Dr. B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY JALANDHAR

INDUSTRIAL TRAINING REPORT

TITLE: Used Car price prediction

INSTITUTE: Itronix Solutions, Jalandhar

SESSION 2017-2021

Submitted By:

Nimish Gupta

ROLL NO: 17104055

DISCIPLINE: ECE

ACKNOWLEDGEMENT

The training opportunity I had with Itronix Solutions, Jalandhar was a great chance for learning and logical development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this training period.
I take this opportunity to express a deep sense of gratitude to **Mr. Karan**, Trainer, Itronix Solutions (Jalandhar), for his cordial support, valuable information and guidance, which helped me in completing this task through various stages.
Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to the **Mr. Varun** of Itronix Solutions for his unfailing cooperation and sparing her valuable time to assist me in various training related issues throughout the training period.
It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to **Dr. R.K. Sunkaria**, Electronics and Communication Department (HOD) and all teachers of the department for their careful and precious guidance which were extremely valuable for my study both theoretically and practically.
Last but not the least I would like to mention here that I am greatly indebted to each and everybody who has been associated with my project at any stage but whose name does not find a place in this acknowledgement.

TABLE OF CONTENTS

S.NO.	CONTENTS	PG NO.

ABSTRACT

This project entitled as “Used Cars price prediction” is about predicting the price of used cars in Indian market. Driverless cars are getting closer to reality and at a faster pace than ever. But it is still a bit farfetched dream to have one in your garage. For the time being, there are still a lot of combustion and hybrid cars that roar around the road, for some its chills. Though the overall data on sales of automobiles shows a huge drop in sales in the last couple of years, cars are still a big attraction for many. Cars are more than just a utility for many. They are often the pride and status of the family. We all have different tastes when it comes to owning a car or at least when thinking of owning one.

Well here of course as the name suggests we are not concentrating on a new car, rather our interest is in knowing the prices of used cars across the country whether it is a royal l luxury sedan or a cheap budget utility vehicle. In this hackathon, you will be predicting the costs of used cars given the data collected from various sources and distributed across various locations in India.

Size of training set: 6,019 records

Size of test set: 1,234 records

FEATURES:

- **Name:** The brand and model of the car.

- **Location:** The location in which the car is being sold or is available for purchase.
- **Year:** The year or edition of the model.
- **Kilometers_Driven:** The total kilometres driven in the car by the previous owner(s) in KM.
- **Fuel_Type:** The type of fuel used by the car.
- **Transmission:** The type of transmission used by the car.
- **Owner_Type:** Whether the ownership is brand new, Second hand or other.
- **Mileage:** The standard mileage offered by the car company in kmpl or km/kg
- **Engine:** The displacement volume of the engine in cc.
- **Power:** The maximum power of the engine in bhp.
- **Seats:** The number of seats in the car.
- **New_Price:** The price of a new car of the same model.
- **Price:** The price of the used car in INR Lakhs.

INTRODUCTION TO MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

Some machine learning methods

Machine learning algorithms are often categorized as supervised or unsupervised.

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

- In contrast, **unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabelled data.

- **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labelled and unlabelled data for training – typically a small amount of labelled data and a large amount of unlabelled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labelled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabelled data generally doesn't require additional resources.

- **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

In the end we can say that **Machine Learning** is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that which makes it more similar to humans: *The ability to learn*. Machine learning is actively being used today, perhaps in many more places than one would expect.

TECHNOLOGY STACK

Theoretical Knowledge	Machine Learning models
Model used	Random Forest Regressor
Language used	Python
Libraries used	• NumPy • Pandas • Sklearn • Matplotlib • Time
Software used to develop	• Anaconda • Jupyter Notebook

WHY STARTING WITH PYTHON?

If your aim is growing into a successful coder, you need to know a lot of things. But, for Machine Learning & Data Science, it is pretty enough to master at least **one coding language** and use it confidently. So, calm down, you don't have to be a programming genius.

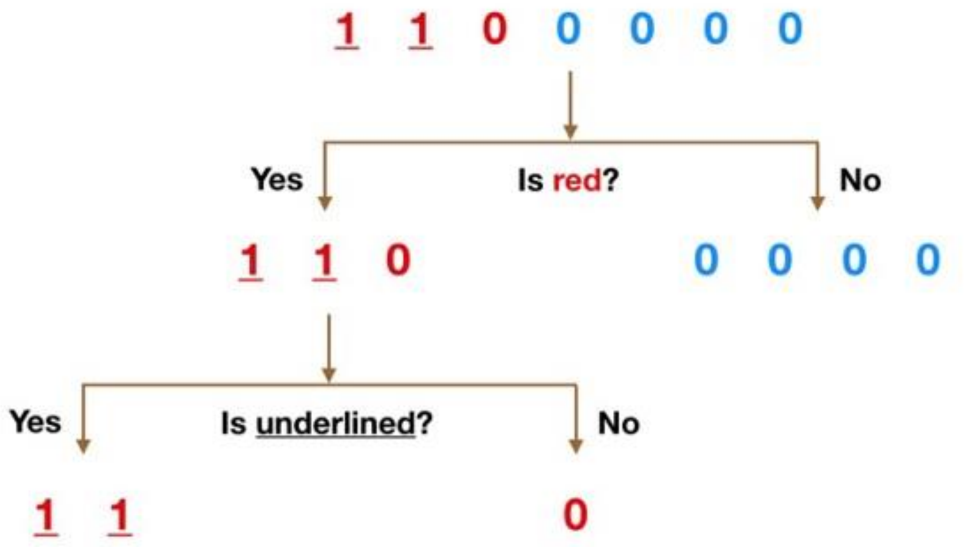
For successful Machine Learning journey, it's necessary to choose the appropriate coding language right from the beginning, as your choice will determine your future. On this step, you must think strategically and arranged correctly the priorities and don't spend time on unnecessary things.

My opinion — **Python** is a perfect choice for beginner to make your focus on in order to jump into the field of machine learning and data science. It is a minimalistic and intuitive language with a full-featured library line (also called frameworks) which significantly reduces the time required to get your first results.

You can also, by the way, consider the **R language**, but personally, I am more prone to **Python**.

Decision Trees

Let’s quickly go over decision trees as they are the building blocks of the random forest model. Fortunately, they are pretty intuitive. I’d be willing to bet that most people have used a decision tree, knowingly or not, at some point in their lives.



Simple Decision Tree Example

It’s probably much easier to understand how a decision tree works through an example.

Imagine that our dataset consists of the numbers at the top of the figure to the left. We have two 1s and five 0s (1s and 0s are our classes) and desire to separate the classes using their features. The features are color (red vs. blue) and whether the observation is underlined or not. So how can we do this?

Color seems like a pretty obvious feature to split by as all but one of the 0s are blue. So we can use the question, “Is it red?” to split our first node. You can think of a node in a tree as the point where the path splits into two — observations that meet the criteria go down the Yes branch and ones that don’t go down the No branch.

The No branch (the blues) is all 0s now so we are done there, but our Yes branch can still be split further. Now we can use the second feature and ask, “Is it underlined?” to make a second split.

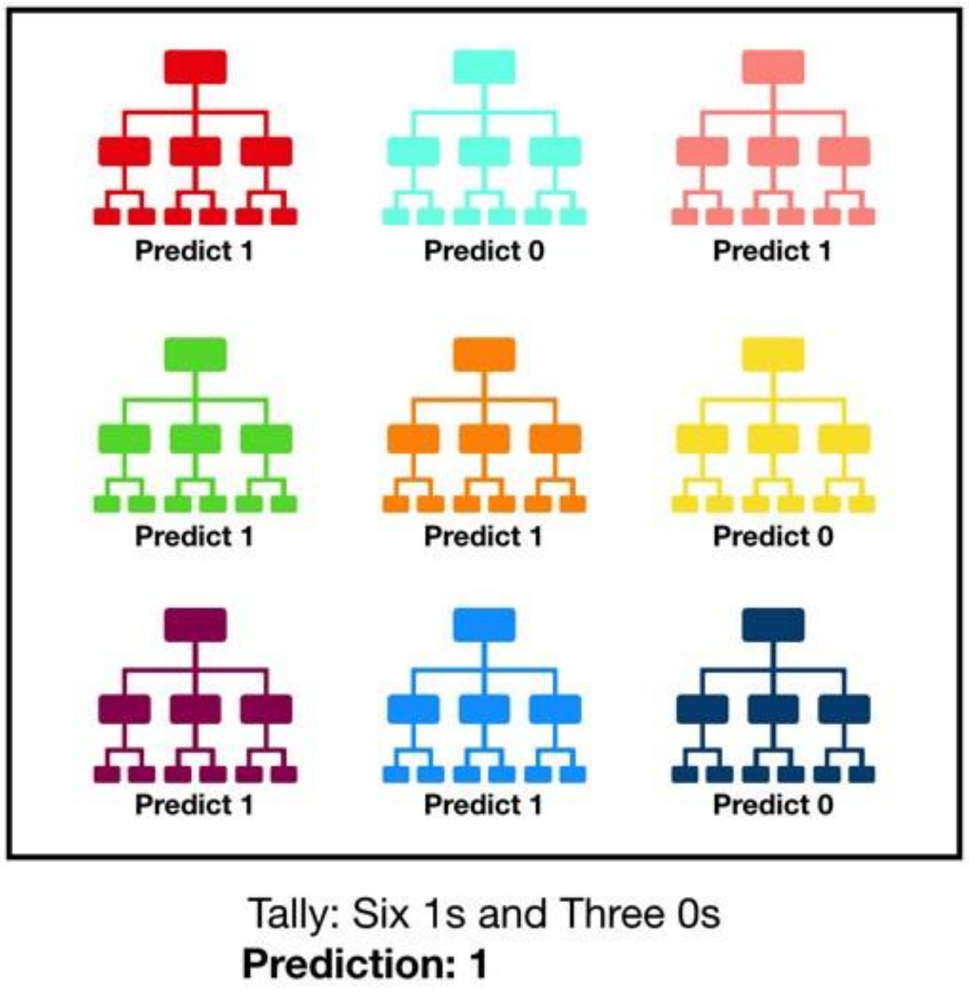
The two 1s that are underlined go down the Yes subbranch and the 0 that is not underlined goes down the right subbranch and we are all done. Our decision tree was able to use the two features to split up the data perfectly. Victory!

Obviously in real life our data will not be this clean but the logic that a decision tree employs remains the same. At each node, it will ask —

What feature will allow me to split the observations at hand in a way that the resulting groups are as different from each other as possible (and the members of each resulting subgroup are as similar to each other as possible)?

The Random Forest Regressor

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction (see figure below).



Visualization of a Random Forest Model Making a Prediction

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. **The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

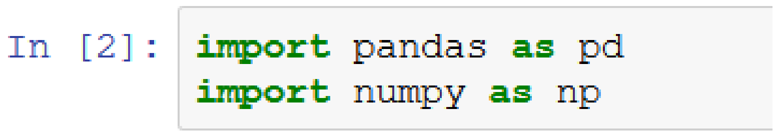
There needs to be some actual signal in our features so that models built using those features do better than random guessing.

The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

USED CARS PREDICTION

1. Importing the libraries:

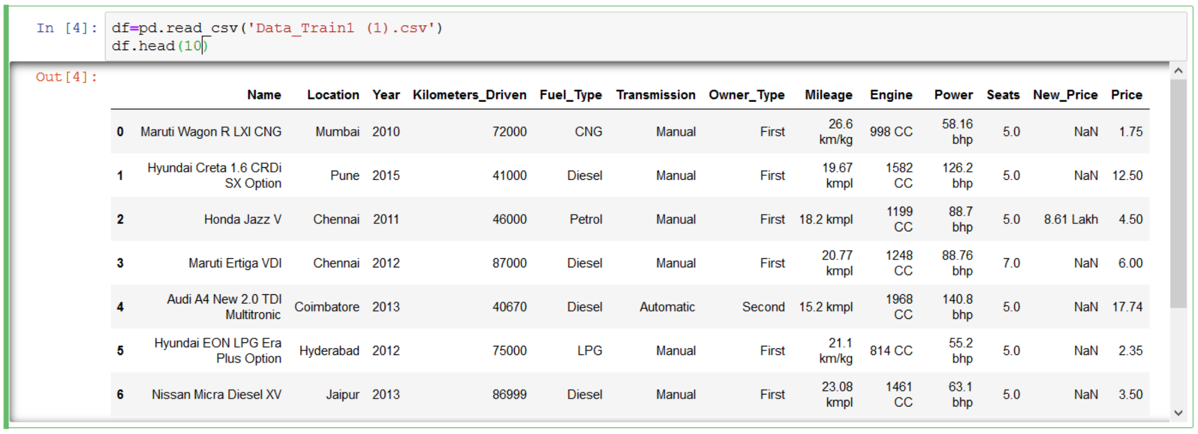
The libraries which are needed for the proper compilation of the program is imported. Here I have imported pandas and numpy for handling the dataframe and performing the calculations.



```
In [2]: import pandas as pd
        import numpy as np
```

2. Reading the data file:

The data file named Data_Train1 (1) is read using pandas library. The file comprises of various features like Name, Location, Year, Kilometer_Driven, Fuel_type,Transmission, Owner_type, Mileage, Engine, Power, Seats, New_Price, Price.



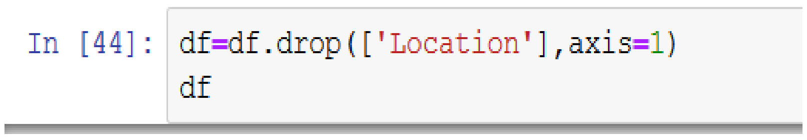
```
In [4]: df=pd.read_csv('Data_Train1 (1).csv')
        df.head(10)
```

Out[4]:

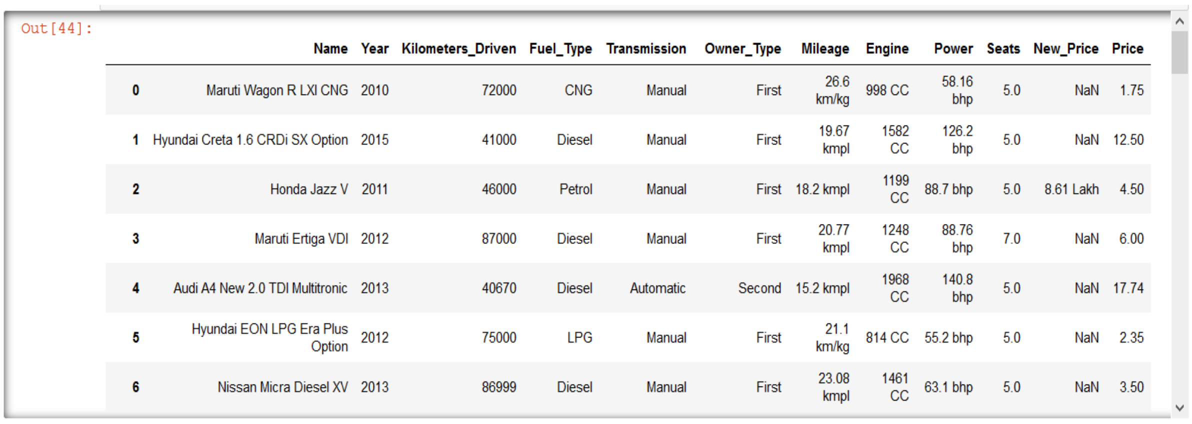
	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN	12.50
2	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	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00
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
5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	LPG	Manual	First	21.1 km/kg	814 CC	55.2 bhp	5.0	NaN	2.35
6	Nissan Micra Diesel XV	Jaipur	2013	86999	Diesel	Manual	First	23.08 kmpl	1461 CC	63.1 bhp	5.0	NaN	3.50

3. Dropping the Location column:

Since the location of the car will not help us in predicting the price of the car so it is better to drop or delete that column because unnecessary columns can mislead to the prediction capacity of the model.



```
In [44]: df=df.drop(['Location'],axis=1)
         df
```

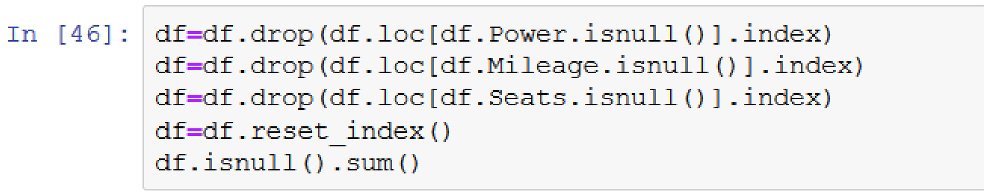


Out[44]:

	Name	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	Maruti Wagon R LXI CNG	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75
1	Hyundai Creta 1.6 CRDi SX Option	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN	12.50
2	Honda Jazz V	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.61 Lakh	4.50
3	Maruti Ertiga VDI	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00
4	Audi A4 New 2.0 TDI Multitronic	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	NaN	17.74
5	Hyundai EON LPG Era Plus Option	2012	75000	LPG	Manual	First	21.1 km/kg	814 CC	55.2 bhp	5.0	NaN	2.35
6	Nissan Micra Diesel XV	2013	86999	Diesel	Manual	First	23.08 kmpl	1461 CC	63.1 bhp	5.0	NaN	3.50

4. Removing all the rows with missing value in them:

I observed that the columns with NULL values are: Power, Mileage and Seats. So, all the rows with either of those rows null are removed and then column by column the number of null values a particular column holds is analysed using the method .isnull().sum().



```
In [46]: df=df.drop(df.loc[df.Power.isnull()].index)
         df=df.drop(df.loc[df.Mileage.isnull()].index)
         df=df.drop(df.loc[df.Seats.isnull()].index)
         df=df.reset_index()
         df.isnull().sum()
```

```
Out[46]: index                  0
         Name                   0
         Year                   0
         Kilometers_Driven      0
         Fuel_Type              0
         Transmission           0
         Owner_Type             0
         Mileage                0
         Engine                 0
         Power                  0
         Seats                  0
         New_Price           5152
         Price                  0
         dtype: int64
```

5. Label Encoding:

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.
So, here we first took the categorical variables into a different variable named as cat and then applied label encoding.

Separating categorical features

```
In [47]: cat=df.iloc[:,[1,2,4,5,6]].values
         cat
```

```
Out[47]: array([['Maruti', 2010, 'CNG', 'Manual', 'First'],
                ['Hyundai', 2015, 'Diesel', 'Manual', 'First'],
                ['Honda', 2011, 'Petrol', 'Manual', 'First'],
                ...,
                ['Mahindra', 2012, 'Diesel', 'Manual', 'Second'],
                ['Maruti', 2013, 'Petrol', 'Manual', 'First'],
                ['Chevrolet', 2011, 'Diesel', 'Manual', 'First']], dtype=object)
```

Applying Label Encoding:

```
In [48]: # Encoding the Independent Variable
         for i in range(5):
             from sklearn.preprocessing import LabelEncoder, OneHotEncoder
             labelencoder_X = LabelEncoder()
             cat[:,i] = labelencoder_X.fit_transform(cat[:,i])
         cat
```

```
Out[48]: array([[18, 12, 0, 1, 0],
                [10, 17, 1, 1, 0],
                [9, 13, 3, 1, 0],
                ...,
                [17, 14, 1, 1, 2],
                [18, 15, 3, 1, 0],
                [4, 13, 1, 1, 0]], dtype=object)
```

6. One Hot Encoding:

What is One Hot Encoding?

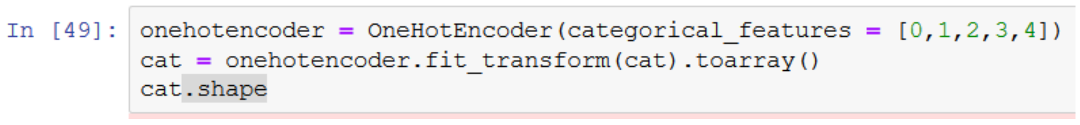
A one hot encoding is a representation of categorical variables as binary vectors.
This first requires that the categorical values be mapped to integer values.
Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.
Let’s make this concrete with a worked example.

Why One Hot Encoding?

Assume we have a sequence of labels with the values ‘red’ and ‘green’.
We can assign ‘red’ an integer value of 0 and ‘green’ the integer value of 1. As long as we always assign these numbers to these labels, this is called an integer encoding. Consistency is important so that we can invert the encoding later and get labels back from integer values, such as in the case of making a prediction.

Next, we can create a binary vector to represent each integer value. The vector will have a length of 2 for the 2 possible integer values.

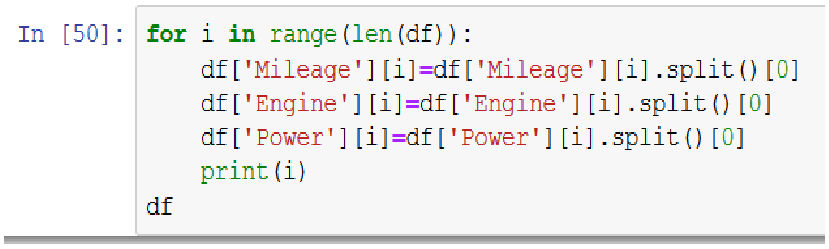
The ‘red’ label encoded as a 0 will be represented with a binary vector [1, 0] where the zeroth index is marked with a value of 1. In turn, the ‘green’ label encoded as a 1 will be represented with a binary vector [0, 1] where the first index is marked with a value of 1.



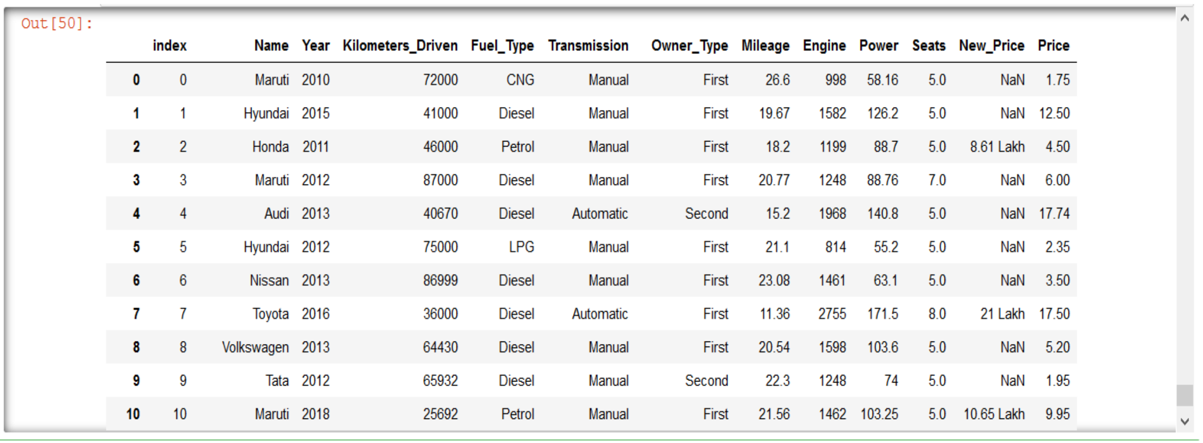
```
In [49]: onehotencoder = OneHotEncoder(categorical_features = [0,1,2,3,4])
         cat = onehotencoder.fit_transform(cat).toarray()
         cat.shape
```

7. Retrieving the Numerical data

The columns such as Mileage, Power and Engine have texts attached to the end of the numerical data. We know we cannot feed a string to our machine learning model. Also, they do not correspond to any category. Hence the best way to deal with them is to delete the string and to get only the numerical



```
In [50]: for i in range(len(df)):
             df['Mileage'][i]=df['Mileage'][i].split()[0]
             df['Engine'][i]=df['Engine'][i].split()[0]
             df['Power'][i]=df['Power'][i].split()[0]
             print(i)
         df
```

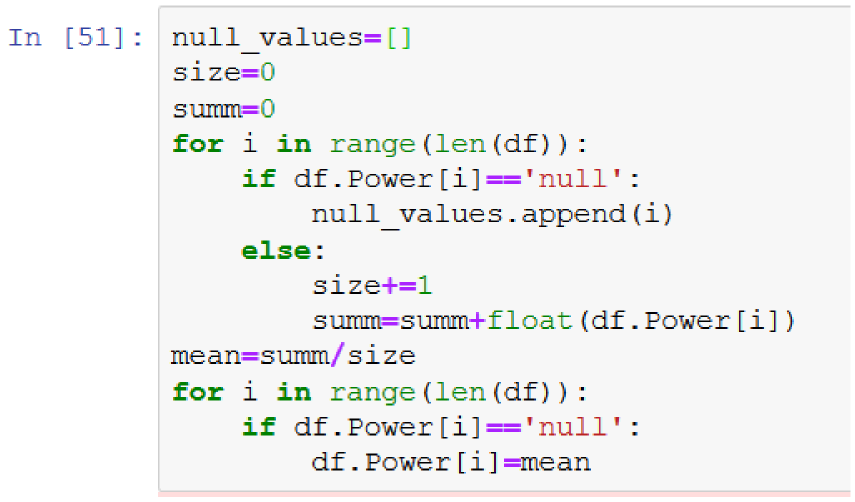


Out[50]:

	index	Name	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	0	Maruti	2010	72000	CNG	Manual	First	26.6	998	58.16	5.0	NaN	1.75
1	1	Hyundai	2015	41000	Diesel	Manual	First	19.67	1582	126.2	5.0	NaN	12.50
2	2	Honda	2011	46000	Petrol	Manual	First	18.2	1199	88.7	5.0	8.61 Lakh	4.50
3	3	Maruti	2012	87000	Diesel	Manual	First	20.77	1248	88.76	7.0	NaN	6.00
4	4	Audi	2013	40670	Diesel	Automatic	Second	15.2	1968	140.8	5.0	NaN	17.74
5	5	Hyundai	2012	75000	LPG	Manual	First	21.1	814	55.2	5.0	NaN	2.35
6	6	Nissan	2013	86999	Diesel	Manual	First	23.08	1461	63.1	5.0	NaN	3.50
7	7	Toyota	2016	36000	Diesel	Automatic	First	11.36	2755	171.5	8.0	21 Lakh	17.50
8	8	Volkswagen	2013	64430	Diesel	Manual	First	20.54	1598	103.6	5.0	NaN	5.20
9	9	Tata	2012	65932	Diesel	Manual	Second	22.3	1248	74	5.0	NaN	1.95
10	10	Maruti	2018	25692	Petrol	Manual	First	21.56	1462	103.25	5.0	10.65 Lakh	9.95

8. Replacing the null values with mean:

Some cell in the dataset were found to have null values in them so these values were replaced with the average of all the values found in the corresponding column. If we do not replace these null values with some other value or the these value rows are not deleted then, the model may fail or train or result into ambiguous results. So, removing the null values from the dataset is really necessary.



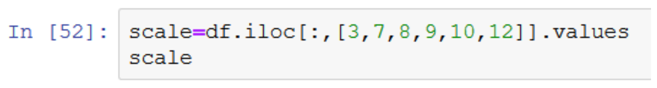
```
In [51]: null_values=[]
         size=0
         summ=0
         for i in range(len(df)):
             if df.Power[i]=='null':
                 null_values.append(i)
             else:
                 size+=1
                 summ=summ+float(df.Power[i])
         mean=summ/size
         for i in range(len(df)):
             if df.Power[i]=='null':
                 df.Power[i]=mean
```

9. Scaling of data:

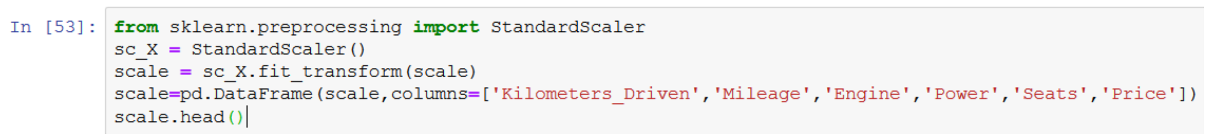
Most of the times, the dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Euclidian distance between two data points in their computations, this is a problem.

If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5kg and 5000gms. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes.

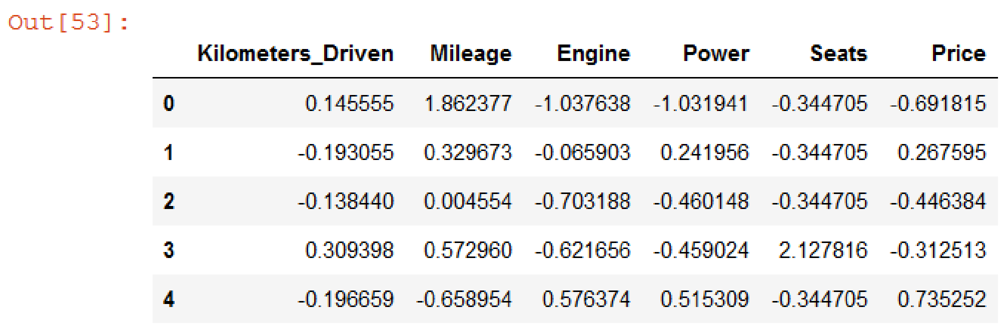
Here I have used Standardization to scale the data.



```
In [52]: scale=df.iloc[:,[3,7,8,9,10,12]].values
         scale
```



```
In [53]: from sklearn.preprocessing import StandardScaler
         sc_X = StandardScaler()
         scale = sc_X.fit_transform(scale)
         scale=pd.DataFrame(scale,columns=['Kilometers_Driven','Mileage','Engine','Power','Seats','Price'])
         scale.head()
```

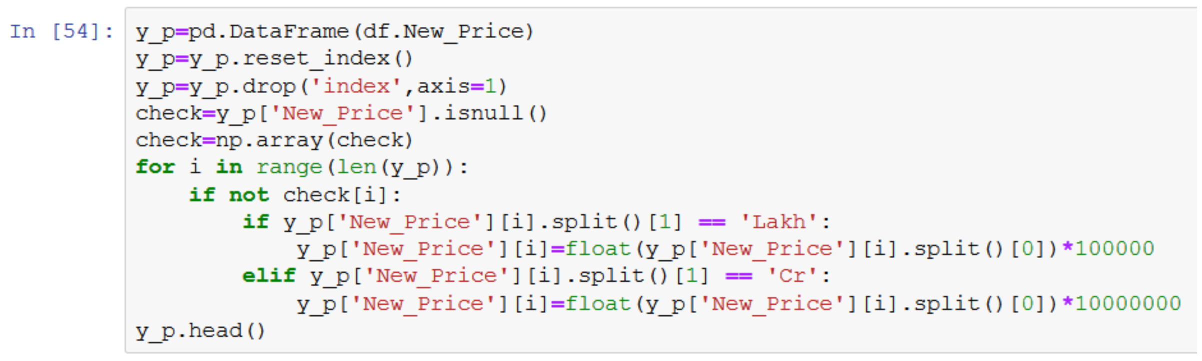


Out[53]:

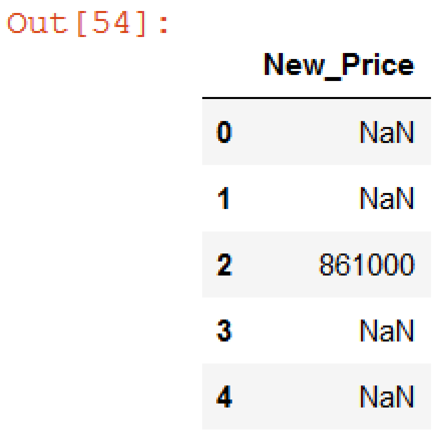
	Kilometers_Driven	Mileage	Engine	Power	Seats	Price
0	0.145555	1.862377	-1.037638	-1.031941	-0.344705	-0.691815
1	-0.193055	0.329673	-0.065903	0.241956	-0.344705	0.267595
2	-0.138440	0.004554	-0.703188	-0.460148	-0.344705	-0.446384
3	0.309398	0.572960	-0.621656	-0.459024	2.127816	-0.312513
4	-0.196659	-0.658954	0.576374	0.515309	-0.344705	0.735252

10.Cleaning the New_Price:

The New_Price column consist of the value in INR with numerical value as well as the string depicting the monetary value e.g. 7 Lakh. Here we cannot stem the word lakh because if we do so with 7 Crore then they both will depict same value and such inconsistency int the data can harm our model. So, wherever lakh is encountered the number is multiplied with 1,00,000 and wherever Crore is encountered the number is multiplied with 1,00,00,000.



```
In [54]: y_p=pd.DataFrame(df.New_Price)
         y_p=y_p.reset_index()
         y_p=y_p.drop('index',axis=1)
         check=y_p['New_Price'].isnull()
         check=np.array(check)
         for i in range(len(y_p)):
             if not check[i]:
                 if y_p['New_Price'][i].split()[1] == 'Lakh':
                     y_p['New_Price'][i]=float(y_p['New_Price'][i].split()[0])*100000
                 elif y_p['New_Price'][i].split()[1] == 'Cr':
                     y_p['New_Price'][i]=float(y_p['New_Price'][i].split()[0])*10000000
         y_p.head()
```

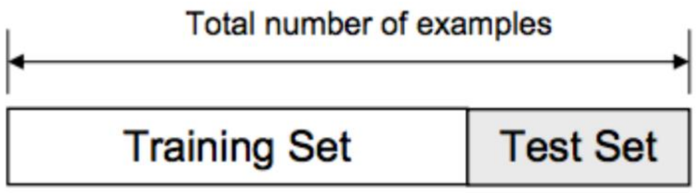


Out[54]:

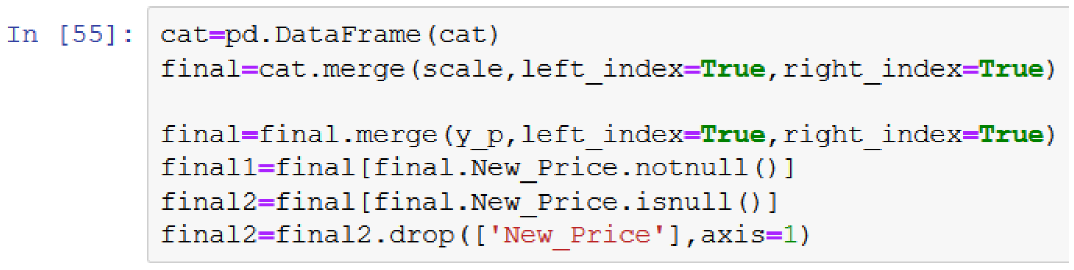
	New_Price
0	NaN
1	NaN
2	861000
3	NaN
4	NaN

11.Train Test Split:

As mentioned, in statistics and machine learning we usually split our data into two subsets: training data and testing data (and sometimes to three: train, validate and test), and fit our model on

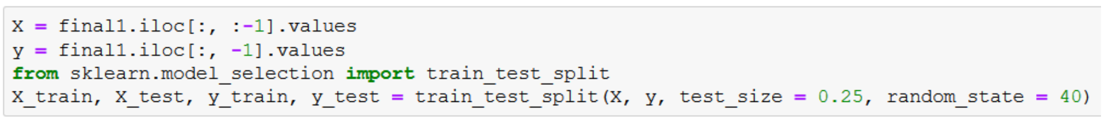


the train data, in order to make predictions on the test data. When we do that, one of two things might happen: we overfit our model or we underfit our model. We don't want any of these things to happen, because they affect the predictability of our model — we might be using a model that has lower accuracy and/or is ungeneralised (meaning you can't generalize your predictions on other data)The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.



```
In [55]: cat=pd.DataFrame(cat)
         final=cat.merge(scale,left_index=True,right_index=True)

         final=final.merge(y_p,left_index=True,right_index=True)
         final1=final[final.New_Price.notnull()]
         final2=final[final.New_Price.isnull()]
         final2=final2.drop(['New_Price'],axis=1)
```

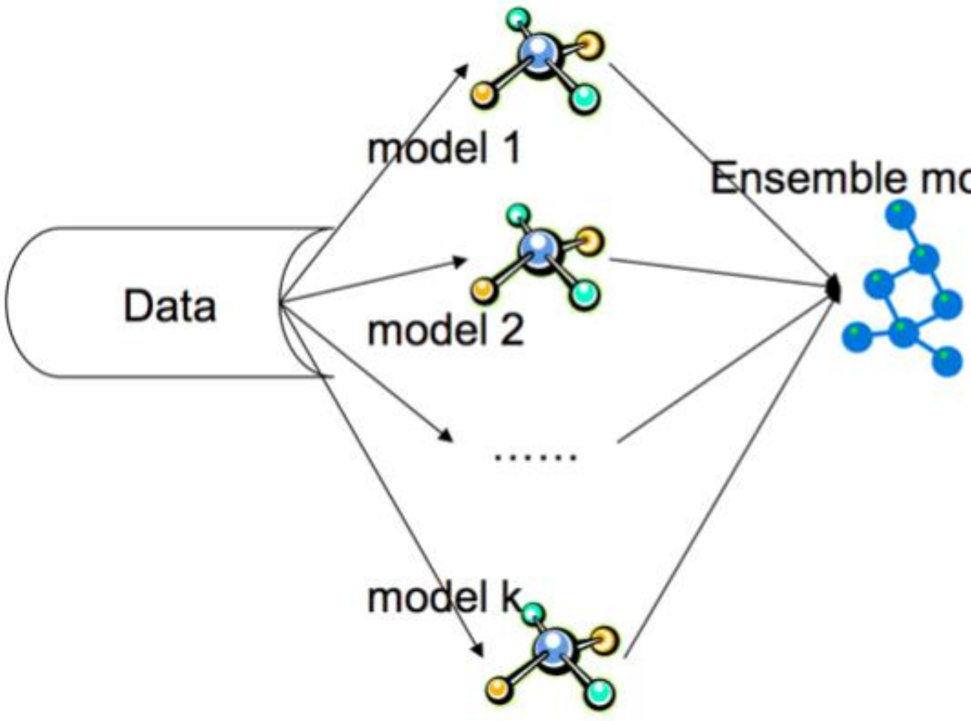


```
X = final1.iloc[:, :-1].values
y = final1.iloc[:, -1].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 40)
```

12.Import Random Forest Regressor

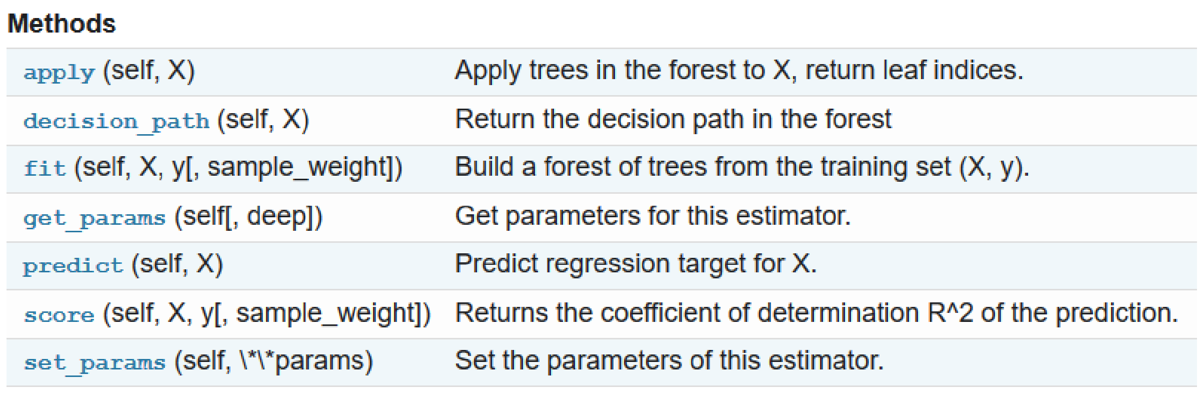
A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

The default values for the parameters controlling the size of the trees (e.g. max_depth, min_samples_leaf, etc.) lead to fully grown and unpruned trees which can potentially be very large on some data sets. To reduce memory consumption, the complexity and size of the trees should be controlled by setting those parameter values.



The features are always randomly permuted at each split. Therefore, the best found split may vary, even with the same training data, max_features=n_features and bootstrap=False, if the improvement of the criterion is identical for several splits enumerated during the search of the best split. To obtain a deterministic behaviour during fitting, random_state has to be fixed.

The default value max_features="auto" uses n_features rather than n_features / 3. The latter was originally suggested in [1], whereas the former was more recently justified empirically in [2].



Methods

`apply` (self, X)	Apply trees in the forest to X, return leaf indices.
`decision_path` (self, X)	Return the decision path in the forest
`fit` (self, X, y[, sample_weight])	Build a forest of trees from the training set (X, y).
`get_params` (self[, deep])	Get parameters for this estimator.
`predict` (self, X)	Predict regression target for X.
`score` (self, X, y[, sample_weight])	Returns the coefficient of determination R^2 of the prediction.
`set_params` (self, **params)	Set the parameters of this estimator.

```
from sklearn.ensemble import RandomForestRegressor
regressor1= RandomForestRegressor(n_estimators = 10, random_state = 2)
regressor1.fit(X, y)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', 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, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=2, verbose=0, warm_start=False)
```

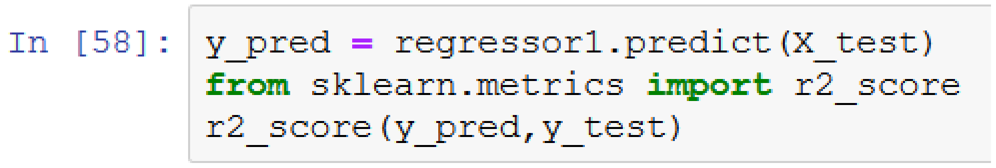
13.Prediction on Test set:

After training the model using Random Forest Regressor on training set, the values of New_Price are predicted on the test set to check how accurately the model is trained.

R^2 (coefficient of determination) regression score function.

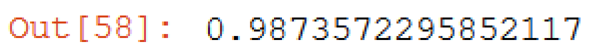
Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

[SYNTAX]:
sklearn.metrics.r2_score(*y_true*, *y_pred*, *sample_weight=None*, *multioutput = 'uniform_average'*)



```
In [58]: y_pred = regressor1.predict(X_test)
         from sklearn.metrics import r2_score
         r2_score(y_pred,y_test)
```

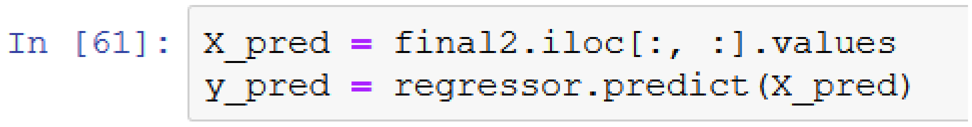
The R2 score obtained using the Random Forest trained model on training set and testing on the test set is :



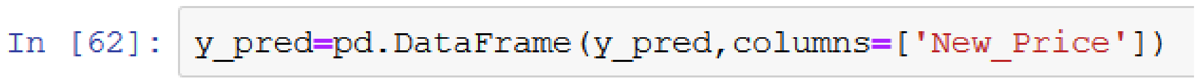
```
Out[58]: 0.9873572295852117
```

14.Getting dataset ready for prediction of Price:

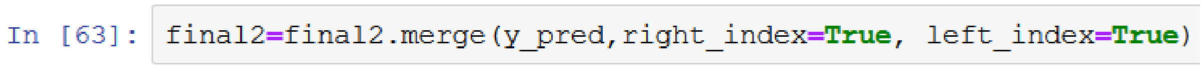
We have rather predicted the New_Price of the cars that is where the values of New_Price were missing and now all the fields except the Price can be used to train the model to predict the Price of the car to be sold. Also, while we have predicted the New_Price the data is scattered so now it must be concatenated properly to get the accurate predictions.



```
In [61]: X_pred = final2.iloc[:, :].values
         y_pred = regressor.predict(X_pred)
```

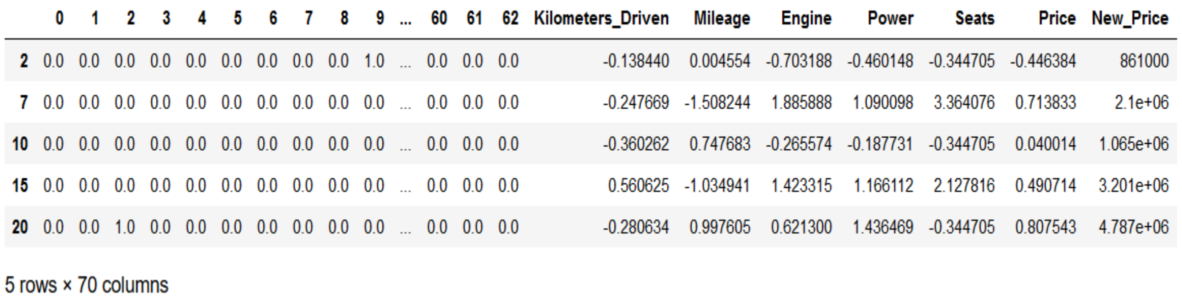


```
In [62]: y_pred=pd.DataFrame(y_pred,columns=['New_Price'])
```



```
In [63]: final2=final2.merge(y_pred,right_index=True, left_index=True)
In [64]: dataset=pd.concat([final1,final2])
```

```
In [65]: dataset.head()
```



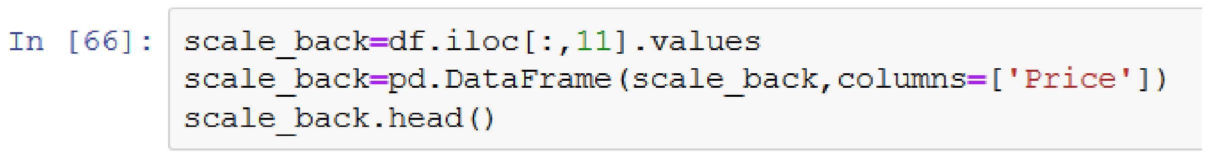
```
      0    1    2    3    4    5    6    7    8    9  ...   60   61   62  Kilometers_Driven   Mileage    Engine     Power     Seats     Price  New_Price
 2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  ...  0.0  0.0  0.0          -0.138440  0.004554 -0.703188 -0.460148 -0.344705 -0.446384     861000
 7  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0          -0.247669 -1.508244  1.885888  1.090098  3.364076  0.713833     2.1e+06
10  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0          -0.360262  0.747683 -0.265574 -0.187731 -0.344705  0.040014   1.065e+06
15  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0           0.560625 -1.034941  1.423315  1.166112  2.127816  0.490714   3.201e+06
20  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0          -0.280634  0.997605  0.621300  1.436469 -0.344705  0.807543   4.787e+06

5 rows × 70 columns
```

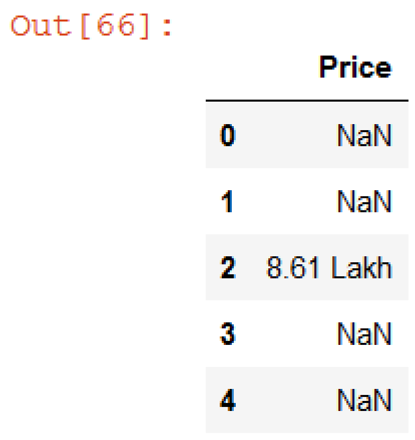
15.Un-scaling the features of dataset

Since, we scaled the data and that included the Price too, so now we need to scale back the price because the price is what we need to predict unscaled. The best way to do this is use inbuilt function of Sklearn but here I have found the desired column of Price from the dataset.

For the above purpose we will use the below written code to un-scale the required data.

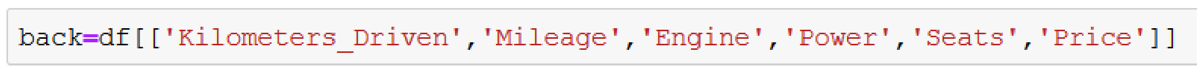


```
In [66]: scale_back=df.iloc[:,11].values
         scale_back=pd.DataFrame(scale_back,columns=['Price'])
         scale_back.head()
```

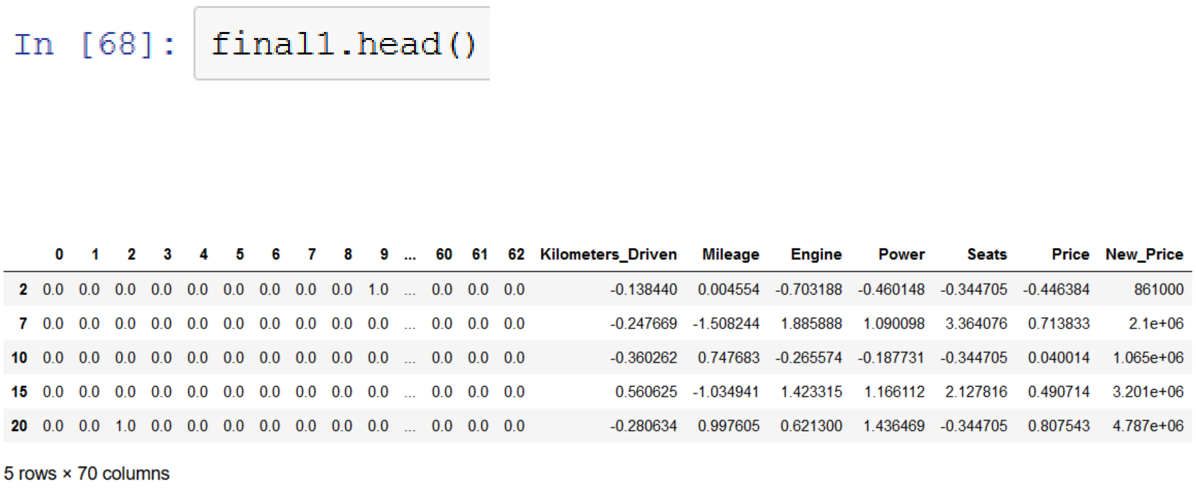


Out[66]:

	Price
0	NaN
1	NaN
2	8.61 Lakh
3	NaN
4	NaN



```
back=df[['Kilometers_Driven','Mileage','Engine','Power','Seats','Price']]
```

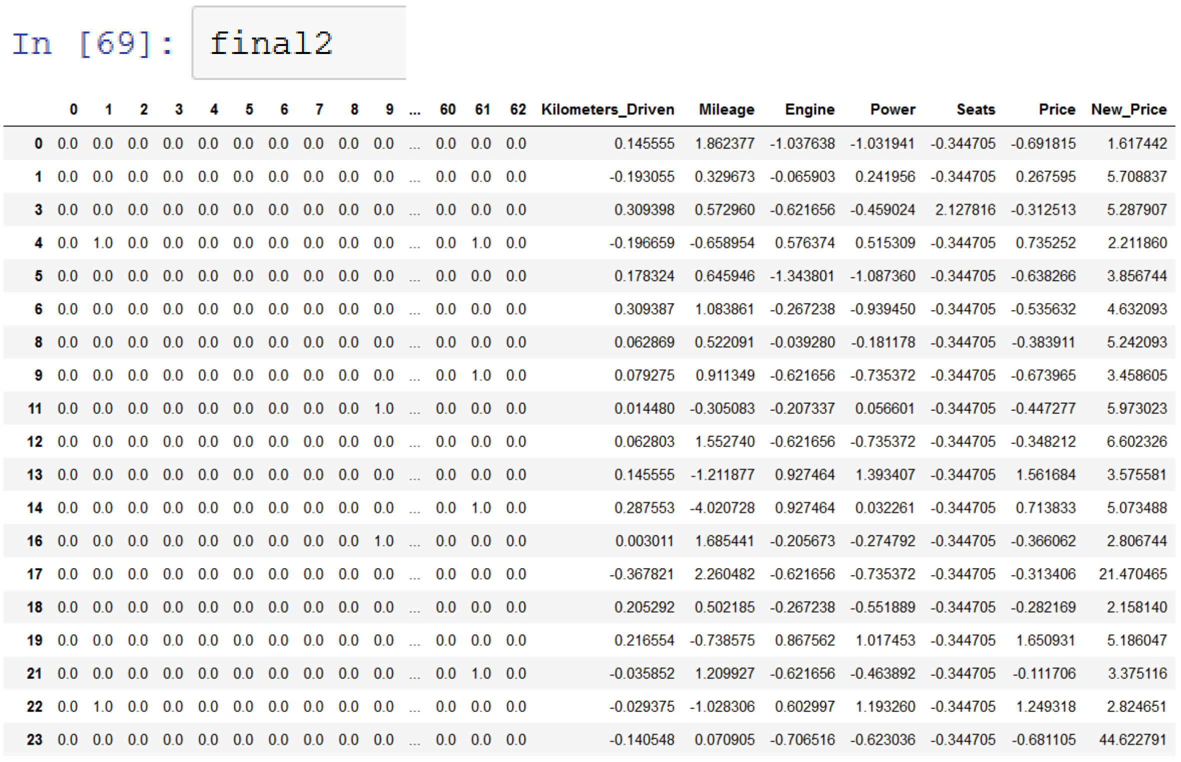


```
In [68]: final1.head()
```

	0	1	2	3	4	5	6	7	8	9	...	60	61	62	Kilometers_Driven	Mileage	Engine	Power	Seats	Price	New_Price
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	-0.138440	0.004554	-0.703188	-0.460148	-0.344705	-0.446384	861000
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	-0.247669	-1.508244	1.885888	1.090098	3.364076	0.713833	2.1e+06
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	-0.360262	0.747683	-0.265574	-0.187731	-0.344705	0.040014	1.065e+06
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.560625	-1.034941	1.423315	1.166112	2.127816	0.490714	3.201e+06
20	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	-0.280634	0.997605	0.621300	1.436469	-0.344705	0.807543	4.787e+06

5 rows × 70 columns

Now, we have got the columns where the New_Price was previously available. The next thing to do is to complete the New_Price column using the Random Forest Regressor we trained previously.

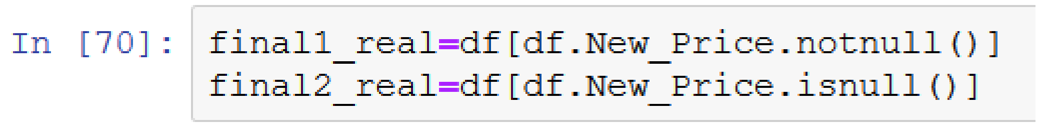


```
In [69]: final2
```

	0	1	2	3	4	5	6	7	8	9	...	60	61	62	Kilometers_Driven	Mileage	Engine	Power	Seats	Price	New_Price
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.145555	1.862377	-1.037638	-1.031941	-0.344705	-0.691815	1.617442
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	-0.193055	0.329673	-0.065903	0.241956	-0.344705	0.267595	5.708837
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.309398	0.572960	-0.621656	-0.459024	2.127816	-0.312513	5.287907
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	-0.196659	-0.658954	0.576374	0.515309	-0.344705	0.735252	2.211860
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.178324	0.645946	-1.343801	-1.087360	-0.344705	-0.638266	3.856744
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.309387	1.083861	-0.267238	-0.939450	-0.344705	-0.535632	4.632093
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.062869	0.522091	-0.039280	-0.181178	-0.344705	-0.383911	5.242093
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.079275	0.911349	-0.621656	-0.735372	-0.344705	-0.673965	3.458605
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.014480	-0.305083	-0.207337	0.056601	-0.344705	-0.447277	5.973023
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.062803	1.552740	-0.621656	-0.735372	-0.344705	-0.348212	6.602326
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.145555	-1.211877	0.927464	1.393407	-0.344705	1.561684	3.575581
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.287553	-4.020728	0.927464	0.032261	-0.344705	0.713833	5.073488
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.003011	1.685441	-0.205673	-0.274792	-0.344705	-0.366062	2.806744
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	-0.367821	2.260482	-0.621656	-0.735372	-0.344705	-0.313406	21.470465
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.205292	0.502185	-0.267238	-0.551889	-0.344705	-0.282169	2.158140
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.216554	-0.738575	0.867562	1.017453	-0.344705	1.650931	5.186047
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	-0.035852	1.209927	-0.621656	-0.463892	-0.344705	-0.111706	3.375116
22	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	-0.029375	-1.028306	0.602997	1.193260	-0.344705	1.249318	2.824651
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	-0.140548	0.070905	-0.706516	-0.623036	-0.344705	-0.681105	44.622791

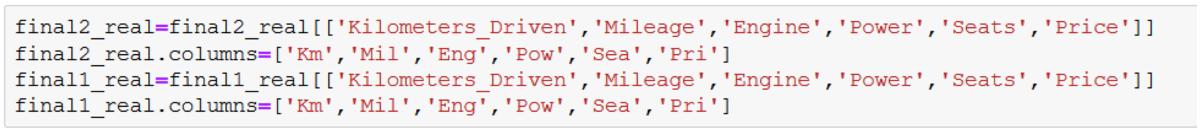
16.Getting the values of missing values in New_Price:

Here we will use the previously trained Random Forest Regressor to predict the missing values of New_Price.

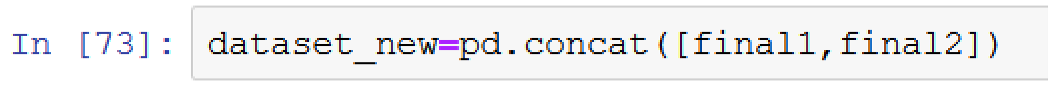


```
In [70]: final1_real=df[df.New_Price.notnull()]
         final2_real=df[df.New_Price.isnull()]
```

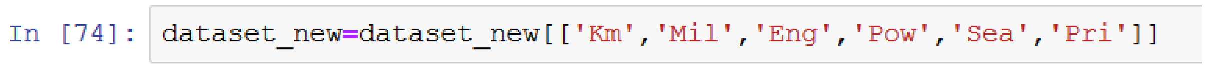
Here, final1 depicts the DataFrame which previously had the New_Price values and final2 is the DataFrame in which we have predicted the value using the previously trained Random Forest Regressor.



```
final2_real=final2_real[['Kilometers_Driven','Mileage','Engine','Power','Seats','Price']]
final2_real.columns=['Km','Mil','Eng','Pow','Sea','Pri']
final1_real=final1_real[['Kilometers_Driven','Mileage','Engine','Power','Seats','Price']]
final1_real.columns=['Km','Mil','Eng','Pow','Sea','Pri']
```



```
In [73]: dataset_new=pd.concat([final1,final2])
```



```
In [74]: dataset_new=dataset_new[['Km','Mil','Eng','Pow','Sea','Pri']]
In [77]: dataset=dataset.reset_index()
         dataset=dataset.drop(['index'],axis=1)
         dataset.head()
```

```
     0    1    2    3    4    5    6    7    8    9  ...   60   61   62  New_Price      Km    Mil   Eng     Pow  Sea    Pri
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  ...  0.0  0.0  0.0     861000   46000   18.2  1199    88.7  5.0   4.50
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0     2.1e+06  36000  11.36  2755   171.5  8.0  17.50
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0   1.065e+06  25692  21.56  1462  103.25  5.0   9.95
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0   3.201e+06 110000   13.5  2477  175.56  7.0  15.00
4  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0   4.787e+06  32982  22.69  1995     190  5.0  18.55

5 rows × 70 columns
```

Here we have got the final dataset on which we will train our model.

17.Training the final dataset:

I. Scaling:

Selecting the numerical features

```
In [78]: scale1=dataset.iloc[:,-7:-1].values
         scale1
```

Response

```
array([[861000.0, 46000, '18.2', '1199', '88.7', 5.0],
       [2100000.0, 36000, '11.36', '2755', '171.5', 8.0],
       [1065000.0, 25692, '21.56', '1462', '103.25', 5.0],
       ...,
       [55.14232558139535, 9000, '12.62', '2198', '158', 7.0],
       [3.103720930232558, 50121, '20.4', '1197', '83.11', 5.0],
       [2.554651162790698, 79000, '24.0', '1120', '70', 5.0]],
      dtype=object)
```

Scaling the data using Standard Scaler

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
scale1 = sc_X.fit_transform(scale1)
scale1=pd.DataFrame(scale1,columns=['New_Price1','Kilometers_Driven','Mileage','Engine','Power','Seats'])
scale1.head()
```

	New_Price1	Kilometers_Driven	Mileage	Engine	Power	Seats
0	0.377043	-0.129227	-0.002075	-0.703929	-0.463574	-0.346495
1	1.308746	-0.232817	-1.521134	1.918502	1.108299	3.379856
2	0.530447	-0.339597	0.744129	-0.260678	-0.187358	-0.346495
3	2.136676	0.533745	-1.045873	1.449970	1.185374	2.137739
4	3.329317	-0.264080	0.995085	0.637623	1.459502	-0.346495

II. Merging the scaled data with categorical features

```
dataset_scaled=dataset.merge(scale1,right_index=True,left_index=True)
dataset_scaled=dataset_scaled.drop(['Km','Eng','Mil','Sea','New_Price','Pow'],axis=1)
```

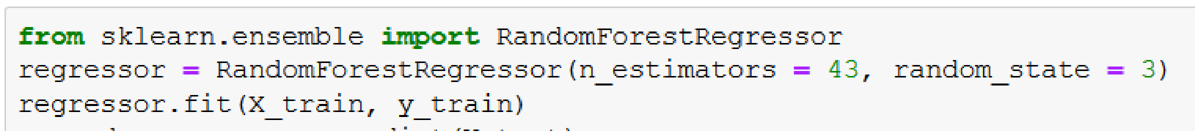
III. Concatenating original price and the dataset

```
price=dataset_scaled[['Pri']]
dataset_scaled=dataset_scaled.drop(['Pri'],axis=1)
dataset_scaled=dataset_scaled.merge(price,right_index=True,left_index=True)
```

IV. Performing train test split

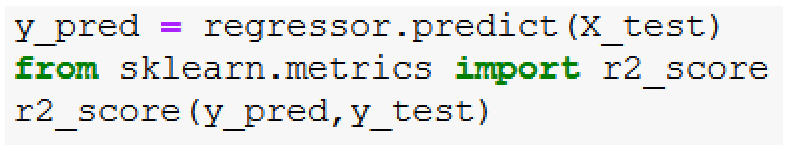
```
X = dataset_scaled.iloc[:, :-1].values
y = dataset_scaled.iloc[:, -1].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 6)
```

V. Training the Random Forest Regressor



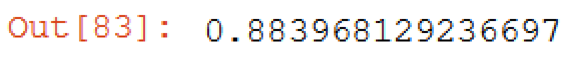
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 43, random_state = 3)
regressor.fit(X_train, y_train)
```

VI. Predicting the r2 score



```
y_pred = regressor.predict(X_test)
from sklearn.metrics import r2_score
r2_score(y_pred,y_test)
```

VII. Final r2 score



```
Out[83]:  0.883968129236697
```

CONCLUSION

As the modern organizations are automated and computers are working as per the instructions, it becomes essential for the coordination of human beings, commodity and computers in a modern organization. This information helps the authorities to perform their duties efficiently and effectively.

The project was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project.

- This prediction of used car prediction will help in selling of the used car at better prices
- Empirical results have shown that the market value of a used car depends on a variety of factors, and that forecast accuracy benefits from incorporating these factors into resale price prediction models
- Some of these studies employed standard techniques to account for nonlinearity such as the log-transformation
- The System has adequate scope for modification in future if it is necessary.
- It effectively overcomes the casualties of selling car at a lower price

BIBLIOGRAPHY

The following books were referred during the analysis and execution phase of the project

1. The Hundred-page Machine Learning book (Andriy Burkov)
2. The elements of Statistical Learning

WEBSITES:

1. www.tutorialpoint.com
2. www.w3schools.com
3. www.analyticsvidya.com