CAR INSURANCE COLD CALL PREDICTION

1 INTRODUCTION

1.1 Overview

Main aim of this project is to predict whether customers contacted will buy car insurance or not based on the information given by the customer.This project uses Machine Learning techniques in JAVA and WEKA to predict the outcome.

1.2 Purpose

This project helps the banks to know whether the customer would buy the insurance or not , which saves time of the bank to call and ask other customers who may buy car insurance.

2 LITRATURE SURVEY

2.1 Existing Problem

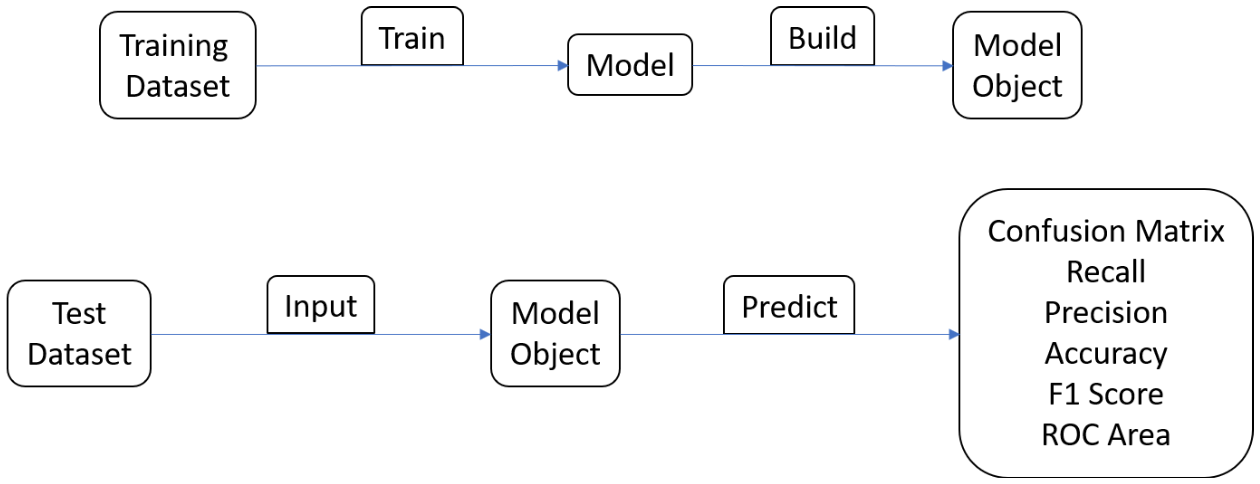
There are many solutions that this problem has been solved in the past. Few of them are kNN, SVM, Decision Tree, Random Forest, AdaBoost, XGBoost, etc.

2.2 Proposed Solution

This project uses Logistic Regression which finds the coefficients of the linear graph (relation between dependent variable and independent variables) using Gradient Descent algorithm to solve the problem.

3 THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software Designing

Hardware Requirements:

-Memory : min- 512MB ; recommended- 1GB or more
-Free disk space : min- 300MB ; recommended- 1GB or more
-Processor speed : min- 800Mhz ; recommended- 1.5Ghz or faster

Software Requirements:

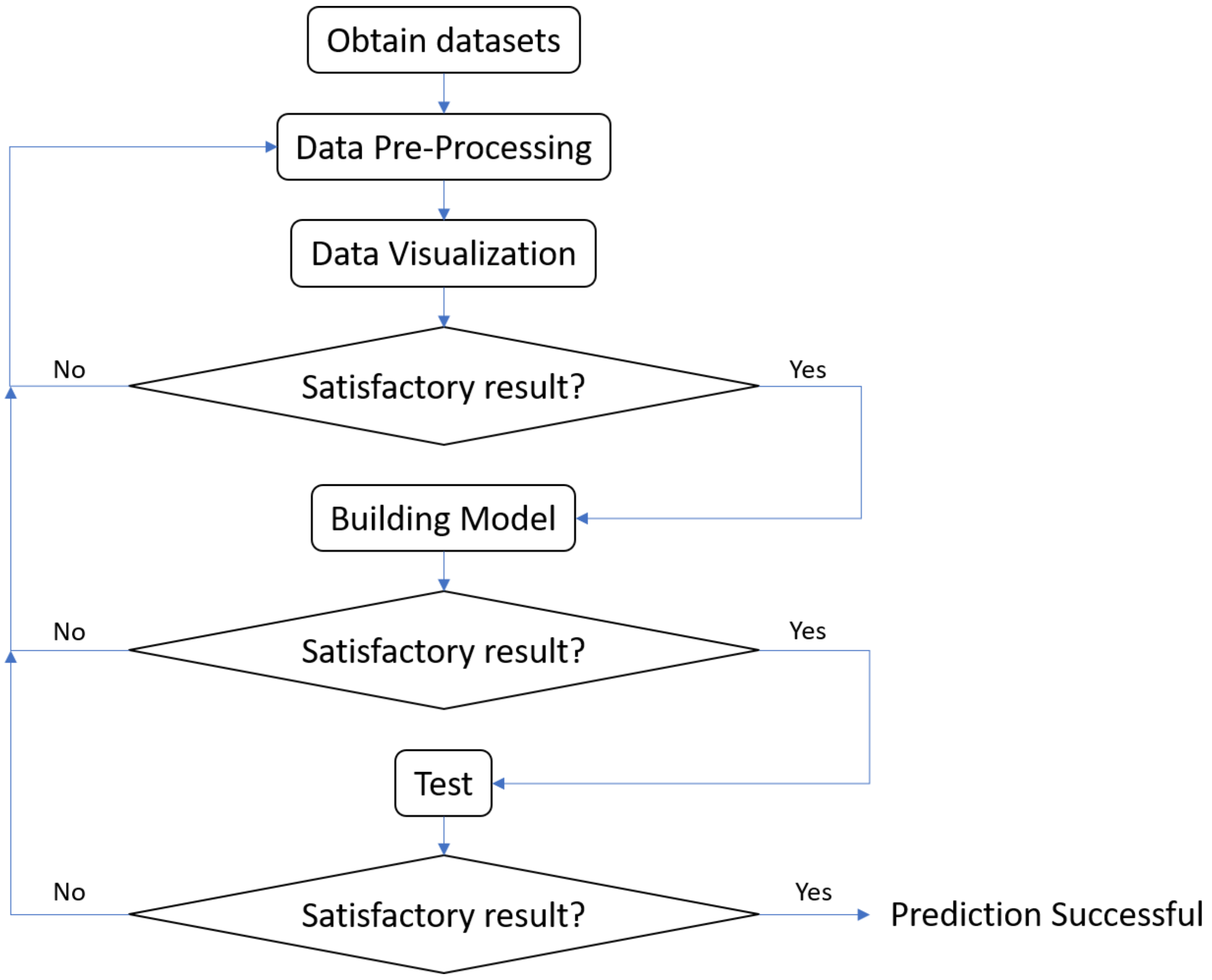
- JAVA 8 or higher

-WEKA

4 EXPERIMENTAL INVESTIGATIONS

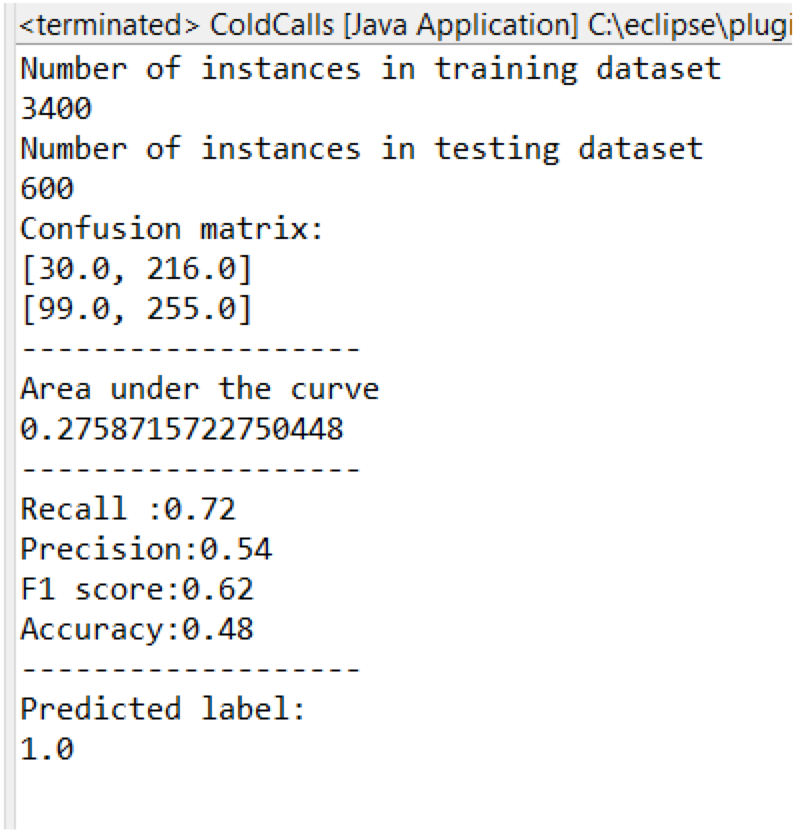
- The dataset provided was not structured properly and had lot of missing values in nominal attributes.
- Few of the attributes were not capable for data processing. This was solved with pre-processing technique.
- The class was not in nominal form to perform logistic regression.
- Few attributes were in nominal form after replacing the missing values these attributes were converted to binary format using One Hot Encoding Technique.
- After data visualization, the data coefficients with less significance were observed and removed.
- It was observed that the attributes (Id, Age, Balance, LastcontactDay, DaysPassed, CallDuration[CallEnd - CallStart]) had almost no significance.
- Date with approximate error of +/- 0.1 for the coefficients is removed to get best results for the test cases that were considered.
-

5 FLOWCHART



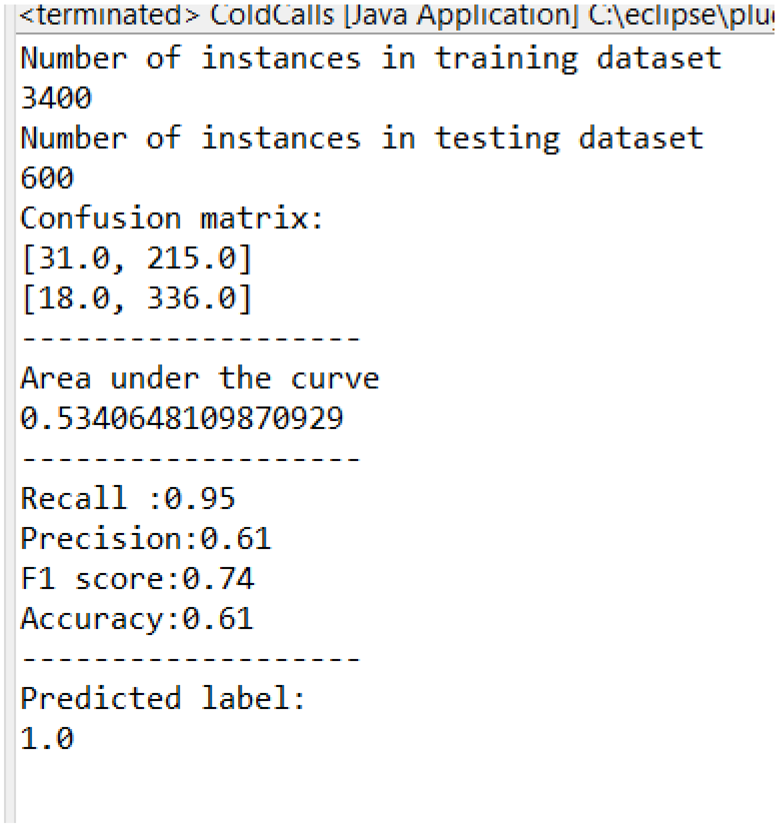
6 RESULT

It is observed that the accuracy and area under ROC Curve of the project is satisfactory with a value of 0.61 and 0.534 respectively, with recall value being 0.95.



```
<terminated> ColdCalls [Java Application] C:\eclipse\plugi
Number of instances in training dataset
3400
Number of instances in testing dataset
600
Confusion matrix:
[30.0, 216.0]
[99.0, 255.0]
-------------------
Area under the curve
0.2758715722750448
-------------------
Recall :0.72
Precision:0.54
F1 score:0.62
Accuracy:0.48
-------------------
Predicted label:
1.0
```

Output of testing dataset by initial trained model



```
<terminated> ColdCalls [Java Application] C:\eclipse\plu
Number of instances in training dataset
3400
Number of instances in testing dataset
600
Confusion matrix:
[31.0, 215.0]
[18.0, 336.0]
-------------------
Area under the curve
0.5340648109870929
-------------------
Recall :0.95
Precision:0.61
F1 score:0.74
Accuracy:0.61
-------------------
Predicted label:
1.0
```

Output of the testing dataset by final trained model

7 ADVANTAGES AND DISADVANTAGES

Using Logistic Regression it is easier to implement, interpret, and very efficient to train, but the major limitation is the assumption of linearity between the dependent variable and the independent variables.

8 APPLICATIONS

This project can be applied in Banks to predict whether their clients are likely to buy Car Insurance or not.

9 CONCLUSION

In conclusion, this project can be used to predict whether a customer will buy car insurance or not. The dataset provided was analyzed, pre-processed and visualized using WEKA. Trained model is built using Eclipse IDE with WEKA library. Test dataset was given to the trained model and output was satisfactory with accuracy of 0.61. Other methods can be implemented to improve the accuracy of the prediction.

10 FUTURE SCOPE

Better algorithms like Random Forest, XGBoost can be used to reduce the errors and increase area under ROC curve and other parameters.

11 BIBILOGRAPHY

https://www.kaggle.com/emmaren/cold-calls-data-mining-and-model-selection
https://www.kaggle.com/loveall/cleaning-visualizing-and-modeling-cold-call-data/notebook

12 APPENDIX

A. Source Code

```
package smartinternz;
import java.util.Arrays;
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 ColdCalls {
	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;
}
	public static void main(String[] args) throws Exception{
		Instances train_data = getInstances("E:\\A.Sai
Preeth\\placements\\oracle\\numeric\\edit\\rem\\rem2\\rem3\\rem4\\carInsurance_tr
ain.arff");
		Instances test_data = getInstances("E:\\A.Sai
Preeth\\placements\\oracle\\numeric\\edit\\rem\\rem2\\rem3\\rem4\\carInsurance_t
est.arff");
		System.out.println("Number of instances in training dataset");
		System.out.println(train_data.size());
		System.out.println("Number of instances in testing dataset");
		System.out.println(test_data.size());
		Classifier classifier = new weka.classifiers.functions.Logistic();
		classifier.buildClassifier(train_data);
		Evaluation eva = new Evaluation(train_data);
		eva.evaluateModel(classifier, test_data);
		double confusion[][] = eva.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( eva.areaUnderROC(0));
		System.out.println("-----------------");
		System.out.print("Recall :");
		System.out.println(Math.round(eva.recall(1)*100.0)/100.0);
		System.out.print("Precision:");
		System.out.println(Math.round(eva.precision(1)*100.0)/100.0);
		System.out.print("F1 score:");
		System.out.println(Math.round(eva.fMeasure(1)*100.0)/100.0);
		System.out.print("Accuracy:");
		double acc = eva.correct()/(eva.correct()+ eva.incorrect());
		System.out.println(Math.round(acc*100.0)/100.0);
```

```
                System.out.println("------------------");
                Instance predicationDataSet = test_data.get(2);
                double value = classifier.classifyInstance(predicationDataSet);
                System.out.println("Predicted label:");
                System.out.print(value);


        }

}
```