

INTRODUCTION:

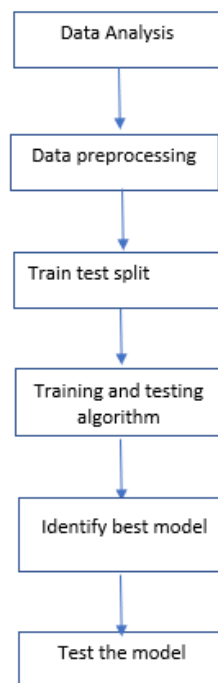
Banknotes are one of the most important assets of a country. Some miscreants introduce fake notes which bear a resemblance to original note to create discrepancies of the money in the financial market. Fake money is an imitation of the genuine notes and is created illegally for various motives. These fake notes are created in all denominations which brings the financial market of the country to a low level. It is difficult for humans to tell true and fake banknotes apart especially because they have a lot of similar features. Fake notes are created with precision, hence there is need for an efficient algorithm which accurately predicts whether a banknote is genuine or not. Accurate separation of original notes from the forged one is needed. With this project we can build an efficient bank note authentication system.

LITERATURE SURVEY:

Preserving genuineness of higher denomination printed Banknotes is one of the critical issues. It has the major role in financial activities of every country. In the present work NN has been trained using Genetic Algorithm. Rule-based as well as statistical techniques are commonly used for solving classification problems. Machine learning algorithms fall in the category of statistical techniques. So, I propose banknote authentication system using machine learning algorithms.

THEORETICAL ANALYSIS:

BLOCK DIAGRAM:



Software requirements of project are Eclipse IDE and Weka tool.

EXPERIMENTAL INVESTIGATIONS:

Analysis made:

Attribute name	Value Type	Description
Variance	Double	It is a measure of the 'spread' of a distribution about its average value.
Skewness	Double	Skewness tells about the direction of variation of the lack of symmetry.
Kurtosis	Double	Kurtosis is a parameter that describes the peakedness of distribution
Entropy	Double	Image entropy is the amount of information which must be coded for, by a compression algorithm.
Class	Nominal	Class contains two values namely 0 and 1 where 0 represents genuine banknotes and 1 represents fake banknotes.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

$$\text{F1 score} = (2 * \text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

where,

True Positive (TP) = the number of cases correctly identified as genuine notes.

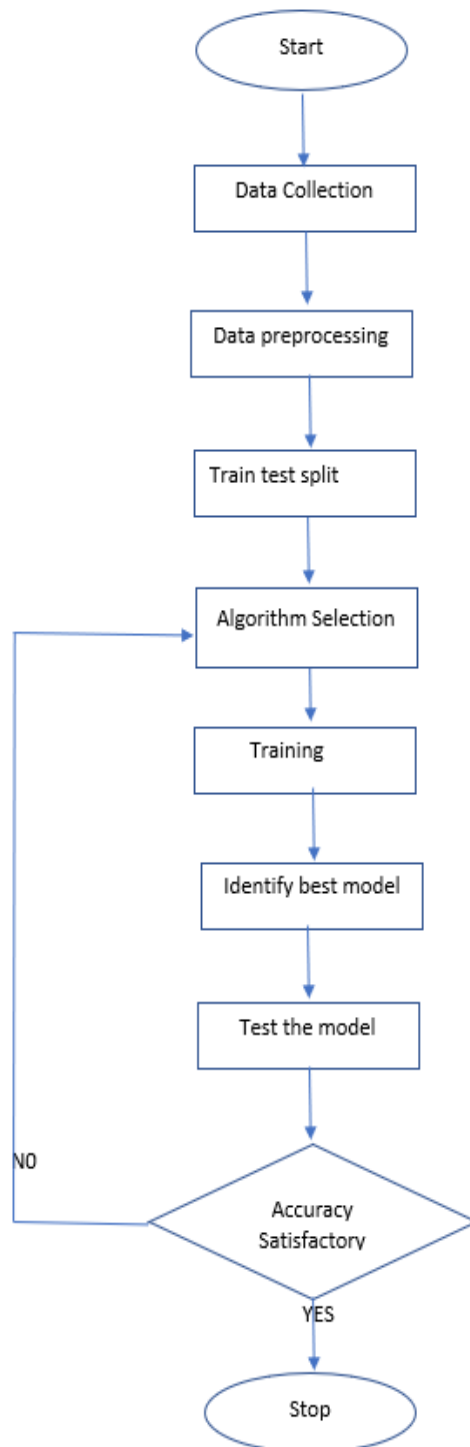
True negative (TN) = the number of cases correctly identified as fake notes.

False positive (FP) = the number of cases incorrectly identified as genuine notes.

False negative (FN) = the number of cases incorrectly identified as fake notes.

- Hold-out method is used which divides the dataset into the ratio of 80:20 (training data: test data) .
- Area under curve is graphical plot shows how classifier and threshold choices perform . This curve illustrates the ability of binary classifier system as its discrimination threshold is varied.
- With the help of confusion matrix Recall, Precision, F1 Score, Accuracy can be calculated and recognition of forged notes among the genuine ones can be done.

FLOWCHART:



RESULT:

The screenshot displays the Eclipse IDE interface for a Maven project named 'org.ml'. The Project Explorer on the left shows the project structure, including source files and dependencies. The main editor shows the 'pom.xml' file, which defines the project's metadata and dependencies. The console at the bottom shows the output of a Maven build, indicating a successful installation and compilation.

Project Structure (Project Explorer):

- org.ml
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - Maven Dependencies
 - src
 - main
 - java
 - org
 - ml
 - data_banknote_authentication_testing.arff
 - data_banknote_authentication_testing1.arff
 - data_banknote_authentication_training.arff
 - data_banknote_authentication_training1.arff
 - data_banknote_authentication.csv
 - DataAnalysis.java
 - LinearReg.java
 - LogisticRegression.java
 - testdata.csv
 - training data.csv
 - ml.arff
 - resources
 - test
 - target
 - testoutput
 - pom.xml

pom.xml Content:

```
1<?xml version="1.0" encoding="UTF-8"?>
2<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4.0.0.xsd">
4  <groupId>com</groupId>
5  <artifactId>org.ml</artifactId>
6  <version>0.0.1-SNAPSHOT</version>
7  <dependencies>
8    <groupId>nz.ac.waikato.cms.weka</groupId>
9    <artifactId>weka-stable</artifactId>
10   <version>3.8.0</version>
11  </dependency>
12  <dependency>
13    <groupId>tech.tablesaw</groupId>
14    <artifactId>tablesaw-core</artifactId>
15    <version>0.38.1</version>
16  </dependency>
17  <dependency>
18    <groupId>tech.tablesaw</groupId>
19    <artifactId>tablesaw-jplot</artifactId>
20    <version>0.38.1</version>
21  </dependency>
22  <!-- Thanks for using https://jar-download.com -->
23  </dependencies>
24  </project>
25
26  <properties>
27    <maven.compiler.source>1.8</maven.compiler.source>
28    <maven.compiler.target>1.8</maven.compiler.target>
29  </properties>
30</project>
```

Console Output:

```
<terminated> C:\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.15.0.2\v20210201-0955\jre\bin\javaw.exe (07-May-2021, 9:56:42 am)
[INFO] --- maven-install-plugin:2.4:install (default-install) @ org.ml ---
[INFO] Installing C:\eclipse\org.ml\target\org.ml-0.0.1-SNAPSHOT.jar to C:\Users\keert\.m2\repository\com\org.ml\0.0.1-SNAPSHOT\org.ml-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\eclipse\org.ml\pom.xml to C:\Users\keert\.m2\repository\com\org.ml\0.0.1-SNAPSHOT\org.ml-0.0.1-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.959 s
[INFO] Finished at: 2021-05-07T09:56:46+05:30
[INFO] -----
```

eclipse - org.ml/src/main/java/org/ml/DataAnalysis.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> DataAnalysis (1) [Java Application] C:\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (07-May-2021, 9:58:26 am)

DataAnalysis

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.

1372 rows X 5 cols

Variance	Skewness	Curtosis	Entropy	Class
3.6216	8.6661	-2.8073	-0.44699	0
4.5459	8.1674	-2.4586	-1.4621	0
3.866	2.6383	1.9242	0.10645	0
3.4566	9.5228	-4.0112	-3.5944	0

data_banknote_authentication.csv

Variance	Skewness	Curtosis	Entropy	Class
-1.3887	-4.8773	6.4774	0.34179	1
-3.7583	-13.4586	17.5932	-2.7771	1
-3.5637	-8.3827	12.393	-1.2823	1
-2.5419	-0.65804	2.6842	1.1952	1

Structure of data_banknote_authentication.csv

Index	Column Name	Column Type
0	Variance	DOUBLE
1	Skewness	DOUBLE
2	Curtosis	DOUBLE
3	Entropy	DOUBLE
4	Class	INTEGER

data_banknote_authentication.csv

Summary	Variance	Skewness	Curtosis	Entropy	Class
Count	1372	1372	1372	1372	1372
Sum	595.084772699998	2637.468481520802	1917.5444048908017	-1634.952745499989	610
Mean	0.433735257069696	1.9223531206414035	1.397627172667638	-1.1916565200437328	0.44460641399416895
Min	-7.0421	-13.7731	-5.2861	-8.5482	0
Max	6.8248	12.9516	17.9274	2.4495	1
Range	13.8669	26.7247	23.2135	10.9977	1
Variance	8.881299121945154	34.44570967967512	18.576359377624264	4.414256203357667	0.2471116611696257
Std. Dev	2.8427625862785577	5.86904674369485	4.310030090106595	2.101013137359609	0.4971032701256688

eclipse - org.ml/src/main/java/org/ml/LinearReg.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- org.ml
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - JRE System Library [J2SE-1.5]
 - Maven Dependencies
 - src
 - main
 - java
 - org
 - ml
 - data_banknote_authentication_testing.arff
 - data_banknote_authentication_training1.arff
 - data_banknote_authentication_training.arff
 - data_banknote_authentication_training1.arff
 - data_banknote_authentication.csv
 - DataAnalysis.java
 - LinearReg.java
 - LogisticRegression.java
 - testdata.csv
 - training data.csv
 - ml.arff
 - resources
 - test
 - target
 - testoutput
 - pom.xml

org.ml/pom.xml

```

1 package org.ml;
2 import java.io.IOException;
3
4
5
6
7
8 public class LinearReg {
9
10     public static void main(String[] args) throws Exception {
11         DataSource source = new DataSource("C:\\\\eclipse\\\\org.ml\\\\src\\\\main\\\\java\\\\org\\\\ml\\\\
12         Instances dataset=source.getDataSet();
13         dataset.setClassIndex(dataset.numAttributes()-1);
14         //Linear Regression
15         LinearRegression lr=new LinearRegression();
16         lr.buildClassifier(dataset);
17
18         Evaluation lreval =new Evaluation(dataset);
19         lreval.evaluateModel(lr,dataset);
20         System.out.println(lreval.toSummaryString());
21
22     }
23
24 }
25

```

org.ml

- LinearReg
- main(String[] args) : void

Markers Properties Servers Data Source Explorer Snippets Console

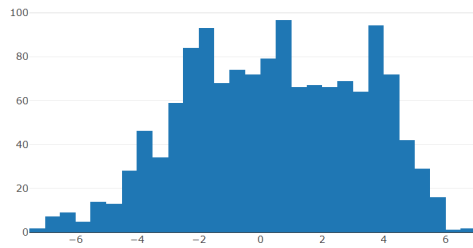
<terminated> LinearReg [Java Application] C:\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (07-May-2021, 10:00:58 am - 10:01:05 am)

INFO: already loaded netlib-native_ref-win-x86_64.dll

Correlation coefficient 0.93
 Mean absolute error 0.1354
 Root mean squared error 0.1827
 Relative absolute error 27.4233 %
 Root relative squared error 36.7632 %
 Total Number of Instances 1372

Writeable Smart Insert 25:1:815

Distribution of VARIANCE



Type here to search

eclipse - org.ml/src/main/java/org/ml/LogisticRegression.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Markers Properties Servers Data Source Explorer Snippets Console

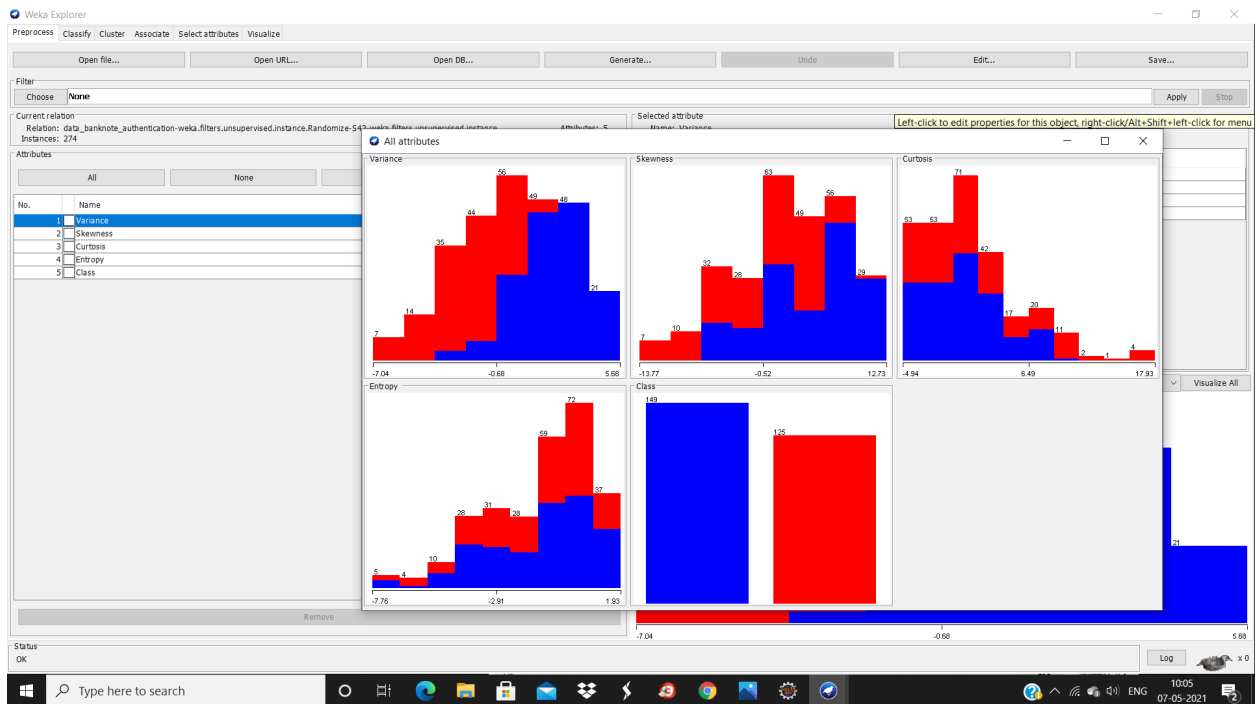
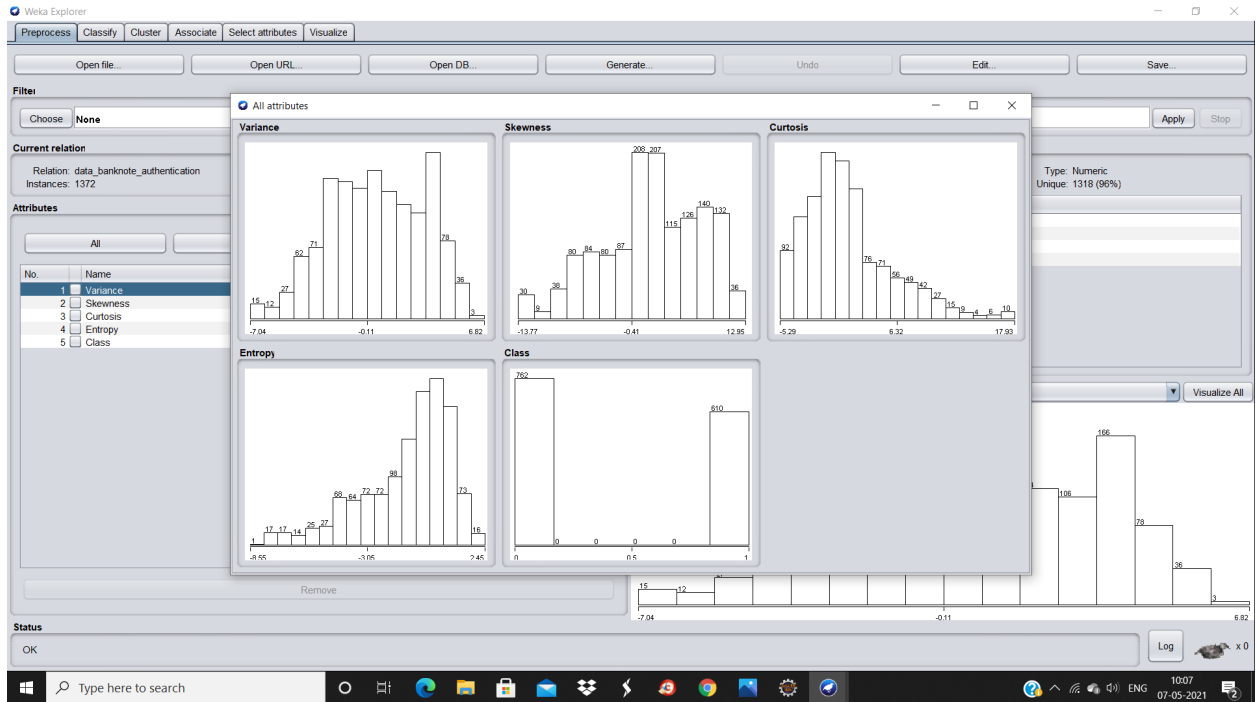
<terminated> LogisticRegression [Java Application] C:\eclipse\org.ml\testoutput\outputbe3e0885-4b29-440f-b9b4-f1e7bb7fe59.html (07-May-2021, 10:02:21 am)

```
1098
** Logistic Regression Evaluation with Datasets **
Correctly Classified Instances      271      98.9051 %
Incorrectly Classified Instances    3      1.0949 %
Kappa statistic                    0.9779
Mean absolute error                 0.0123
Root mean squared error             0.0898
Relative absolute error             2.4887 %
Root relative squared error         18.0217 %
Total Number of Instances          274

Confusion matrix:
[148.0, 1.0]
[2.0, 123.0]
-----
Area under the curve
0.9997852348993289
-----
[Correct, Incorrect, Kappa, Total cost, Average cost, KB relative, KB information, Correlation, Complexity 0, Complexity scheme, Complexity improvement, MAE, RMSE, RAE, RRSE, Coverage, Region size, TP ra
Recall:0.98
Precision:0.99
F1 score:0.99
Accuracy:0.99
-----
Predicted label:
1.0
```

Type here to search

1002 07-05-2021





ADVANTAGES :

- Fake notes in the market can be reduced.
- security features of bank note can be improved.

DISADVANTAGES:

- In my banknote authentication system accuracy is 99%.

APPLICATIONS:

- Safe transactions
- Illegal stocking of products can be reduced
- Inflation can be reduced by recognising counterfeit currency which may affect common man adversely

CONCLUSION:

Banknote authentication is an important task. It is difficult to detect fake banknote. Machine learning algorithms can help in this regard. With the help of logistic regression evaluation with data sets, confusion matrix and area under curve specified instance can be predicted with the accuracy of 99%.

FUTURE SCOPE:

In future, this work can be extended by categorizing the notes into different categories as Genuine, Low-Quality forgery, High-Quality forgery, Inappropriate ROI.

BIBLIOGRAPHY:

- VSH solutions wesite
- International journal of Soft computing and Engineering
- Sensors Review
- https://www.researchgate.net/post/Where_must_we_use_variance_and_mean_of_image

Code:

```

1 <project xmlns="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">
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>com</groupId>
4   <artifactId>org.ml</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <dependencies>
7     <dependency>
8       <groupId>nz.ac.waikato.cms.weka</groupId>
9       <artifactId>weka-stable</artifactId>
10      <version>3.8.0</version>
11    </dependency>
12    <dependency>
13      <groupId>tech.tablesaw</groupId>
14      <artifactId>tablesaw-core</artifactId>
15      <version>0.38.1</version>
16    </dependency>
17    <dependency>
18      <groupId>tech.tablesaw</groupId>
19      <artifactId>tablesaw-jsplot</artifactId>
20      <version>0.38.1</version>
21    </dependency>
22    <!-- Thanks for using https://jar-download.com -->
23
24  </dependencies>
25
26  <properties>
27    <maven.compiler.source>1.8</maven.compiler.source>

```

```
28         <maven.compiler.target>1.8</maven.compiler.target>
29     </properties>
30 </project>
```

Data Analysis:

```
1 package org.ml;
2 import java.io.IOException;
3 import tech.tablesaw.api.Table;
4 import tech.tablesaw.plotly.Plot;
5 import tech.tablesaw.plotly.components.Figure;
6 import tech.tablesaw.plotly.components.Layout;
7 import tech.tablesaw.plotly.traces.HistogramTrace;
8
9 public class DataAnalysis {
10
11     public static void main(String args[])
12     {
13         System.out.println("DataAnalysis");
14
15         try {
16             Table bank_data =
17 Table.read().csv("C:\\eclipse\\org.ml\\src\\main\\java\\org
18 \\ml\\data_banknote_authentication.csv");
19             System.out.println(bank_data.shape());
20
21             System.out.println(bank_data.first(4));
22             System.out.println(bank_data.last(4));
23
24             System.out.println(bank_data.structure());
25
26             System.out.println(bank_data.summary());
27
28
29         } catch (IOException e) {
```

```

30         e.printStackTrace();
31     }
32 }
33 }

```

Linear Regression:

```

1     package org.ml;
2     import java.io.IOException;
3     import weka.classifiers.Evaluation;
4     import weka.classifiers.functions.LinearRegression;
5     import weka.core.Instances;
6     import weka.core.converters.ConverterUtils.DataSource;
7
8     public class LinearReg {
9         public static void main(String[] args) throws Exception
10        {
11            DataSource source =new
12            DataSource("C:\\\\eclipse\\\\org.ml\\\\src\\\\main\\\\java\\\\org
13            \\\\ml\\\\data_banknote_authentication.csv");
14            Instances dataset=source.getDataSet();
15            dataset.setClassIndex(dataset.numAttributes()-1);
16            //linear Regression
17            LinearRegression lr=new LinearRegression();
18            lr.buildClassifier(dataset);
19
20            Evaluation lreval =new Evaluation(dataset);
21            lreval.evaluateModel(lr,dataset);
22            System.out.println(lreval.toSummaryString());
23
24        }
25    }

```

Logistic Regression:

```

1      package org.ml;
2      import java.util.Arrays;
3
4      import weka.classifiers.Classifier;
5      import weka.classifiers.evaluation.Evaluation;
6      import weka.core.Instance;
7      import weka.core.Instances;
8      import weka.core.converters.ConverterUtils.DataSource;
9      public class LogisticRegression{
10         public static Instances getInstances (String filename)
11         {
12
13             DataSource source;
14             Instances dataset = null;
15             try {
16                 source = new DataSource(filename);
17                 dataset = source.getDataSet();
18
19                 dataset.setClassIndex(dataset.numAttributes()-1);
20
21             } catch (Exception e) {
22                 // TODO Auto-generated catch block
23                 e.printStackTrace();
24
25             }
26
27             return dataset;
28         }
29
30         public static void main(String[] args) throws
31         Exception{
32             Instances train_data =
33             getInstances("C:\\eclipse\\org.ml\\src\\main\\java\\org\\ml\\data
34             _banknote_authentication_training1.arff");
35             Instances test_data =
36             getInstances("C:\\eclipse\\org.ml\\src\\main\\java\\org\\ml\\data
37             _banknote_authentication_testing1.arff");
38             System.out.println(train_data.size());

```

```

35
36         /** Classifier here is Linear Regression */
37         Classifier classifier = new
weka.classifiers.functions.Logistic();
38         /** */
39         classifier.buildClassifier(train_data);
40
41
42         /**
43         * train the algorithm with the training data and
evaluate the
44         * algorithm with testing data
45         */
46         Evaluation eval = new Evaluation(train_data);
47         eval.evaluateModel(classifier, test_data);
48         /** Print the algorithm summary */
49         System.out.println("** Logistic Regression
Evaluation with Datasets **");
50         System.out.println(eval.toSummaryString());
51 //         System.out.print(" the expression for the input
data as per algorithm is ");
52 //         System.out.println(classifier);
53
54         double confusion[][] = eval.confusionMatrix();
55         System.out.println("Confusion matrix:");
56         for (double[] row : confusion)
57             System.out.println( Arrays.toString(row));
58         System.out.println("-----");
59
60         System.out.println("Area under the curve");
61         System.out.println( eval.areaUnderROC(0));
62         System.out.println("-----");
63
64
65         System.out.println(eval.getAllEvaluationMetricNames());
66
67         System.out.print("Recall :");
68
69         System.out.println(Math.round(eval.recall(1)*100.0)/100.0);

```

```
69         System.out.print("Precision:");
70
71         System.out.println(Math.round(eval.precision(1)*100.0)/100.0);
72         System.out.print("F1 score:");
73
74         System.out.println(Math.round(eval.fMeasure(1)*100.0)/100.0);
75
76         System.out.print("Accuracy:");
77         double acc = eval.correct()/(eval.correct()+
78 eval.incorrect());
79         System.out.println(Math.round(acc*100.0)/100.0);
80
81         System.out.println("-----");
82         Instance predicationDataSet = test_data.get(2);
83         double value =
84 classifier.classifyInstance(predicationDataSet);
85         /** Prediction Output */
86         System.out.println("Predicted label:");
87         System.out.print(value);
88     }
89 }
```