# Introduction

## Overview:

It is likely that sales of second-hand imported (reconditioned) cars and used cars will increase. In many developed countries, it is common to lease a car rather than buying it outright. A lease is a binding contract between a buyer and a seller (or a third party – usually a bank, insurance firm or other financial institutions) in which the buyer must pay fixed installments for a pre-defined number of months/years to the seller/financer. After the lease period is over, the buyer has the possibility to buy the car at its residual value, i.e. its expected resale value.

## Purpose:

Commercial interest to seller/financers to be able to predict the salvage value (residual value) of cars with accuracy.

# Literature Survey

## Existing problem:

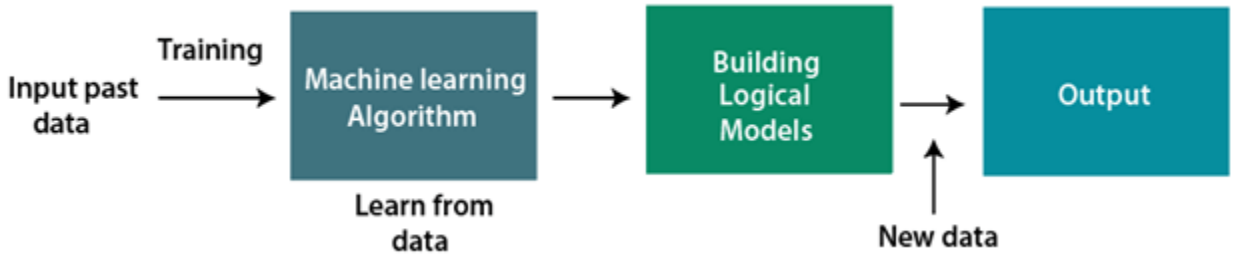
Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases.

## Proposed Solution:

Predictive analytics is the process of using computer models to forecast future events using sophisticated programs and depend on certain efficient technologies such as artificial intelligence, data mining, and machine learning to process massive information. Using these technologies the model would analyze and predict the future happening based on the current conditions.
The predictive analytics could enhance the overall banking experience for the customers in several ways, it could find it unsettling that financial institutions that have no much information, and that rely on computers for decision making could affect one's life. On the other hand, computers are always available and would provide a similar service to every customer without being partial to any customers.

# Theoritical Analysis

## Block Diagram




## Hardware and software requirements of the project

## Hardware requirements:

1)If your tasks are small and can fit in a complex sequential processing, you don't need a big system. You could even skip the GPUs altogether. A CPU such as i7–7500U can train an average of ~115 examples/second
2)256 MB RAM
3)1 Gb hard free drive space

### Software Requirements:

1. eclipse ide for java developers

2. weka library

# Experimental Investigation

### Data Processing:

Stages of Data Preprocessing are:

1. Data Cleaning
2. Data Integration
3. Data Reduction
4. Data Transformation

### Data Cleaning:

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.
In this dataset datacleaning is done in weka by selecting filters such as remove duplicates,missingvaluebyuserinput and applyting them to dataset.

1. Remove duplicate is used for removing duplicate elements in a dataset.

1. MissingvaluebyuserInput filter is used to replace missing values in dataset by user input.

## Data Integration:

Data Integration refers to the process of unifying data from multiple data sources.

## Data Reduction:

Data Reduction mechanism can be used to reduce the representation of the large dimensional data. By using a data reduction technique, you can reduce the dimensionality that will improve the manageability and visibility of data. Further, you can achieve similar accuracies.

Principal Component Analysis is also known as Karhunen-Loeve or K-L method, is used to reduce components to handlable attributes from a large number of the dimensions. In other words, Principal Component Analysis combines the important features of attributes and reduces the variables by introducing alternative variables. After the Principal Component Analysis is done, multiple dimensions can be represented into a manageable number of variables.

### Data Transformation:

Data Transformation is the technique of converting data from one format to another.

1. It is performed in weka through filters such as discretization,standardize,nominaltobinary
   Discretize will discretize the values according to a number of bins (n). Weka will simply cut the range of the values in n subsets, and give the value of the subset to the instances. This is if your attribute is really a continuous variable.

1. Numeric To Nominal is to transform some Numeric values into a Nominal variable, if this attributes has few unique values. For example, if you have an ID attribute which clusters your dataset in few subsets, it may be wise to convert it into a Nominal attribute instead of treating it like a number. This applies for attributes which are not really continuous, but treated as numeric .

## DATA VISUALIZATION:

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

# Model Building:

The algorithm used for building model is logistic regression.
LOGISTIC REGRESSION:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X.

**Metrics used**:

Precision is a metric that quantifies the number of correct positive predictions made.

1. Precision = TruePositives / (TruePositives + FalsePositives)

Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made.

1. Recall = TruePositives / (TruePositives + FalseNegatives)

F-Measure provides a way to combine both precision and recall into a single measure that captures both properties.

1. F1 Score = 2*(Recall * Precision) / (Recall + Precision)

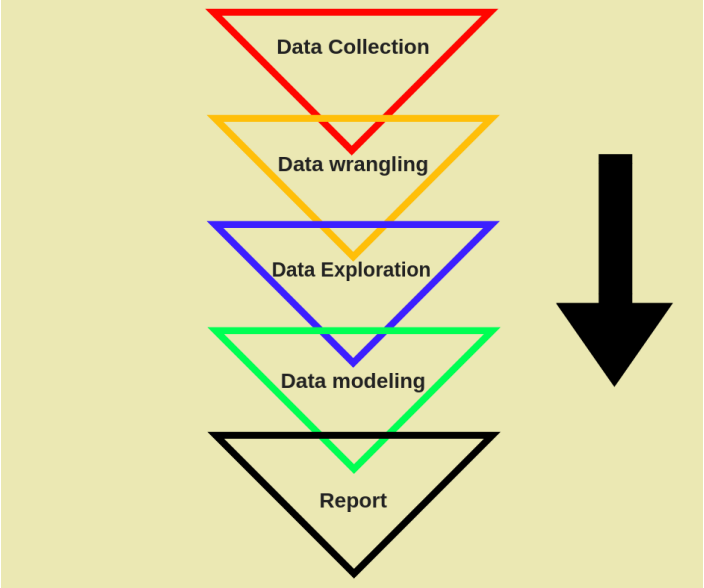
**Accuracy** - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

1. Accuracy = TP+TN/TP+FP+FN+TN

# Testing the model:

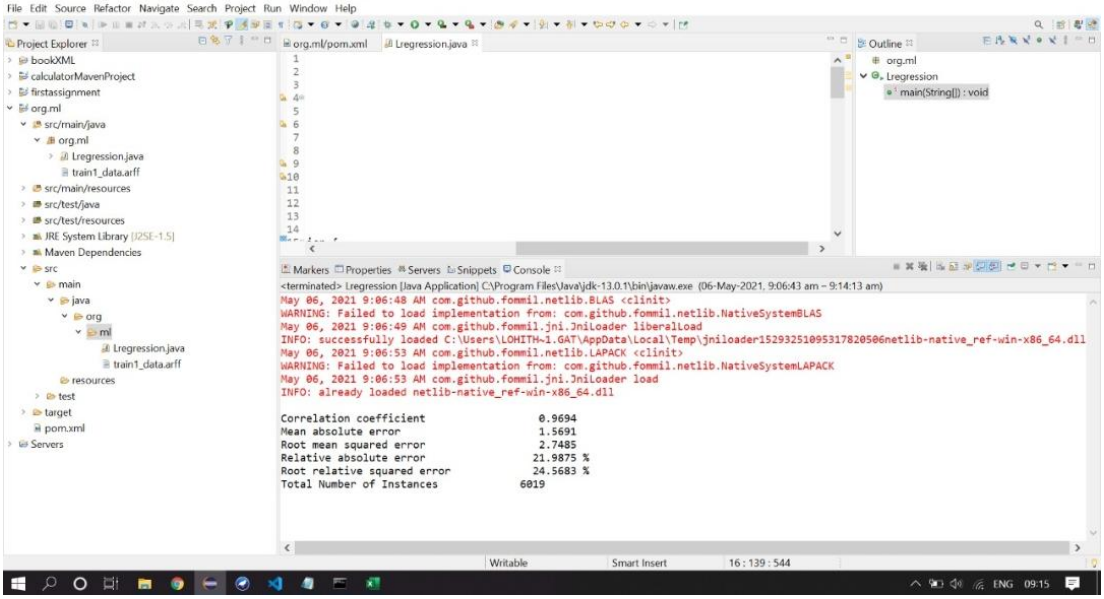
Now that we have trained our model lets see how it did on the data .As you can see , the scores are very close which indicates that we avoided over-fitting. I should mention that this is a good indication that we have not over-fit the model, however it is not the end all be all. Next we'll discuss another method to prevent over-fitting of our data and hopefully improve our ability to generalize over new data.

# Flow Chart

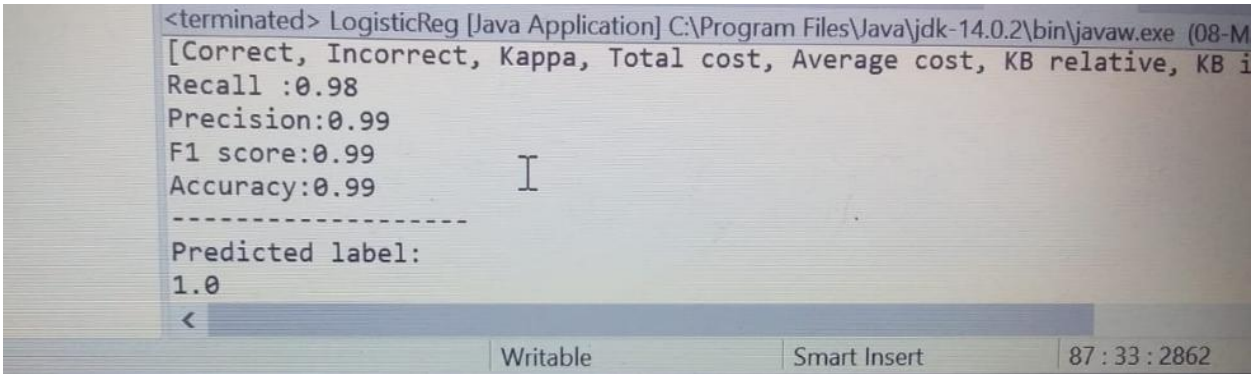



# Result

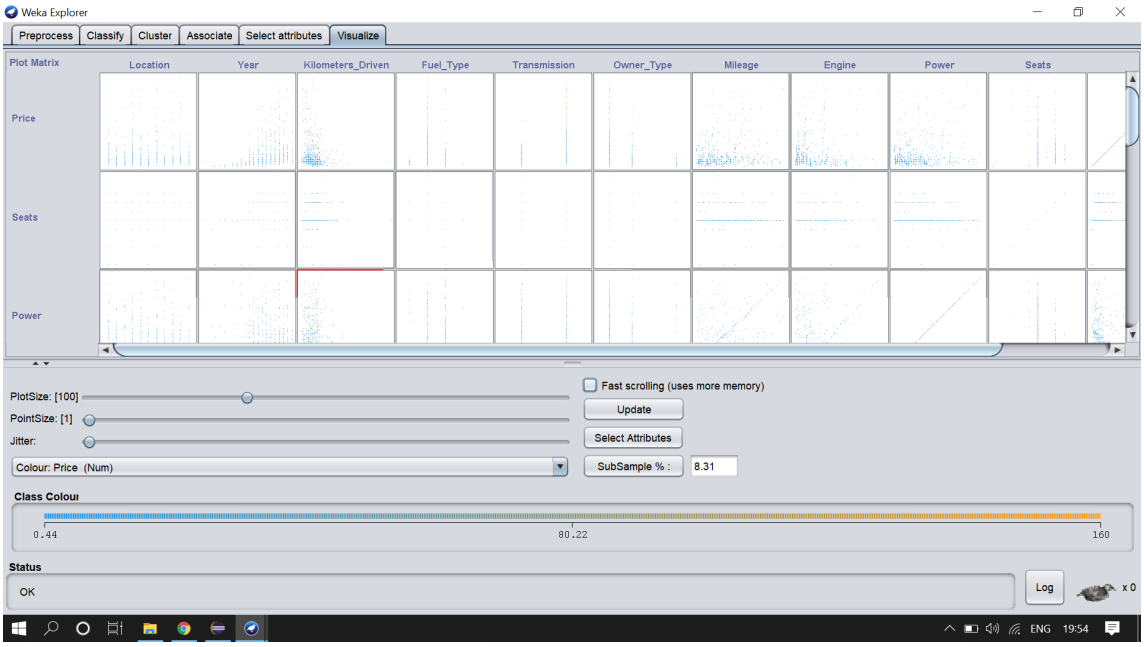
Model Bulid:



Predicition:




visualisation:




# Advantages and Disadvantages

## Advantages:

**Stay one step ahead in performance**

After all, it's a *forecasting* technology.It will take all of your campaign data (and previous) to determine what's most likely to fail or succeed in the future.

**It saves time and energy**

**Decoding Customers Sentiment:**Customer analytics through predictive tools provide customers choices and behaviour to the bank.

Disadvantages:

**It can be intimidating to adopt**.

**You have to spend time using it for its full effect.**

## Conclusion:

Commercial interest to seller/financers to be able to predict the salvage value (residual value) of cars with accuracy.

Appendix:

```
package org.ml;


import java.util.Arrays;

import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.LinearRegression;
import weka.core.DenseInstance;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class Lregression {
	public static void main(String[] args) throws Exception {
		DataSource src=new DataSource("C:\\java eclipse work\\org.ml\\src\\main\\java\\org\\ml\\train_data.arff");
		//DataSource src1=new DataSource("C:\\java eclipse work\\org.ml\\src\\main\\java\\org\\ml\\test_data.arff");

		Instances ds=src.getDataSet();
		ds.setClassIndex(ds.numAttributes()-1);
		LinearRegression lr=new LinearRegression();
		lr.buildClassifier(ds);
		Instances ds1=src1.getDataSet();
```

```
                ds1.setClassIndex(ds1.numAttributes()-1);
                LinearRegression lr1=new LinearRegression();
                lr.buildClassifier(ds1);
                Evaluation lr_eval=new Evaluation(ds);
                lr_eval.evaluateModel(lr, ds);
                System.out.println(lr_eval.toSummaryString());
                Instance car = ds1.lastInstance();
                double price = lr.classifyInstance(car);
                System.out.println("------------------------");
                System.out.println("PRECTING THE PRICE : "+price);
                Classifier classifier = new weka.classifiers.functions.LinearRegression();
                classifier.buildClassifier(ds);
                double confusion[][] = lr_eval.confusionMatrix();
                System.out.println("Confusion matrix:");
                for (double[] row : confusion)
                        System.out.println(      Arrays.toString(row));
                System.out.println("-------------------");
                System.out.println("Area under the curve");
                System.out.println( lr_eval.areaUnderROC(0));
                System.out.println("-------------------");
                System.out.println(lr_eval.getAllEvaluationMetricNames());
                System.out.print("Recall :");
                System.out.println(Math.round(lr_eval.recall(1)*100.0)/100.0);
                System.out.print("Precision:");
                System.out.println(Math.round(lr_eval.precision(1)*100.0)/100.0);
                System.out.print("F1 score:");
                System.out.println(Math.round(lr_eval.fMeasure(1)*100.0)/100.0);
                System.out.print("Accuracy:");
                double acc = lr_eval.correct()/(lr_eval.correct()+ lr_eval.incorrect());
                System.out.println(Math.round(acc*100.0)/100.0);
                System.out.println("-------------------");
                Instance predicationDataSet = ds1.get(2);
                double value = classifier.classifyInstance(predicationDataSet);
                System.out.println("Predicted label:");
                System.out.print(value);
        }
}
```