),1) )),1)
df = pd.DataFrame(array1_regressor_KNN,columns=['Predicted_Value_regressor_KNN','Original_Value_regressor_KNN'])
display(df.head(15))
plt.scatter(df['Predicted_Value_regressor_KNN'],df['Original_Value_regressor_KNN'])


print('Score is',round(regressor_KNN.score(X_test,y_test),4))
accuracy = round(regressor_KNN.score(X_test,y_test),4)
print('Accuracy % is ',accuracy*100,'%')
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                    weights='uniform')
[10.87  3.3   4.77 ...  4.88  8.74  3.31]
```

	Predicted_Value_regressor_KNN	Original_Value_regressor_KNN
0	10.873333	7.25
1	3.300000	4.25
2	4.766667	3.90
3	17.246667	8.41
4	15.600000	13.48
5	4.450000	4.25

Accuracy Percentage:

```
Score is 0.6381
Accuracy % is  63.81 %
Mean absolute error %: 2.4658333333333333
Root mean quare error  0.51 %
```



Support Vector Machine

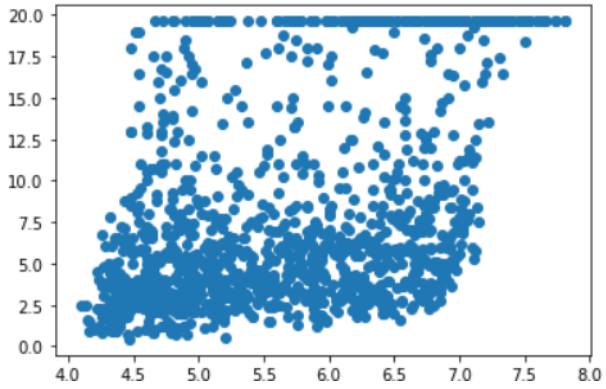
```
>>> from sklearn.svm import SVR

X_KNN = new_data_dummies.loc[:, new_data_dummies.columns != 'Price']
y_KNN = new_data_dummies.iloc[:, 6].values

X_train, X_test, y_train, y_test = train_test_split(X_KNN, y_KNN, test_size = 0.2, random_state = 0)

regressor_SVM = SVR(C=1.0, epsilon=0.2)
regressor_SVM.fit(X_train, y_train)                    #Matrix of feature X_train and Predictor y_train
print(regressor_SVM)

y_pred = regressor_SVM.predict(X_test)
print(y_pred)
array1_regressor_SVM = np.concatenate((y_pred.reshape(len(y_pred),1), (y_test.reshape(len(y_test),1) )),1)
array1_regressor_SVM_df = pd.DataFrame(array1_regressor_SVM,columns=['Predicted_Value_regressor_KNN','Original_Value_regr
display(df.head(15))
plt.scatter(array1_regressor_SVM_df['Predicted_Value_regressor_KNN'],array1_regressor_SVM_df['Original_Value_regressor_K

print('Score is',round(regressor_SVM.score(X_test,y_test),4))
accuracy = round(regressor_SVM.score(X_test,y_test),4)
print('Accuracy % is ',accuracy*100,'%')
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.2, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
[5.65 4.4  5.1  ... 6.33 5.54 5.4 ]
```

	Predicted_Value_regressor_svm	Original_Value_regressor_svm
0	5.645281	7.25
1	4.404418	4.25
2	5.097864	3.90
3	6.804852	8.41
4	7.144637	13.48
5	5.820676	4.25

Accuracy of the Model

```
Score is -0.0326
Accuracy % is  -3.26 %
Mean absolute error %: 4.177743357387335
Root mean quare error  -43.91 %
```



Sampling and Evaluation Settings

Data Sampling refers to obtaining a sample from the existing dataset. So, here we use Sampling techniques and select a subset of the data to be analyzed. It is difficult for us to carryout our analyzing our Analytics methods on the entire dataset. So, sampling helps us to partition our dataset so that we can consume our time for analyzing.

We have different methods like

1. **Random Sampling:** where we try to pick random data from the dataset. This includes selecting random selecting every class element in the dataset randomly in which it may or may not include every class tuple in the sample dataset.
2. **Stratified Sampling:** An equal number of objects are drawn from each group of different size.

Hyper-parameter Optimization

Lorem ipsum datum ….

Evaluation

We evaluate the model based on the accuracy score of the model. In this project we have performed the evaluation metrics to see how accurate the model is?

Support Vector Machine

```
Score is -0.0326
Accuracy % is  -3.26 %
Mean absolute error %: 4.177743357387335
Root mean quare error  -43.91 %
```

KNN Regression Model

```
Score is 0.6381
Accuracy % is  63.81 %
Mean absolute error %: 2.4658333333333333
Root mean quare error  0.51 %
```

Decision Tree Regression

```
Score is 0.92
Accuracy % is  92.0 %
Mean absolute error %: 0.9759966777408675
Root mean quare error  0.92 %
Accuracy of the model  0.9
```

Random Forest Regression

```
Score is 0.9466
0.9466
Accuarcy Percentage is : 94.66 %
Mean absolute error %: 0.7995263084691308
Root mean quare error  0.94 %
Accuracy of the model  0.9
```

Linear Regression Model

```
Score is 0.9
Mean absolute error %: 1.3845973940772283
Root mean quare error  0.88 %
Accuracy of the model  0.9
Accuarcy % of the model  90.0 %
Explained Varience Score 0.8991
```

Results and Conclusion:

We have successfully identified the results of our model above and it is the time to recommend a solution to the business. If we look at the system, we generated using the regression models we are taking into to the consideration all the attributes that are most important to the car price prediction. This will help us to incorporate the new car details making best use of all the data provided.

So, instead of continuously working on the car price prediction using the manual analysis is no longer useful and consume a lot amount of time. So, deploying this system into their business will help to reduce the time consumption and predict price of the car. The main advantage of our system is we deployed all the attributes that user needs to validate a car before buying it.

So, the best solution that we can recommend the business after interpreting the results is to deploy this solution to reduce the workflow and predict the price of the car accurately. We have got models that gave us 94% accuracy. Making the best use of this system to rum at the constant intervals of time will be a great addon to business. And when it comes to the User, the user can give his inputs to the system and check for how much can he sell his car. On the other hand, a user who wants to buy a car can gives his criterion of buying a car like power_bhp should be this etc..., Deploying this system will be best useful for both the company and the user.

This system will give the user and business a real-world idea about the used cars price.