Car Resale Value Prediction using Machine Learning Techniques

CONTENTS

1. INTRODUCTION

1.1 OVERVIEW

The prices of the new manufactured cars in the automobile industry by the manufacturer with some extra cost which are incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money which they are investing to be worthy. But due to the increased price of new cars and the problems of customers to buy new cars due to improper financial status, used car sales are on increase in demand.

Due to the high raise in demand for the used cars, there is a need for a car resale price prediction system to effectively determine the quality and worth of the car using different features, which are very much required while assessing the price of resale car.

1.2 PURPOSE

Client will be able to efficiently determine the price that the chosen resale vehicle must cost. So, that people can invest money according to their preferences.

In this project, we suggest a solution using Machine Learning algorithms implemented in JAVA.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

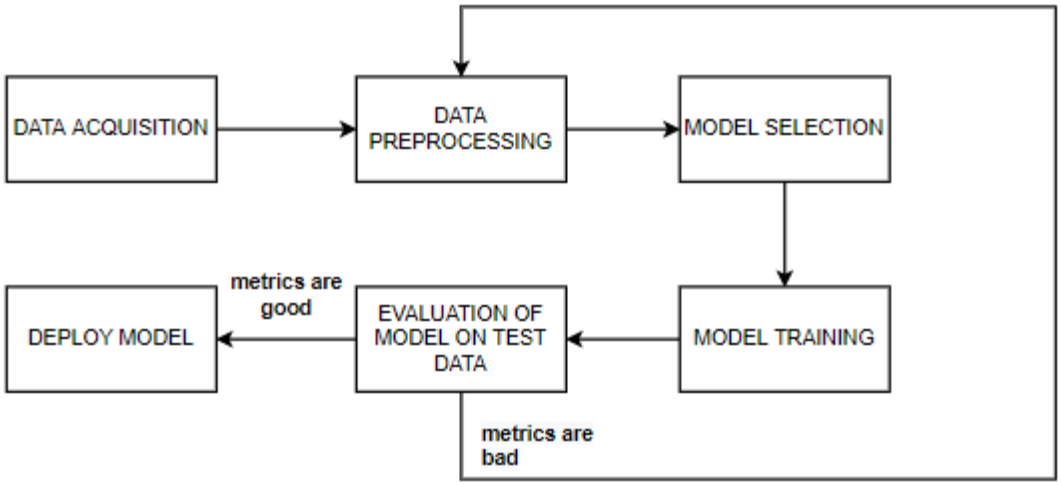
Existing models use Linear Regression algorithm, Decision Tree algorithms as the basic models. Problem with these models is that, it takes a huge amount of time for training the model, and if any attributes are categorical, proper encoding techniques must be used, and yet due to high correlated data, this technique produces mid-level percentage of accuracy (70 - 80%), and if over fitting took place if we tried to remove the duplicate values.

2.2 PROPOSED SOLUTION

Proposed model is built on Random Forest regressor algorithm, instead of using conventional Linear Regression, and Decision Trees models. This model takes less amount of training time, and the correlation coefficient is high, and low error metrics, thus making suitable for modelling.

3. THEORITICAL ANALYISIS

3.1 BLOCK DIAGRAM



3.2 SOFTWARE DESIGNING

1. JAVA 16.0
2. WEKA 3.8
3. MAVEN
4. ECLIPSE IDE

4. EXPERIMENTAL INVESTIGATIONS

There are separate datasets, one for the training, and other for testing the built model.

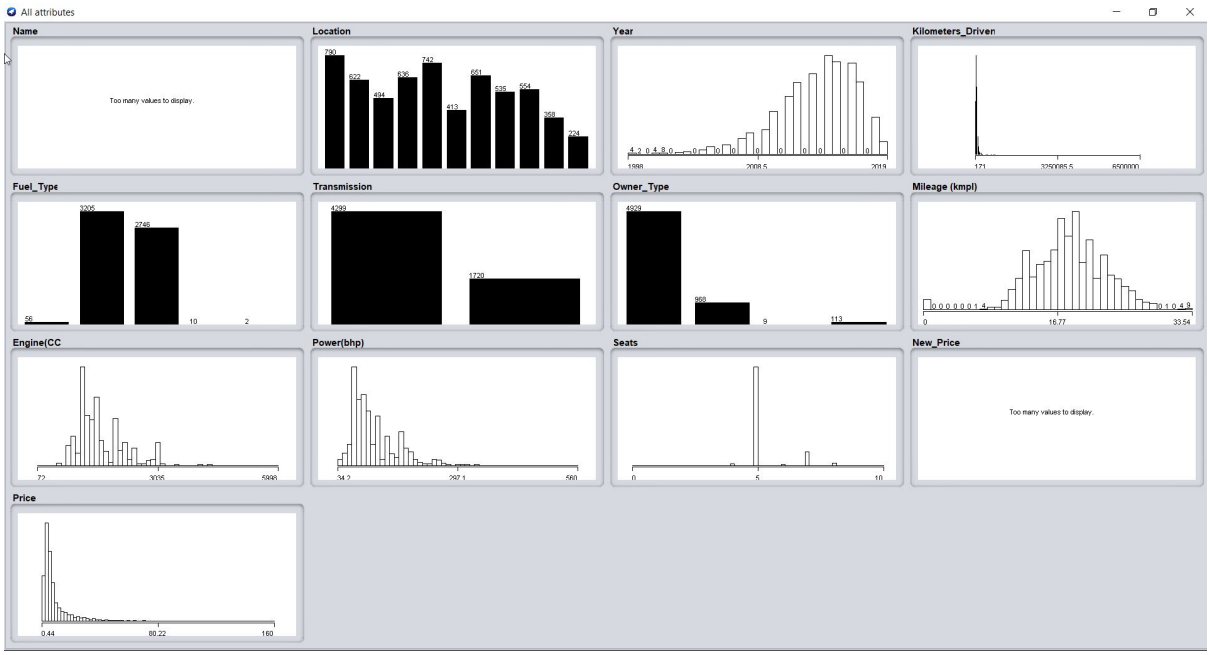
TRAINING DATASET:

Training data consist of 12 independent variables and 1 dependent variable. Many attributes consisted of much missing data. The following table depicts the details of train data and test data.

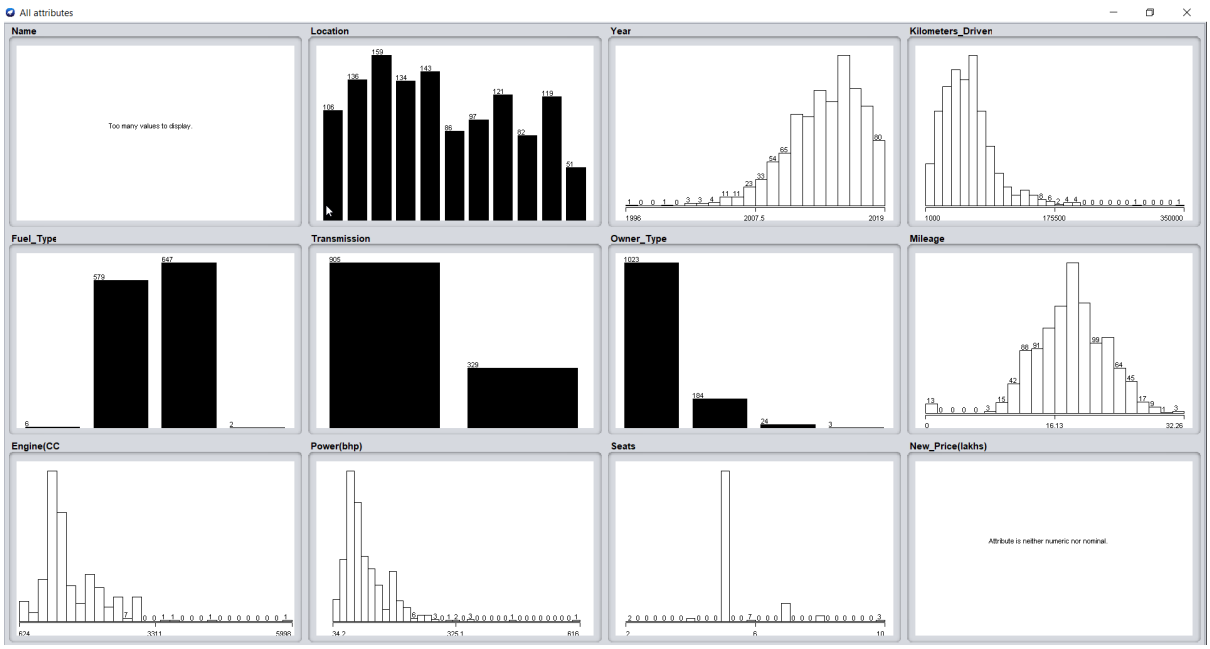
Train Data		
S.no	Attribute	Missing data
1	Name	0 (0%)
2	Location	0 (0%)
3	Year	0 (0%)
4	Kilometers_Driven	0 (0%)
5	Fuel_Type	0 (0%)
6	Transmission	0 (0%)
7	Owner_Type	0 (0%)
8	Mileage (kmpl)	0 (0%)
9	Engine (CC)	36 (1%)
10	Power (bhp)	143 (2%)
11	Seats	42 (1%)
12	New_Price	5192 (86%)
13	Price	0 (0%)

Test Data		
S.no	Attribute	Missing data
1	Name	0 (0%)
2	Location	0 (0%)
3	Year	0 (0%)
4	Kilometers_Driven	0 (0%)
5	Fuel_Type	0 (0%)
6	Transmission	0 (0%)
7	Owner_Type	0 (0%)
8	Mileage (kmpl)	0 (0%)
9	Engine (CC)	10 (1%)
10	Power (bhp)	32 (3%)
11	Seats	11 (1%)
12	New_Price	1052 (85%)

There are total of 6020 rows and 13 columns in the training dataset.



There are total of 1235 rows and 12 columns in the testing dataset.



Meaning of each attribute is shown below.

Attribute	Description

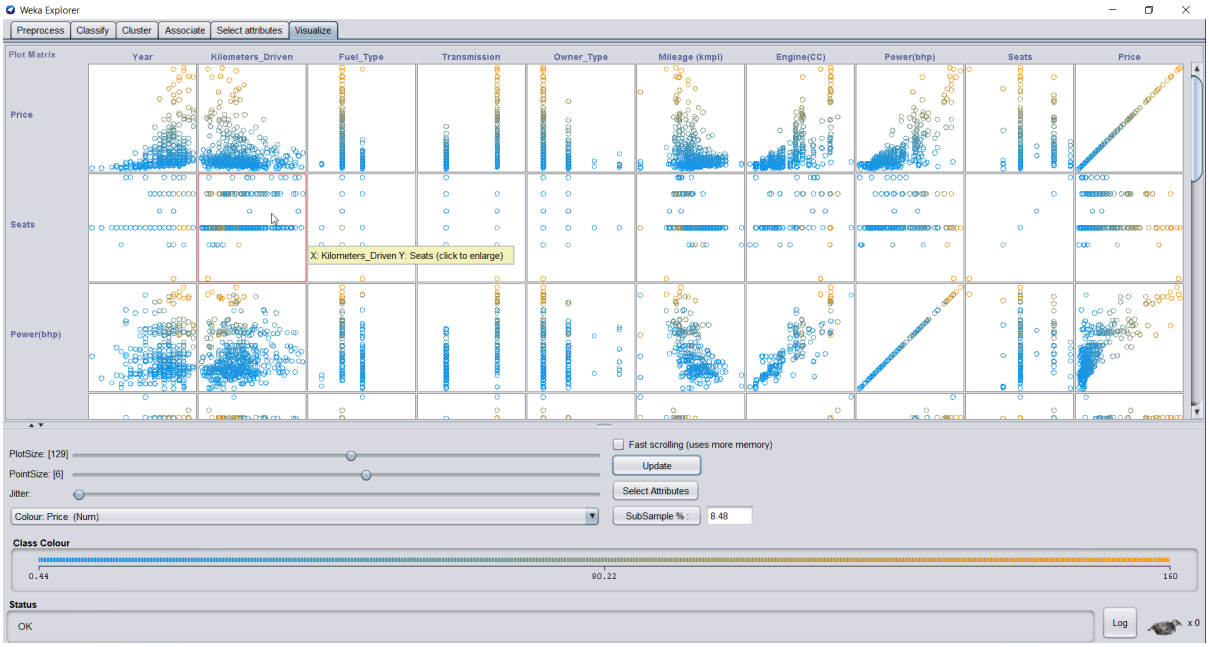
Name	The brand and model of the car
Location	The location in which the car is being sold or is available for purchase
Year	The year or edition of the model
Kilometers_Driven	The total kilometers driven in the car by the previous owner(s) in KM
Fuel_Type	The type of fuel used by the car
Transmission	The type of transmission used by the car
Owner_Type	Whether the ownership is Firsthand, 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	Price of new model
Price	The price of the used car in INR Lakhs

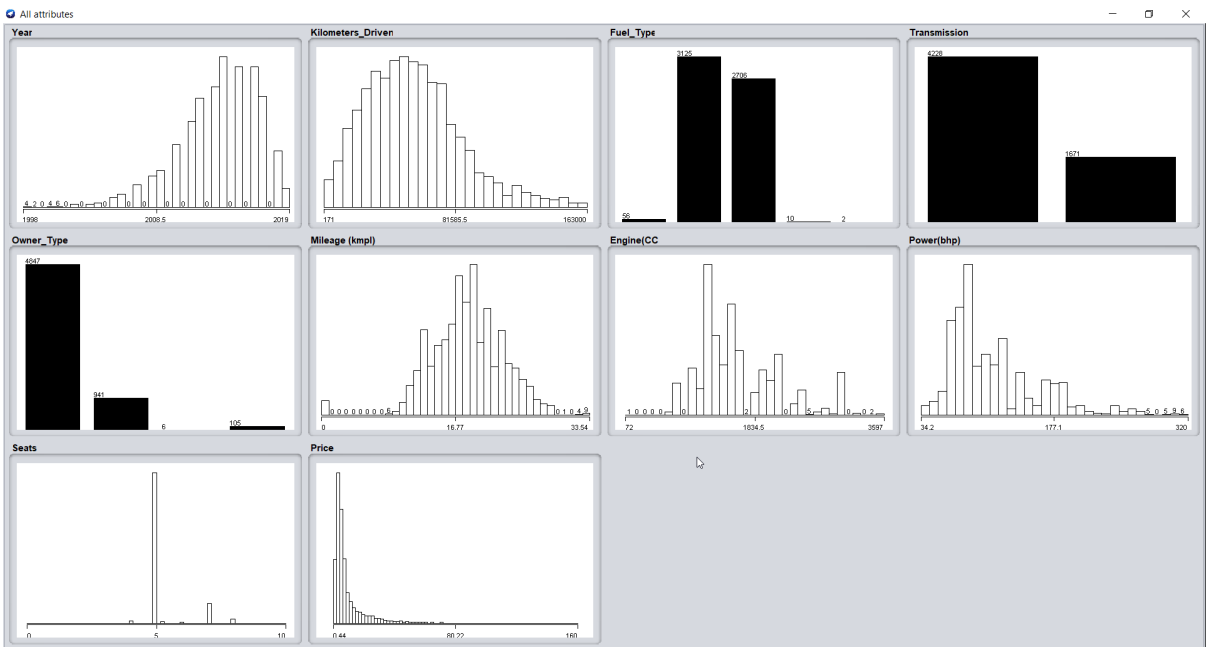
DATA CLEANING:

As depicted in the train dataset properties table, there are many missing values in the dataset. These missing will affect the accuracy in building the model. So, there is a need to remove the missing data, or to fill the missing data. **"RemoveMissingValue"** filer has been used in-order to replace missing value with median. And in for categorical variables such as Seats, etc. Missing values were replaced by mode.

DATA VISUALIZATION:

Visualizing the data is the most crucial part in analyzing the model. We use WEKA explorer to visualize the data in the form of Histogram plots. Below figures shows the modified data after using the "RemoveMissingValue" filter and removing unwanted attributes such "name","Location","New_Price".





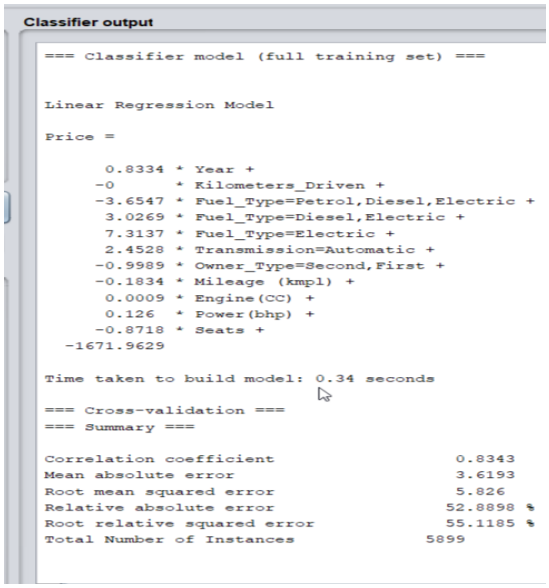
From the scatter plot, we can see that some field were evenly distributed.

We are using the following algorithms in selecting the best fit algorithm.

1. Linear Regression
2. Random Forest

When Linear Regression is used the model showed an accuracy measure of

83%, with the following metric evaluations show below.



When Random Forest regressor is used the model showed an accuracy measure of 94%, with the following metric evaluations show below

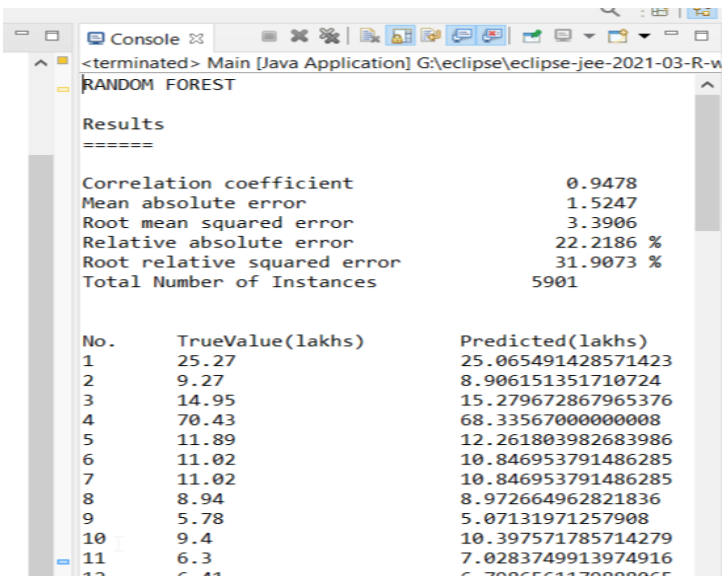
5. RESULT:

From the above analysis, the most suitable algorithm for building the model is to use Random Forest regressor. Hence, the final model was being built using Random forest regressor Below table, shows the accuracy percentage of both models for comparative analysis.

MODEL	ACCURACY
Linear Regression	83.4%
Random Forest	94.4%

.

The below figure shows the output of the random forest regressor, which was run on the test dataset. And actual values to that of the predicted values is also being shown in the figure.



6.ADVANTAGES AND DISADVANTAGES:

Advantages:

1. Accuracy
2. Generic model i.e. can be used for any type of raw data
3. Robust
4. Feature Engineering is not needed

Disadvantages:

1. Sometimes may undergo overfitting, if dataset is small.

7. APPLICATIONS:

This model can be used in following sectors:

1. Petrol price prediction
2. Land resale value prediction

8. CONCLUSION:

After trying and testing 2 different algorithms, the best accuracy was found out in Random Forest Regressor (94.4%) while Machine Learning algorithm produced an accuracy of (83.4%). While new features can be created via feature engineering which may help in predicting the target variable. overall, Random Forest Regressor classifier provides the best result in terms of accuracy for the given dataset, without any feature engineering needed. Because of its simplicity and the fact that it can be implemented relatively easy and quick.

9. FUTURE SCOPE:

The designed model can be enhanced using neural networks, gradient decent algorithm. These algorithms help in better performance, irrespective of the size of dataset, the same models can be used for big-data analysis also.

10. BIBILOGRAPHY:

[1] https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf

[2] https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56

[3] https://machinelearningmastery.com/use-regression-machine-learning-algorithms-weka/

11. APPENDIX

a. SOURCE CODE

web.xml

```
<projectxmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https
://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com</groupId>
```

```
<artifactId>org.ml</artifactId>
<version>0.0.1-SNAPSHOT</version>
<dependencies>
<dependency>
<groupId>nz.ac.waikato.cms.weka</groupId>
<artifactId>weka-stable</artifactId>
<version>3.8.0</version>
</dependency>
<dependency>
<groupId>tech.tablesaw</groupId>
<artifactId>tablesaw-core</artifactId>
<version>0.38.1</version>
</dependency><dependency>
<groupId>tech.tablesaw</groupId>
<artifactId>tablesaw-jsplot</artifactId>
<version>0.38.1</version>
</dependency><!-- Thanks for using https://jar-download.com --></dependencies>
<properties>
<maven.compiler.source>1.8</maven.compiler.source>
<maven.compiler.target>1.8</maven.compiler.target>
</properties>
</project>
```

Main.java

```
package org.ml;
import java.io.*;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Random;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.functions.LinearRegression;
import weka.classifiers.trees.RandomForest;
import weka.core.Instances;
public class Main {

    public static String trainDataPath="";
    public static String testDataPath="";
    public static void main(String args[]) throws Exception{
        String trainFileName="\\\\train1.arff";
        String testFileName="\\\\test1.arff";
        trainDataPath=getCurrentDirectoryPath()+trainFileName;
        //System.out.println(trainDataPath);
        testDataPath=getCurrentDirectoryPath()+testFileName;
        BufferedReader br=new BufferedReader(new FileReader(trainDataPath));
        Instances trainData=new Instances(br);
        BufferedReader brTest=new BufferedReader(new FileReader(testDataPath));
```

```
            Instances testData=new Instances(brTest);
            testData.setClassIndex(testData.numAttributes()-1);
            trainData.setClassIndex(trainData.numAttributes()-1);
            br.close();
            brTest.close();
            randomForest(testData,trainData);
        }
        /*
         getCurrentDirectoryPath()
         -> return type- String
         -> returns current addresses of the working data-sets.
         -> implemented to make the code generic, to use the project in any system.
         */
        private static String getCurrentDirectoryPath() {
            String path=System.getProperty("user.dir")+"\\src\\main\\java\\org\\ml";
            //System.out.println(path);
            return path.replace("\\","\\\\");
        }
        /*
         *randomForest(Instances i1, Instances i2)
         *return type- void
         *complete regressor and the training and testing is done
         */
        private static void randomForest(Instances testData, Instances trainData) throws Exception {
            // TODO Auto-generated method stub
            System.out.println("RANDOM FOREST");
            RandomForest rf=new RandomForest();
            rf.setNumIterations(100);
            Evaluation evaluation = new Evaluation(trainData);
            evaluation.crossValidateModel(rf, trainData,10 , new Random(1));
            rf.buildClassifier(testData);
            printResults(evaluation,rf,testData);
        }
        /*printResults(Evaluation e, RandomForest regressor, Instances instance)
         * return type- void
         * results and testing summary is being printed
         */
        private static void printResults(Evaluation eval,RandomForest rf,Instances testData) throws
Exception {
            System.out.println("------RESULTS------");
            System.out.println(eval.toSummaryString("",true));
            System.out.println();
            for(int i=0;i<testData.numInstances();i++) {
                String trueLabel;
                trueLabel=testData.instance(i).toString(testData.classIndex());
                double predIndex=rf.classifyInstance(testData.instance(i));
                System.out.println((i+1)+"\t"+trueLabel+"\t\t\t"+predIndex);
            }
```

```
    }
}
```