INTRODUCTION

1.1 **OVERVIEW** :

An insurance policy is an arrangement by which a company undertakes to provide a guarantee of compensation for specified loss, damage, illness, or death in return for the payment of a specified premium. A premium is a sum of money that the customer needs to pay regularly to an insurance company for this guarantee. Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimize its business model and revenue

1.2 **PURPOSE**

The main aim of this use case is to predict, whether the customer would be interested in Vehicle insurance provided information about demographics (gender, age, region code type), Vehicles (Vehicle Age, Damage),Policy (Premium, sourcing channel) etc

LITERATURE SURVEY

2.1 EXISTING PROBLEM

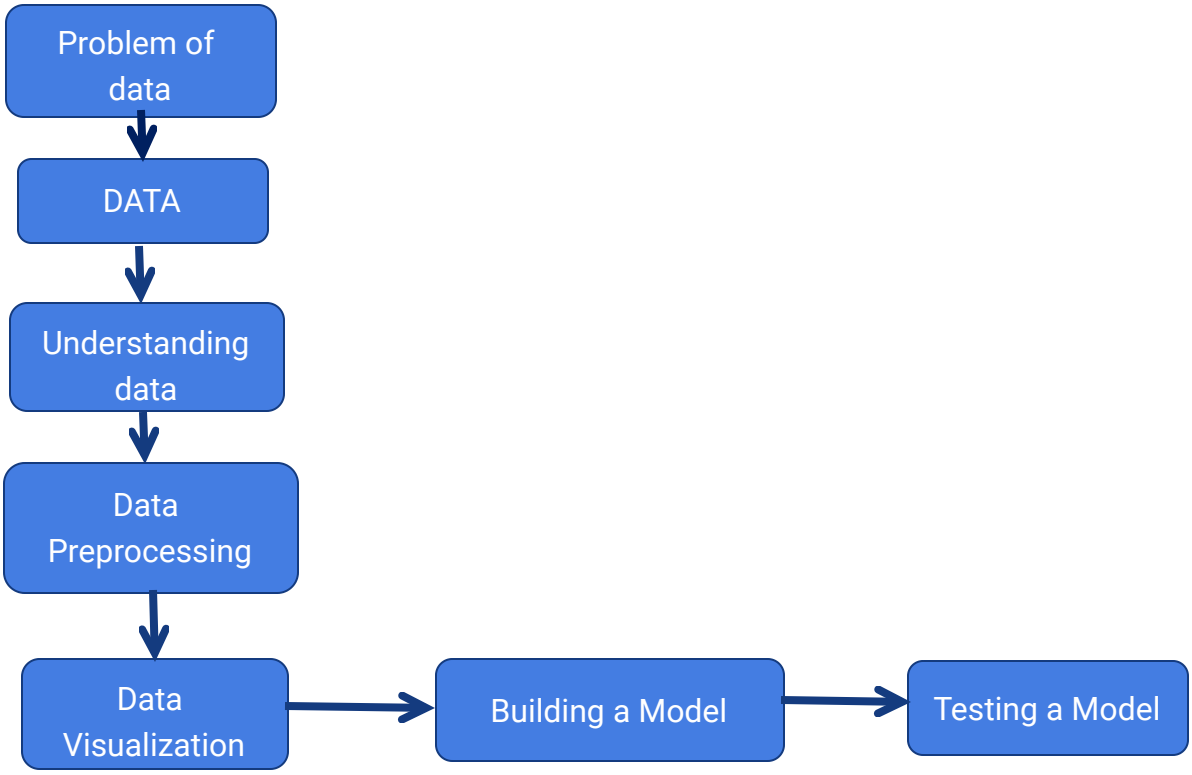
The customers don't know much about the process of insurance and the documentation what they have to submit while applying to Insurance.

2.2 Proposed Solution

Here, we predict the vehicle insurance on the given details and we will give the explanation about the insurance.If any incident occurs what claims did he get and we tell about the difference between the premium account and normal account.In this we describe the benefits of the accounts.

THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM



HARDWARE / SOFTWARE DESIGNING

SOFTWARE REQUIRED

1.WEKA SOFTWARE
2.ECLIPSE IDE
3.INSTALL TABLE SAW AND WEKA PACKAGES

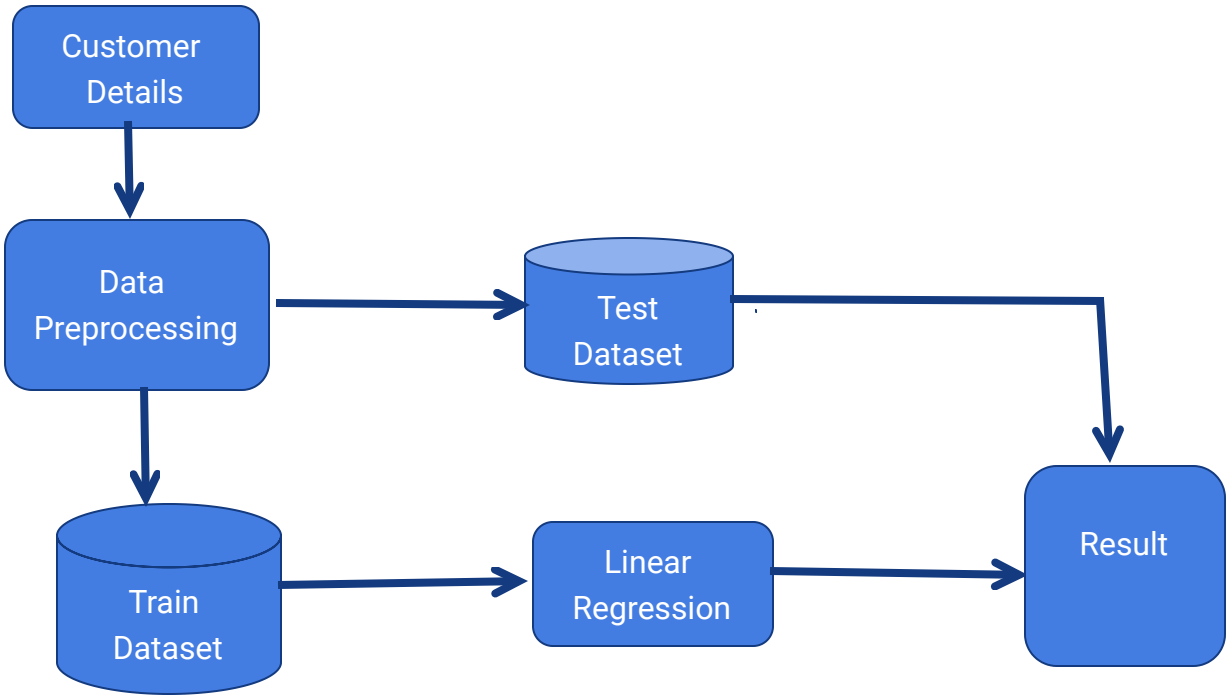
EXPERIMENTAL INVESTIGATION

- People are not much familiar with the insurance .some people may not apply for the renewal of insurance.
- In earlier , there are less number of insurance companies .There are only

government insurance company

- In insurance there will be two types one is normal and other is premium.
- Normal only some benefits compare to premium.

FLOWCHART



RESULT

Correlation coefficient	1
Mean absolute error	6.6518
Root mean squared error	8.1034
Relative absolute error	0.032 %
Root relative squared error	0.0307 %
Total Number of Instances	780

Time taken to test model on supplied test set: 0.06 seconds

=== Summary ===

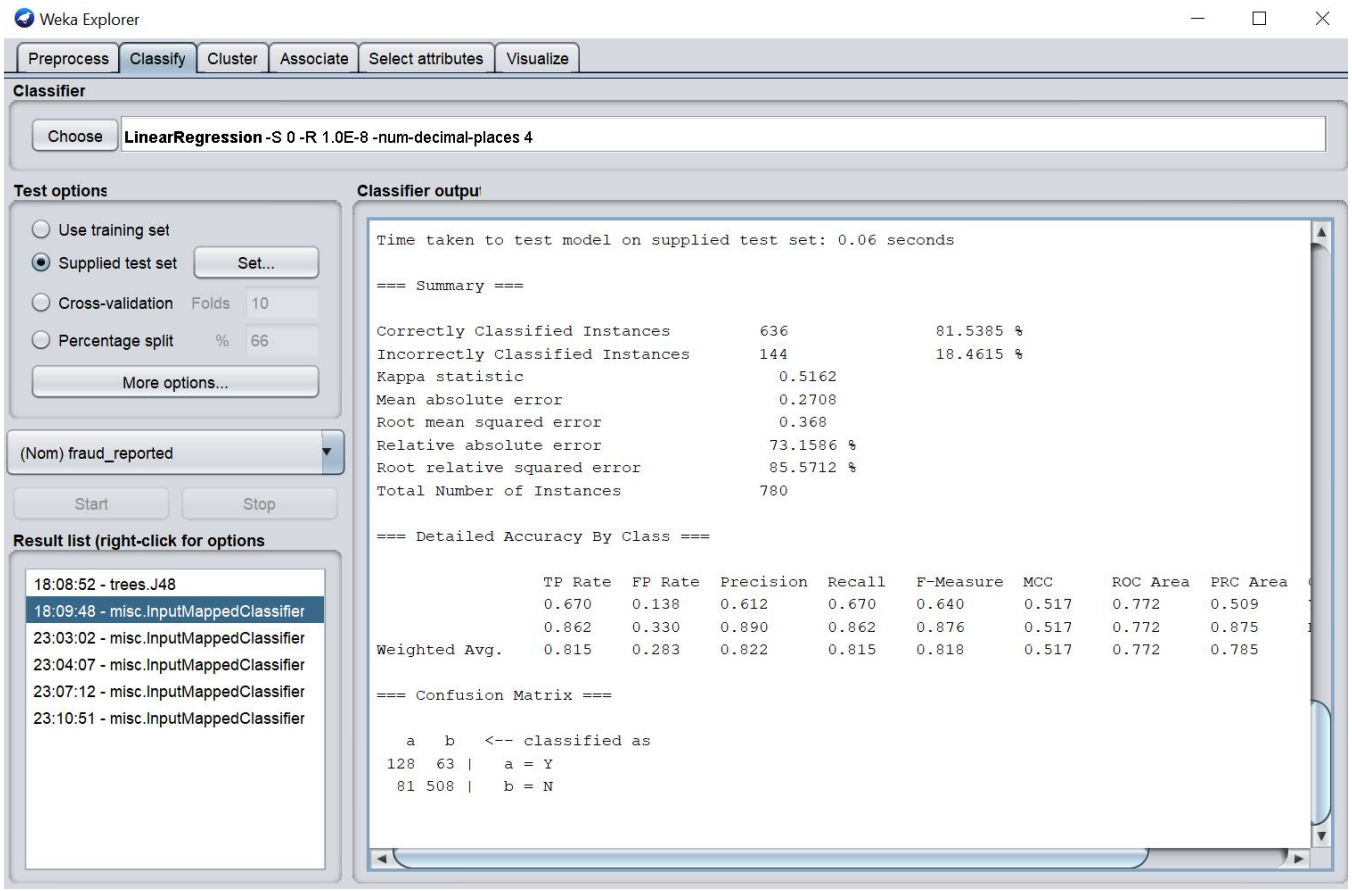
Correctly Classified Instances	636	81.5385 %
Incorrectly Classified Instances	144	18.4615 %
Kappa statistic	0.5162	
Mean absolute error	0.2708	
Root mean squared error	0.368	
Relative absolute error	73.1586 %	
Root relative squared error	85.5712 %	
Total Number of Instances	780	

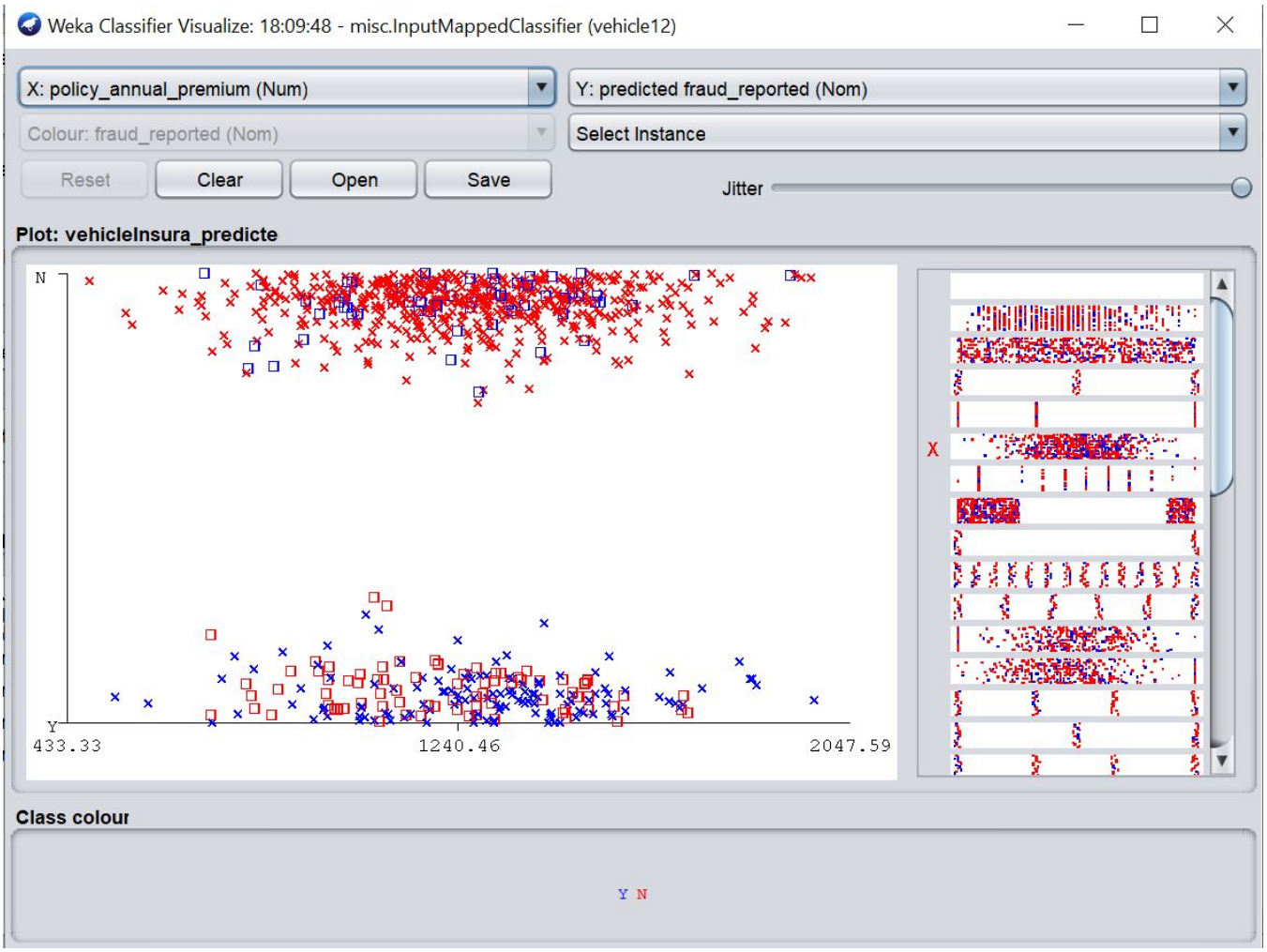
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.670	0.138	0.612	0.670	0.640	0.517	0.772	0.509	Y
	0.862	0.330	0.890	0.862	0.876	0.517	0.772	0.875	N
Weighted Avg.	0.815	0.283	0.822	0.815	0.818	0.517	0.772	0.785	

=== Confusion Matrix ===

a	b	<-- classified as
128	63	a = Y
81	508	b = N


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) fraud_reported
Start
Stop
Result list (right-click for options
18:08:52 - trees.J48
18:09:48 - misc.InputMappedClassifier
23:03:02 - misc.InputMappedClassifier
23:04:07 - misc.InputMappedClassifier
23:07:12 - misc.InputMappedClassifier
23:10:51 - misc.InputMappedClassifier
Classifier output
Time taken to test model on supplied test set: 0.06 seconds
=== Summary ===
Correctly Classified Instances 636 81.5385 %
Incorrectly Classified Instances 144 18.4615 %
Kappa statistic 0.5162
Mean absolute error 0.2708
Root mean squared error 0.368
Relative absolute error 73.1586 %
Root relative squared error 85.5712 %
Total Number of Instances 780
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area
0.670 0.138 0.612 0.670 0.640 0.517 0.772 0.509
0.862 0.330 0.890 0.862 0.876 0.517 0.772 0.875
Weighted Avg. 0.815 0.283 0.822 0.815 0.818 0.517 0.772 0.785
=== Confusion Matrix ===
a b <-- classified as
128 63 | a = Y
81 508 | b = N



Advantages and Disadvantages

Advantages

- Its Efficiency is 80%
- It Predicts accurately

Applications

- It is used in insurance company

Conclusion

- The customer can know whether he can apply for premium or normal.The result of insurance is given in few Minutes

Future Scope

- It works very efficiently .

- If we give the predictions the growth of insurance will be increased

BIBILOGRAPHY

https://www.kaggle.com/c/auto-insurance-fall-2017/data
https://www.synthesized.io/data-template-pages/vehicle-insurance-claim-prediction

APPENDIX

package org.dl;

```
import java.io.IOException;

import java.util.Arrays;

import tech.tablesaw.api.Table;
import tech.tablesaw.plotly.Plot;
import tech.tablesaw.plotly.components.Figure;
import tech.tablesaw.plotly.components.Layout;
import tech.tablesaw.plotly.traces.BoxTrace;
import tech.tablesaw.plotly.traces.ScatterTrace;
import weka.classifiers.Classifier;
import weka.classifiers.evaluation.Evaluation;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class DataAnalysis{

	public static Instances getInstances (String filename)
	{

		DataSource source;
		Instances dataset = null;
		try {
			source = new DataSource(filename);
```

```
                dataset = source.getDataSet();
                dataset.setClassIndex(dataset.numAttributes()-1);


            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();

            }

            return dataset;
        }
        @SuppressWarnings("static-access")
        public static void main(String args[]) {
            System.out.println("data analytics");
            try {
            Table
insurance_data=Table.read().csv("C:\\Users\\hp\\Desktop\\1214\\org.dl\\src\\main\\java\\org\\dl\\Vehicle12.csv");

            System.out.println(insurance_data.shape());
            System.out.println(insurance_data.structure());
            System.out.println(insurance_data.summary());
            Layout l1=Layout.builder().title("age distribution").build();
            //box-plot
            BoxTrace
t1=BoxTrace.builder(insurance_data.categoricalColumn("fraud_reported"),
insurance_data.nCol("vehicle_claim")).build();
            Plot.show(new Figure(l1,t1));
            //scatter-plot
            ScatterTrace
s=ScatterTrace.builder(insurance_data.nCol("policy_annual_premium"),insurance_data.nCol("vehicle_claim")).build();
            Plot.show(new Figure(l1,s));
            }
            catch(IOException e) {
                e.printStackTrace();
            }
            Instances train_data =
```

```
getInstances("C:\\Users\\hp\\Desktop\\Train.arff");
                                        Instances test_data =
getInstances("C:\\Users\\hp\\Desktop\\Test.arff");
                                        System.out.println("The train data size is "+train_data.size());

                                        /** Classifier here is Linear Regression */
                                        Classifier classifier = new weka.classifiers.functions.Logistic();
                                        /** */
                                        try {
                                                classifier.buildClassifier(train_data);
                                        } catch (Exception e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }


                                        /**
                                         * train the algorithm with the training data and evaluate the
                                         * algorithm with testing data
                                         */
                                        Evaluation eval;
                                        try {
                                                eval = new Evaluation(train_data);
                                                eval.evaluateModel(classifier, test_data);
                                                System.out.println("** Logistic Regression Evaluation
with Datasets **");
                                                System.out.println(eval.toSummaryString());
                                                System.out.print(" the expression for the input data as
per alogorithm is ");
                                                System.out.println(classifier);

                                                double confusion[][] = 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( eval.areaUnderROC(0));
```

```
                                        System.out.println("------------------");

System.out.println(eval.getAllEvaluationMetricNames());

                                        System.out.print("Recall :");

System.out.println(Math.round(eval.recall(1)*100.0)/100.0);

                                        System.out.print("Precision:");

System.out.println(Math.round(eval.precision(1)*100.0)/100.0);
                                        System.out.print("F1 score:");

System.out.println(Math.round(eval.fMeasure(1)*100.0)/100.0);

                                        System.out.print("Accuracy:");
                                        double acc = eval.correct()/(eval.correct()+
eval.incorrect());
                                        System.out.println(Math.round(acc*100.0)/100.0);


                                        System.out.println("------------------");
                                } catch (Exception e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                }

                                /** Print the algorithm summary */

                                Instance predicationDataSet = test_data.get(2);
                                double value;
                                try {
                                        value = classifier.classifyInstance(predicationDataSet);
                                        System.out.println("Predicted label:");
                                System.out.print(value);
                                } catch (Exception e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
```

```
                }
                /** Prediction Output */


            }

        }
```