Loan Eligibility prediction

1. INTRODUCTION

1.1 Overview

Loans are the core business of banks. The main profit comes directly from the loan's interest. The loan companies grant a loan to a person after a thorough background check which involves an intensive process of verification and validation. However, they still don't have assurance if the applicant is able to repay the loan with no difficulties. If the applicant fails to repay the loan or is unable to pay back the loan it will cause a huge loss to the bank.

1.2 Purpose

The main purpose of this use-case is to build a predictive model to predict if an applicant is able to repay the loan to the company or not.

2. LITERATURE SURVEY

2.1 Existing Problem

The banks give loans to those applicants who can repay them later. So the banks have to do a thorough background check before lending the loan to an applicant. But it is a tedious task and requires a lot of resources. It will also consume a lot of time for the banks to do the verification process for each applicant. Even after all the verification process they still donot have assurance if the applicant is able to repay the loan.

The existing process of loan application:

Visit the branch of the financial lender. Procure the personal loan application form and enter all the required details. Submit relevant documents that prove

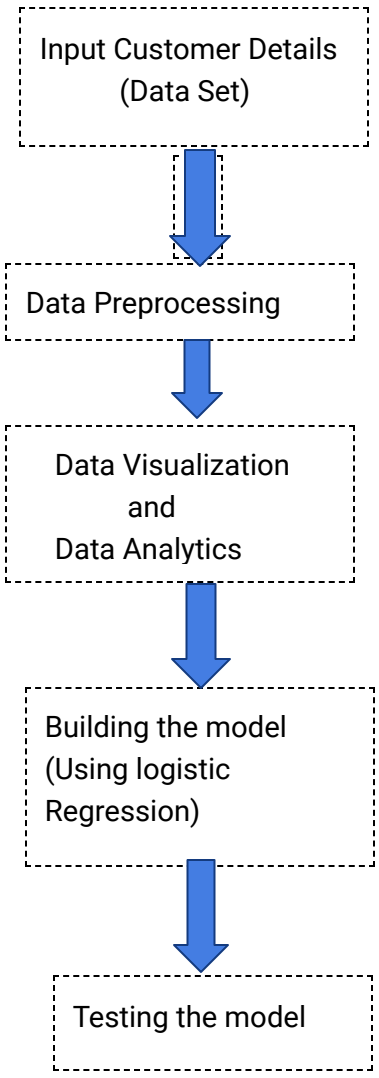
one's income, age, address and identity. The lender will then verify the documents and check the eligibility of the applicant.

2.2 Proposed Solution

The better way is to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, Property details and others. Using a predictive model they can predict if a particular applicant is able to repay the loan or not in the future.

3. THEORITICAL ANALYSIS

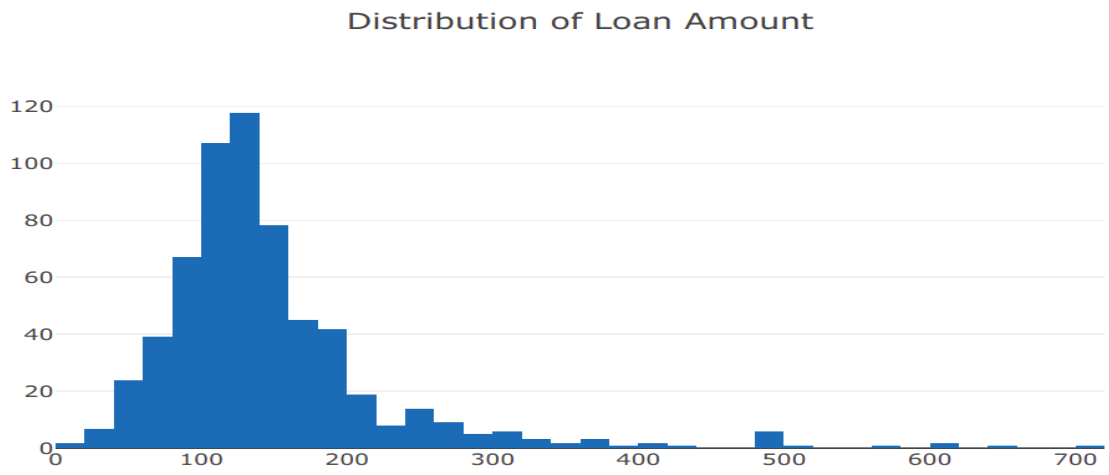
3.1 Block Diagram



3.2 Hardware / Software Designing

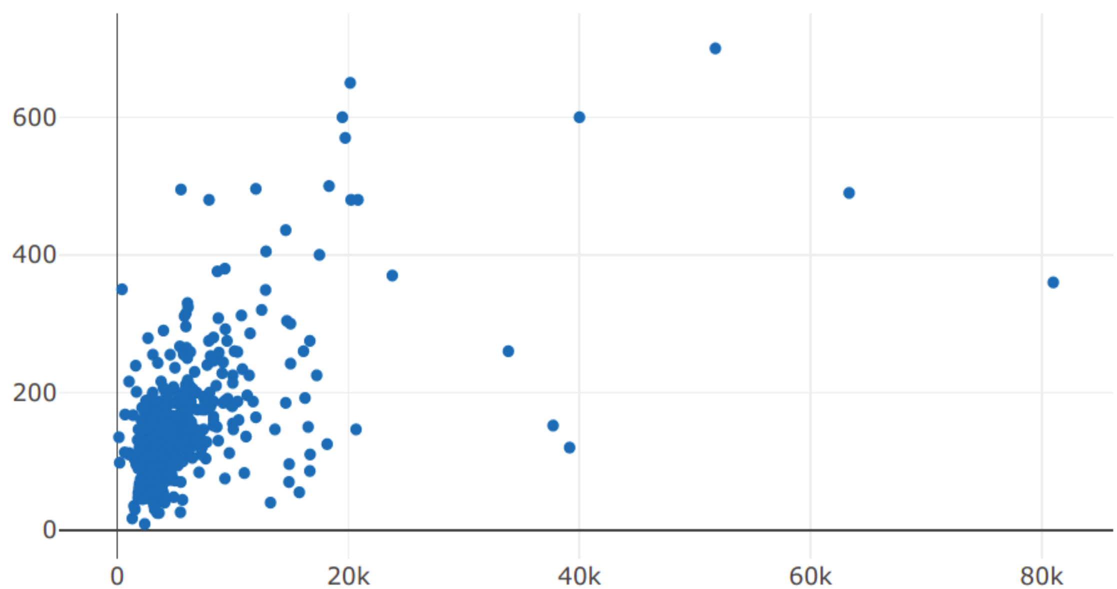
Hardware Required:
Windows 7 and above (64-bit)
RAM **:** 4GB
Processor **:** Minimum Pentium 2 266 MHz processor
Browsers **:** Chrome
Software Required:
Java JDK 10
Weka 3.8.5
Eclipse IDE

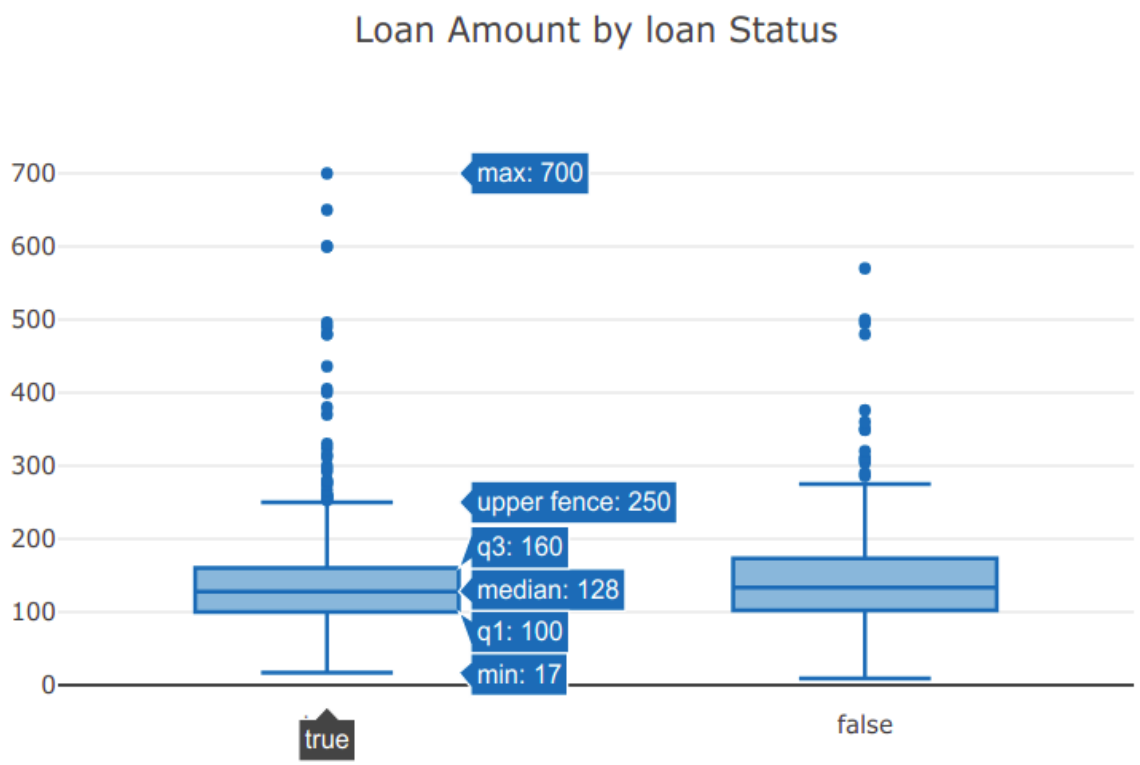
4. EXPERIMENTAL INVESTIGATIONS

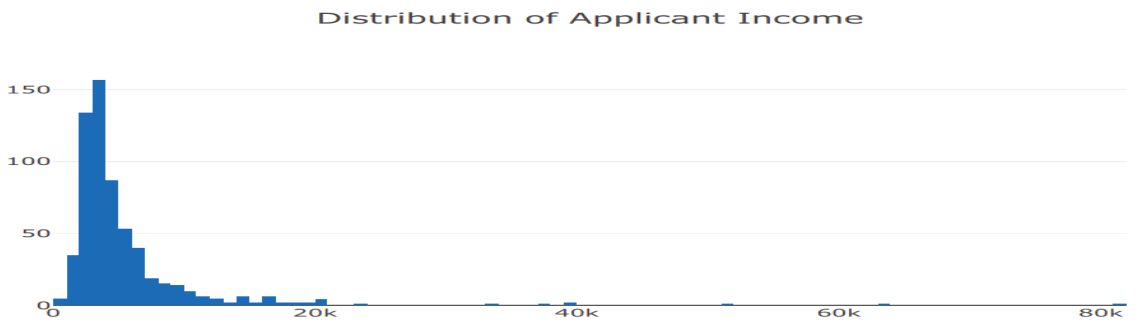


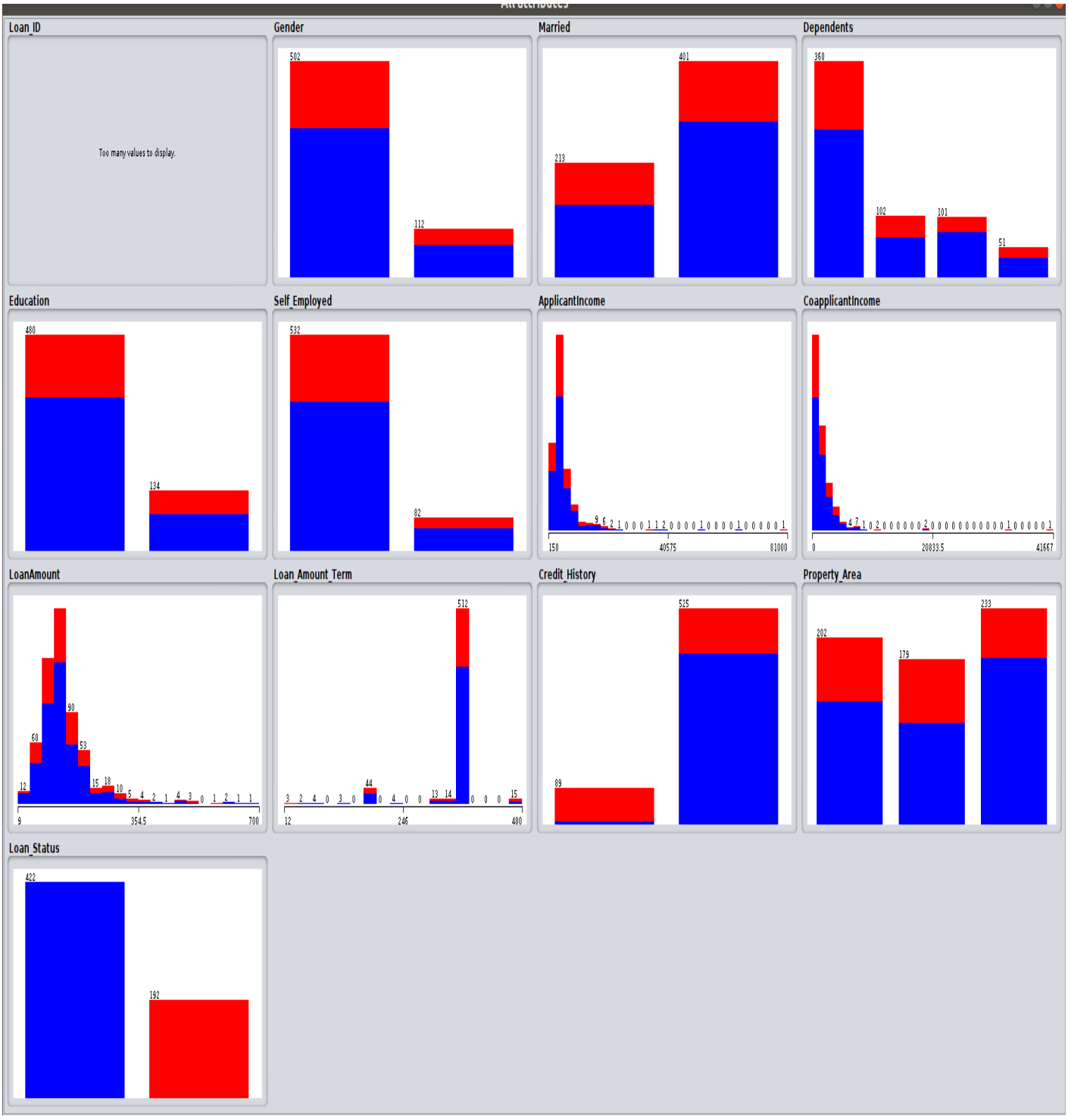
```
<terminated> DataAnalysis [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (05-May-2021, 7:23:53 pm)
Data Visualization
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
614 rows X 13 cols
    Structure of loanpredict_traindata1.csv
 Index  |      Column Name      |  Column Type  |
-----------------------------------------------
     0  |              Loan_ID  |       STRING  |
     1  |               Gender  |       STRING  |
     2  |              Married  |       STRING  |
     3  |           Dependents  |       STRING  |
     4  |            Education  |       STRING  |
     5  |        Self_Employed  |       STRING  |
     6  |      ApplicantIncome  |      INTEGER  |
     7  |    CoapplicantIncome  |       DOUBLE  |
     8  |           LoanAmount  |       DOUBLE  |
     9  |     Loan_Amount_Term  |      INTEGER  |
    10  |       Credit_History  |      INTEGER  |
    11  |        Property_Area  |       STRING  |
    12  |          Loan_Status  |      BOOLEAN  |
                                                                                                  loanpredict_traindata1.csv
  Summary   |  Loan_ID   |  Gender  |  Married  |  Dependents  |  Education  |  Self_Employed  |   ApplicantIncome    |  CoapplicantIncome   |      LoanAmount      |  Loan_Amount_
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
     Count  |       614  |     614  |      614  |         614  |        614  |            614  |                 614  |                 614  |                 614  |
    Unique  |       614  |       2  |        2  |           4  |          2  |              2  |                      |                      |                      |
       Top  |  LP002888  |    Male  |      Yes  |           0  |   Graduate  |             No  |                      |                      |                      |
 Top Freq.  |         1  |     502  |      401  |         360  |        480  |            532  |                      |                      |                      |
       sum  |            |          |           |              |             |                 |             3317724  |         995444.919989  |    89897.06756399997  |               2
      Mean  |            |          |           |              |             |                 |  5403.4592833876195  |  1621.2457980276872  |   146.41216215635174  |  342.00000000
       Min  |            |          |           |              |             |                 |                 150  |                   0  |                   9  |
       Max  |            |          |           |              |             |                 |               81000  |               41667  |                 700  |
     Range  |            |          |           |              |             |                 |               80850  |               41667  |                 691  |
  Variance  |            |          |           |              |             |                 |  37320390.167181246  |   8562929.518385973  |    7062.295974604298  |  4143.8172920
  Std. Dev  |            |          |           |              |             |                 |   6109.041673387181  |   2926.248369223975  |    84.03746768319651  |  64.372488626
     false  |            |          |           |              |             |                 |                      |                      |                      |
      true  |            |          |           |              |             |                 |                      |                      |                      |
Opening in existing browser session.
[14218:14218:0505/192355.089406:ERROR:broker_posix.cc(43)] Invalid node channel message
```

Scatter plot of Loan Amount and Applicant Income

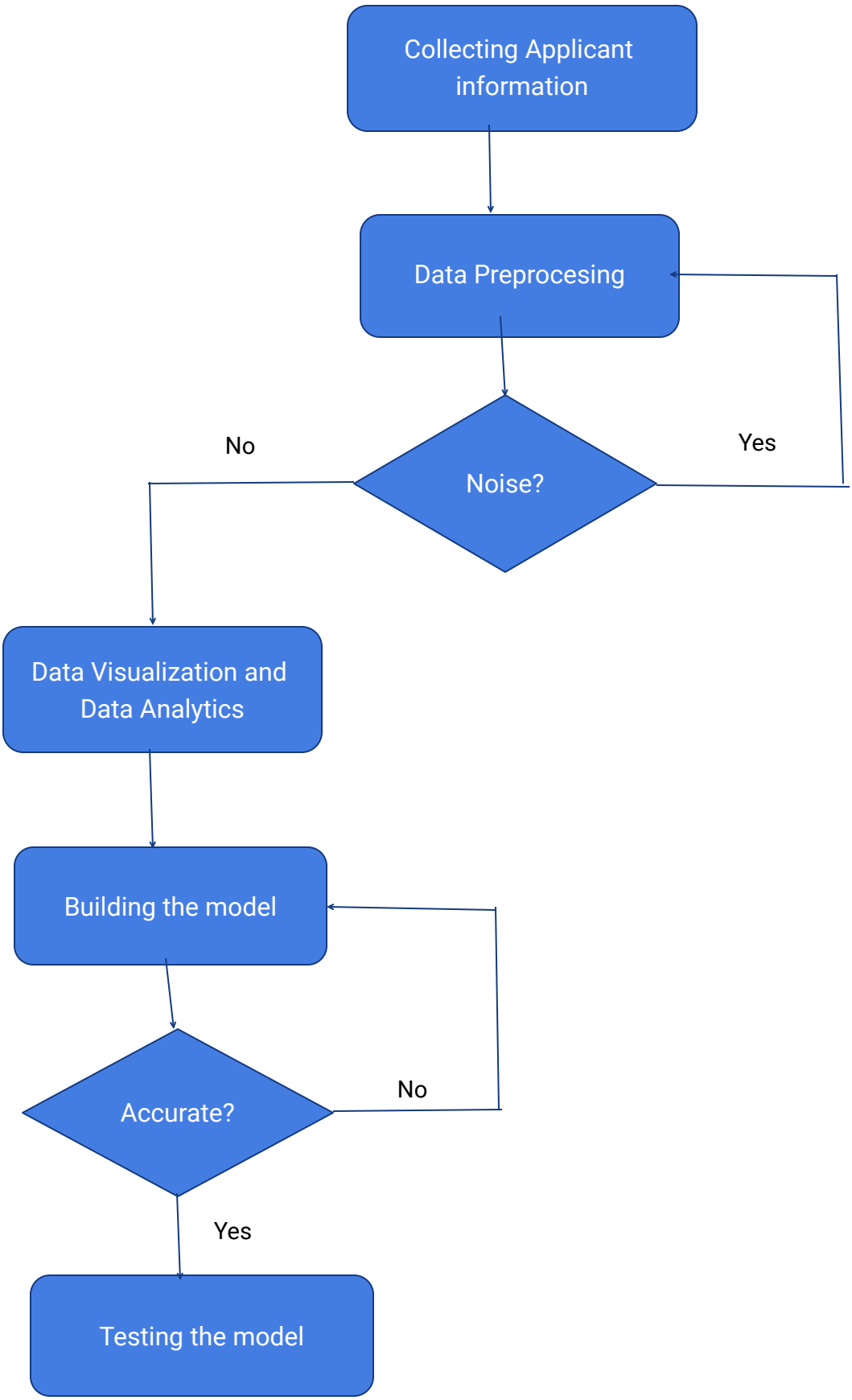



Loan Amount by loan Status
700
600
500
400
300
200
100
0
max: 700
upper fence: 250
q3: 160
median: 128
q1: 100
min: 17
true
false

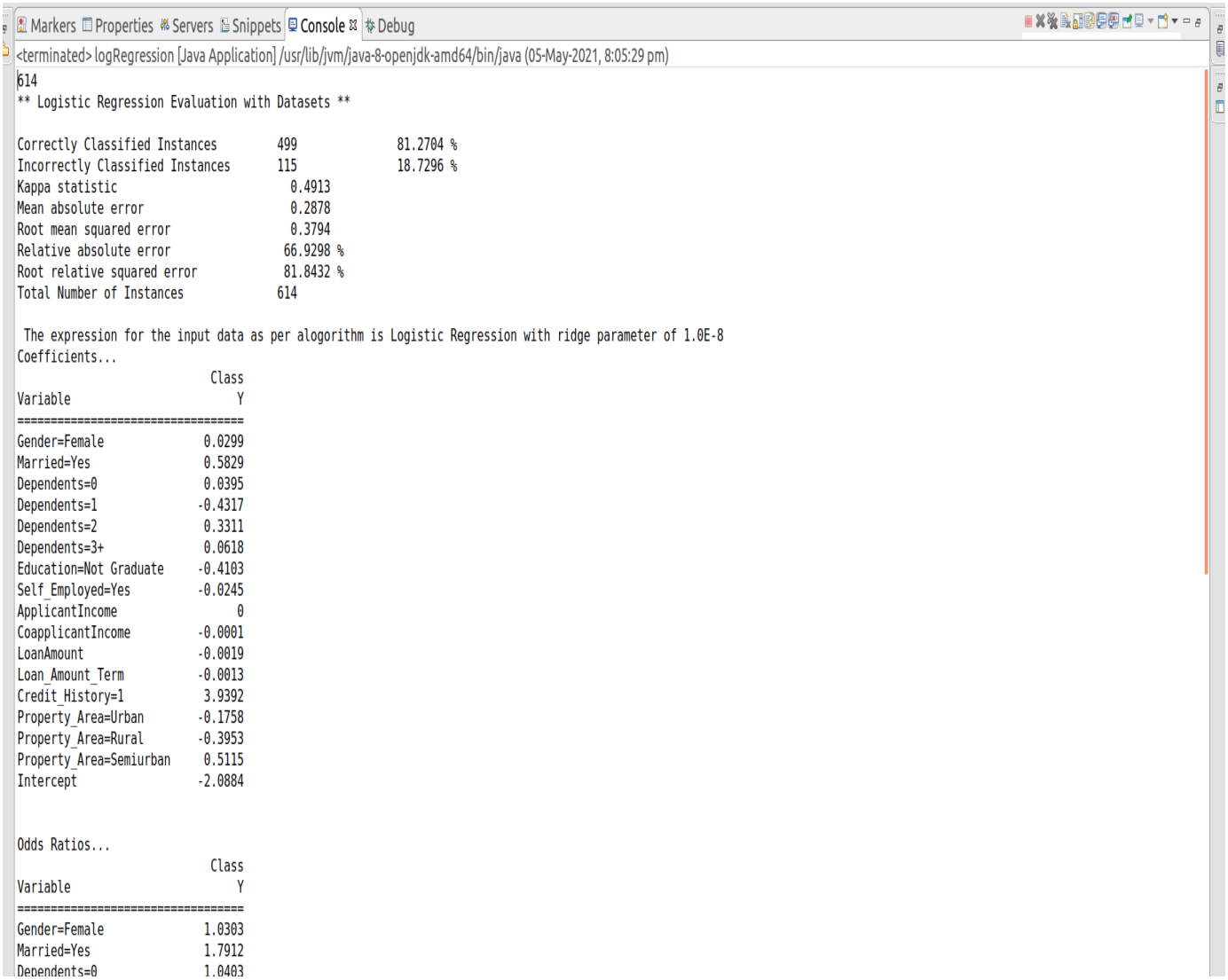

Distribution of Applicant Income
150
100
50
0
0
20k
40k
60k
80k

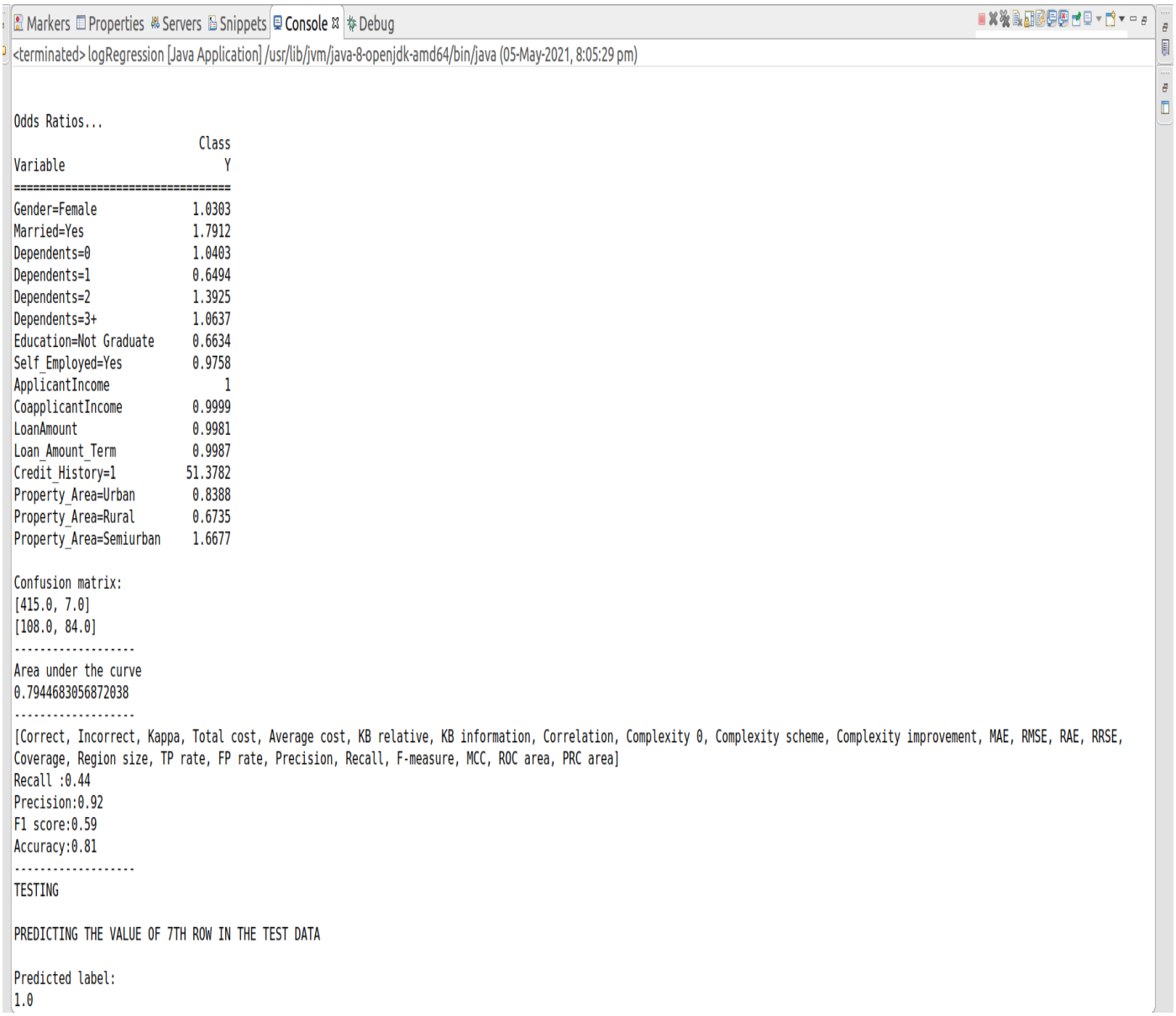

Loan_ID
Too many values to display.
Gender
502
112
Married
213
401
Dependents
360
102
101
51
Education
480
134
Self_Employed
532
82
ApplicantIncome
150
40575
81000
CoapplicantIncome
0
20833.5
41667
LoanAmount
12
60
90
53
15
18
10
5
4
2
1
4
3
0
1
2
1
1
9
354.5
700
Loan_Amount_Term
3
2
4
0
3
0
44
0
4
0
0
13
14
512
0
0
0
15
12
246
480
Credit_History
89
525
Property_Area
202
179
233
Loan_Status
422
192

5. FLOWCHART


Collecting Applicant information
Data Preprocesing
Noise?
No
Yes
Data Visualization and Data Analytics
Building the model
Accurate?
No
Yes
Testing the model

6. RESULT


Markers Properties Servers Snippets Console Debug
<terminated> logRegression [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (05-May-2021, 8:05:29 pm)
614
** Logistic Regression Evaluation with Datasets **
Correctly Classified Instances 499 81.2704 %
Incorrectly Classified Instances 115 18.7296 %
Kappa statistic 0.4913
Mean absolute error 0.2878
Root mean squared error 0.3794
Relative absolute error 66.9298 %
Root relative squared error 81.8432 %
Total Number of Instances 614
The expression for the input data as per alogorithm is Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
Class
Variable Y
Gender=Female 0.0299
Married=Yes 0.5829
Dependents=0 0.0395
Dependents=1 -0.4317
Dependents=2 0.3311
Dependents=3+ 0.0618
Education=Not Graduate -0.4103
Self_Employed=Yes -0.0245
ApplicantIncome 0
CoapplicantIncome -0.0001
LoanAmount -0.0019
Loan_Amount_Term -0.0013
Credit_History=1 3.9392
Property_Area=Urban -0.1758
Property_Area=Rural -0.3953
Property_Area=Semiurban 0.5115
Intercept -2.0884
Odds Ratios...
Class
Variable Y
Gender=Female 1.0303
Married=Yes 1.7912
Dependents=0 1.0403



7. ADVANTAGES AND DISADVANTAGES

Advantages:

1) Loan Prediction is very helpful for employee of banks as well as for the applicant also.

2) It is to provide quick, immediate and easy way to choose the deserving applicants.

3) It automatically calculates the weight of each features taking part in loan processing and on new test data and same features are processed with respect to their associated weight.

Disadvantages:

1) The disadvantage of this model is that it emphasize different weights to each factor but in real life sometime loan can be approved on the basis of single strong factor only, which is not possible through this system.

2) It is not 100% accurate, so there may be some cases where the prediction can go wrong.

8. APPLICATIONS

Hotel Booking

They try to predict users' intentions and recognize entities. Where will you go, where do you prefer to stop, what are you planning to do? Some predictions are made even if the user didn't type anything in the search line yet.

Creditcard Fraud Prediction:

Detecting fraud transactions is of great importance for any credit card company. To detect potential frauds so that customers are not charged for items that they did not purchase we can build a classifier that tells if a

transaction is a fraud or not.

Medical Insuarance Prediction

It predicts the risk for medical insurance and identify those patients most at risk of being re-admitted.

9. CONCLUSION

This study has proposed a comprehensive research and model development for the prediction of the default loans.
We have built predictive model using logistic Regression to predict if a applicant is eligible to take the loan from the bank. It takes attributes such as Gender, Income, Property, Credit history etc and tells if a person is eligible or not.
The accuracy of the model built is 81% which means that the predictions made are correct 81 times in 100 times.
Recall : 0.44
Precision : 0.92
F score : 0.59
Accuracy : 0.81
Area under the curve : 0.794468

10. FUTURE SCOPE

This project can be enhanced by improving the factors such as accuracy, recall, precision and F-score .

11. BIBILOGRAPHY

https://towardsdatascience.com/predict-loan-eligibility-using-machine-learning-models-7a14ef904057
https://machinelearningmastery.com/save-machine-learning-model-make-predictions-weka/
https://dzone.com/articles/beautiful-java-visualization-with-tablesaws-plotly
https://towardsdatascience.com/ml-basics-loan-prediction-d695ba7f31f6

12. APPENDIX

Code for data visualization:

```
package org.ml;
import java.io.IOException;
import tech.tablesaw.api.Table;
import tech.tablesaw.plotly.Plot;
import tech.tablesaw.plotly.api.AreaPlot;
import tech.tablesaw.plotly.api.BubblePlot;
import tech.tablesaw.plotly.api.ScatterPlot;
import tech.tablesaw.plotly.components.Figure;
import tech.tablesaw.plotly.components.Layout;
import tech.tablesaw.plotly.traces.BarTrace;
import tech.tablesaw.plotly.traces.BoxTrace;
import tech.tablesaw.plotly.traces.Histogram2DTrace;
import tech.tablesaw.plotly.traces.HistogramTrace;
import tech.tablesaw.plotly.traces.ScatterTrace;
public class DataAnalysis {
public static void main(String args[]) {
	System.out.println("Data Visualization");
	try {
		Table data =
Table.read().csv("/home/joshita/eclipse-workspace/org.ml/src/main/java/org/ml/loanpredict_traindata_after_preprocess.csv");
		System.out.println(data.shape());
		System.out.println(data.structure());
		System.out.println(data.summary());
		//HISTOGRAM
		 Layout layout1 = Layout.builder().title("Distribution of Loan Amount").build();
		HistogramTrace trace1 = HistogramTrace.builder(data.nCol("LoanAmount")).build();
```

```
            Plot.show(new Figure(layout1, trace1));
            Layout layout4 = Layout.builder().title("Distribution of Applicant Income").build();
            HistogramTrace trace4 = HistogramTrace.builder(data.nCol("ApplicantIncome")).build();
            Plot.show(new Figure(layout4, trace4));
            Layout layout7 = Layout.builder().title("Distribution of Co-Applicant Income").build();
            HistogramTrace trace7 = HistogramTrace.builder(data.nCol("CoapplicantIncome")).build();
            Plot.show(new Figure(layout7, trace7));

           //BOX PLOT
            Layout layout2 = Layout.builder().title("Loan Amount by loan Status").build();
            BoxTrace trace2 =
BoxTrace.builder(data.categoricalColumn("Loan_Status"),data.nCol("LoanAmount")).build();
                                                                    Plot.show(new Figure(layout2, trace2));

 //SCATTER PLOT
Layout layout3 = Layout.builder().title("Scatter plot of Loan Amount and Applicant Income").build();
ScatterTrace trace3 = ScatterTrace.builder(data.nCol("ApplicantIncome"),data.nCol("LoanAmount")).build();
Plot.show(new Figure(layout3, trace3));

 //BAR GRAPH
Layout layout5 = Layout.builder().title("Bar Graph").build();
BarTrace trace5 =
BarTrace.builder(data.categoricalColumn("Property_Area"),data.nCol("LoanAmount")).build();
                                                                    Plot.show(new Figure(layout5,trace5));

 //2D HISTOGRAM
Layout layout6 = Layout.builder().title("2D Histogram for Applicant income and Coapplicant Income").build();
 Histogram2DTrace trace6 = Histogram2DTrace.builder(data.nCol("ApplicantIncome"),
data.nCol("CoapplicantIncome")).build();
Plot.show(new Figure(layout6,trace6));
}
catch (IOException e) {
        // TODO Auto-generated catch block
          e.printStackTrace();
      }
   }
}
```

Code for Model building:

```
package org.ml;
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 logRegression {
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("/home/joshita/eclipse-workspace/org.ml/src/main/java/org/ml/loanpredict_traindata_after_
preprocess.arff");
Instances test_data =
getInstances("/home/joshita/eclipse-workspace/org.ml/src/main/java/org/ml/loanpredict_testdata_after_
preprocess.arff");
System.out.println(train_data.size());

Classifier classifier = new weka.classifiers.functions.Logistic();
classifier.buildClassifier(train_data);
Evaluation 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)/10
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("-------------------");
System.out.println("TESTING");
System.out.println();
System.out.println("PREDICTING THE VALUE OF 7TH ROW IN THE TEST DATA");
System.out.println();
Instance predicationDataSet = test_data.get(7);
double value = classifier.classifyInstance(predicationDataSet);
/** Prediction Output */
System.out.println("Predicted label:");
System.out.print(value);
    }
}
```