INTRODUCTION:

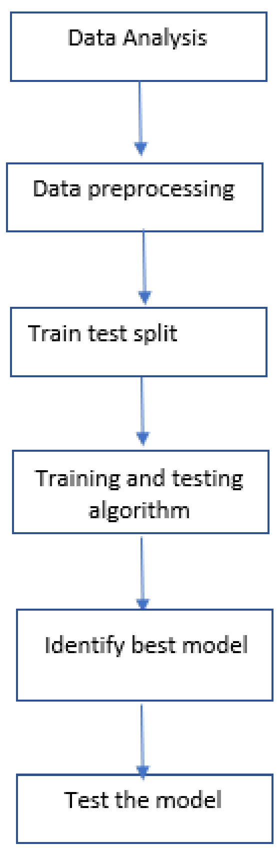
Lot of miscreants induces fake notes into the market which resemble exactly the original note. Accurate seperation of original notes from the forged one is needed.With this project we can build a efficient bank note authentication system.

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 presesnt work NN has been trained using Genetic Algorithm.Rule_based as well as statistical techniques are commomly used for solving classifiaction 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:

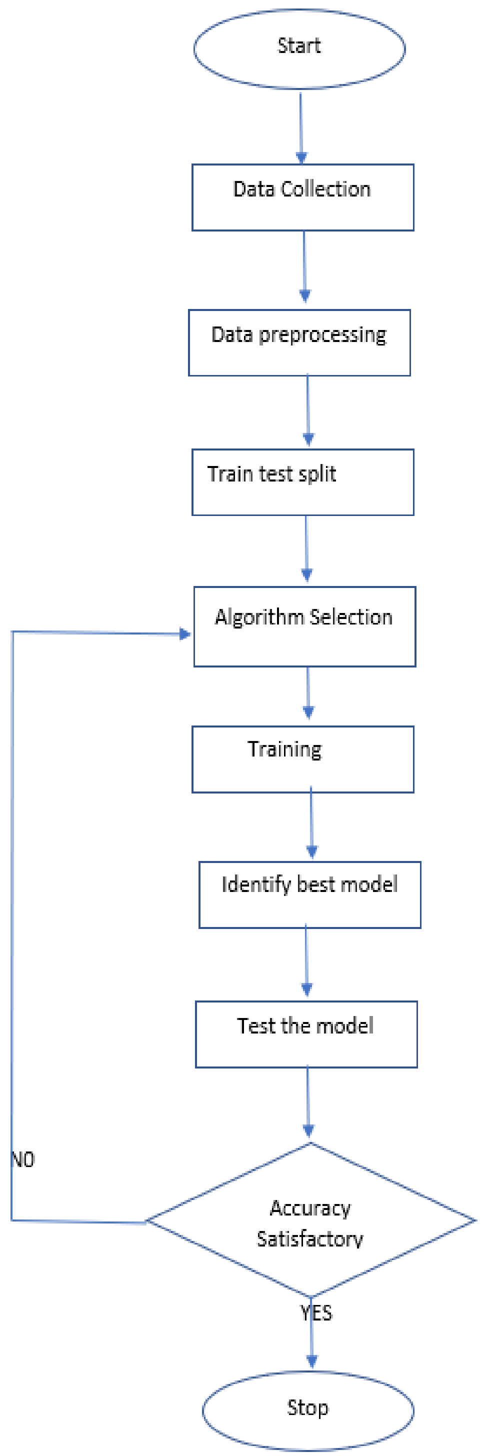
Analysis made:

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.

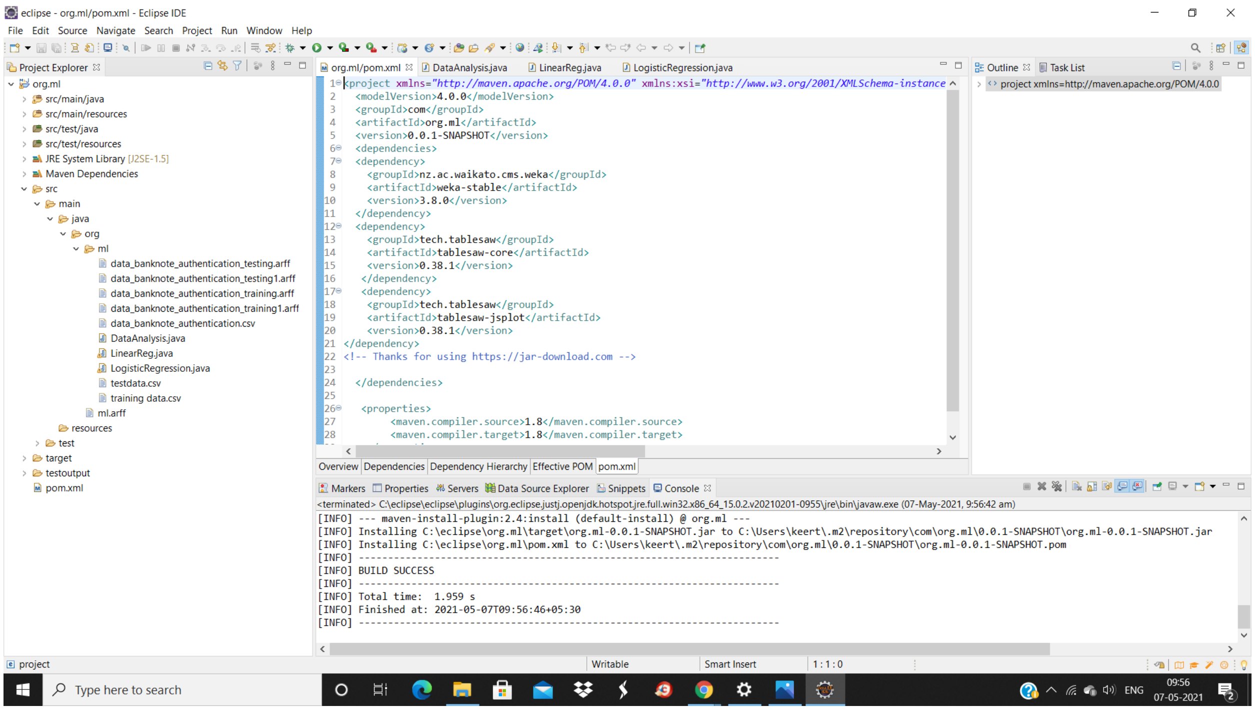
Accuracy = (TP +TN)/(TP+TN + FP+ FN)
Precision = TP/(TP + FP)
Recall = TP/(TP +TN)
F1 score=(2*precision*recall)/(precision+recall)
where,
True Positive (TP) = the number of cases correctly identified as genuine notes.
True negative (TN) = the number of cases correctly identified as fake notes.
False positive (FP) = the number of cases incorrectly identified as genuine notes.
False negative (FN) = the number of cases incorrectly identified as fake notes.

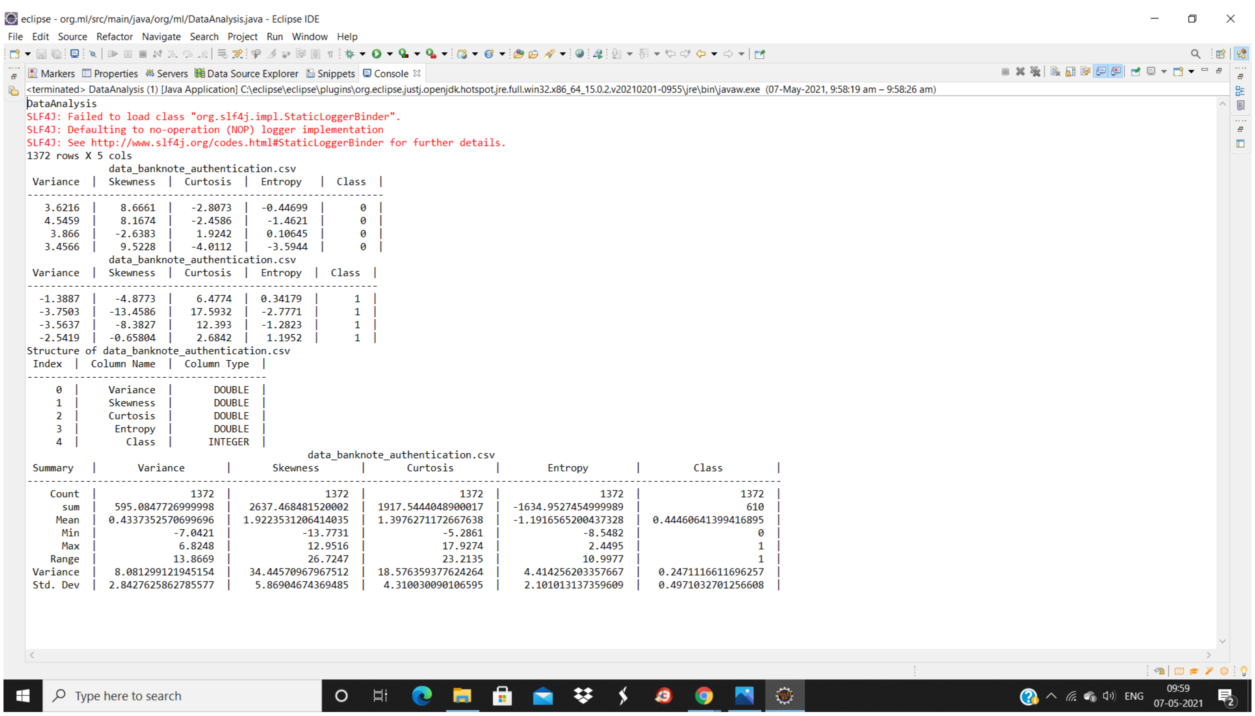
- Hold-out method is used which divides the dataset into the ratio of 80:20 (training data: test data) .
- Area under curve is graphical plot shows how classifier and threshold choices perform . This curve illustrates the ability of binary classifier system as its discrimination threshold is varied.
- 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.

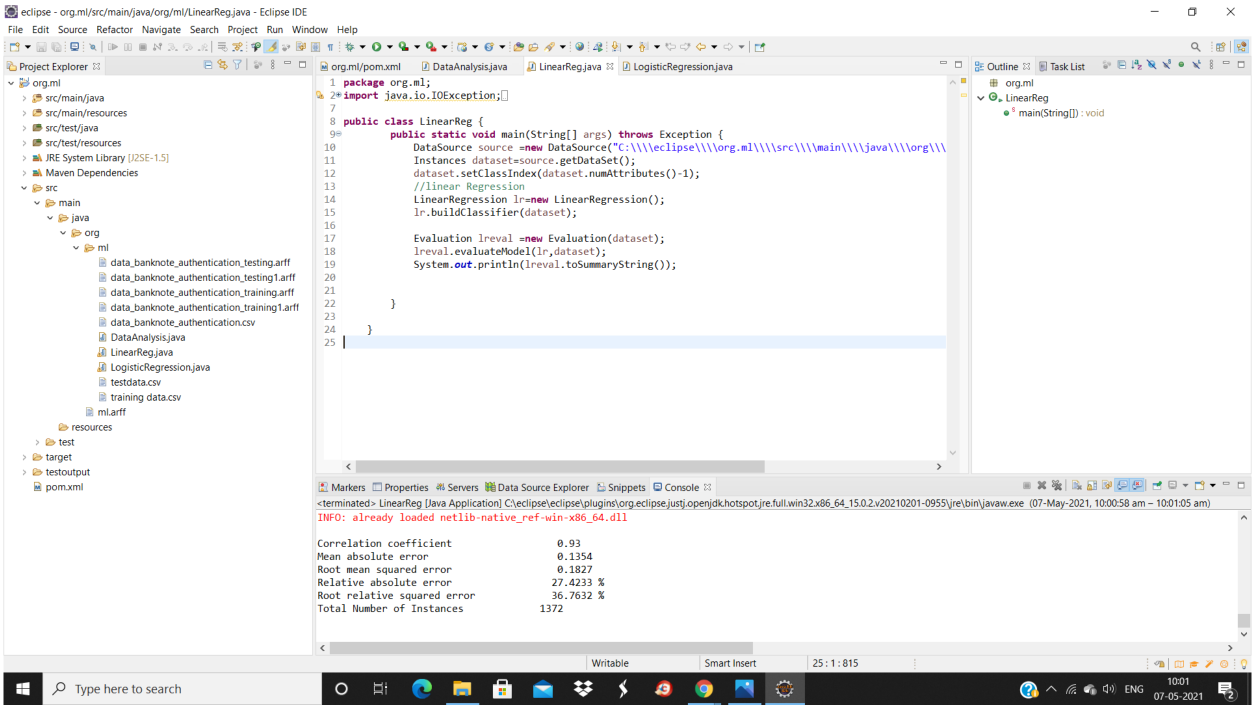
FLOWCHART:

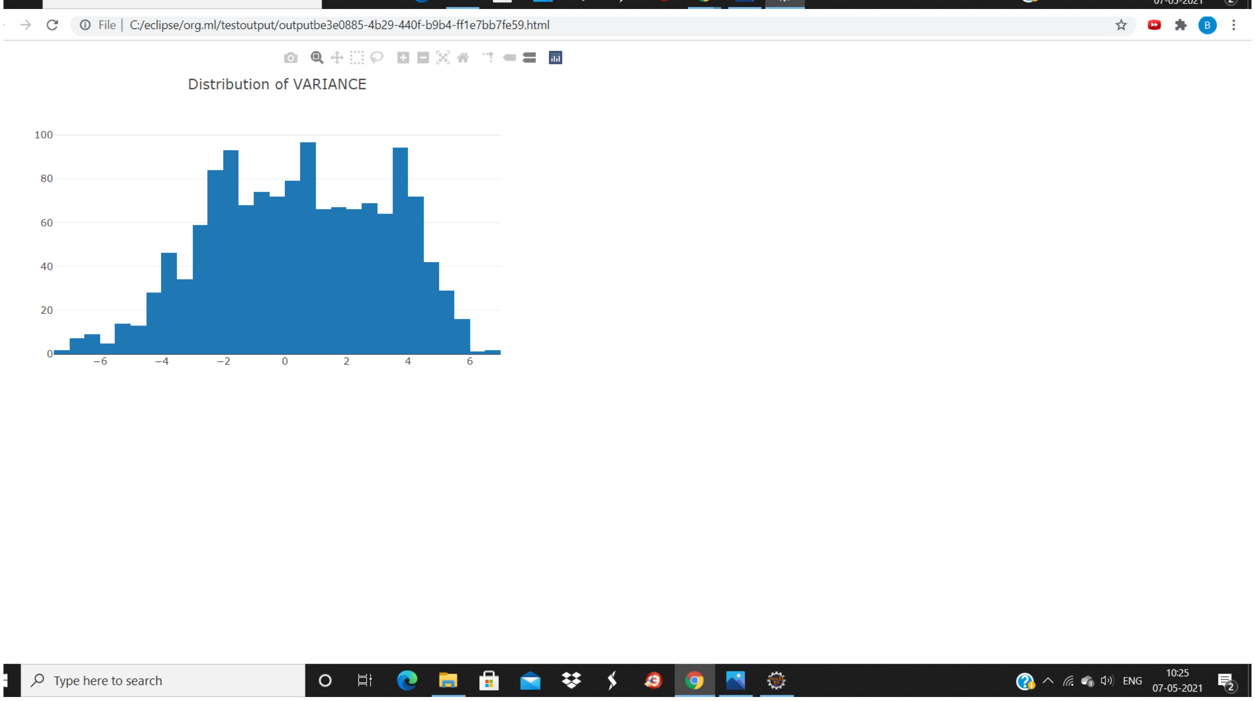

Start
Data Collection
Data preprocessing
Train test split
Algorithm Selection
Training
Identify best model
Test the model
Accuracy Satisfactory
NO
YES
Stop

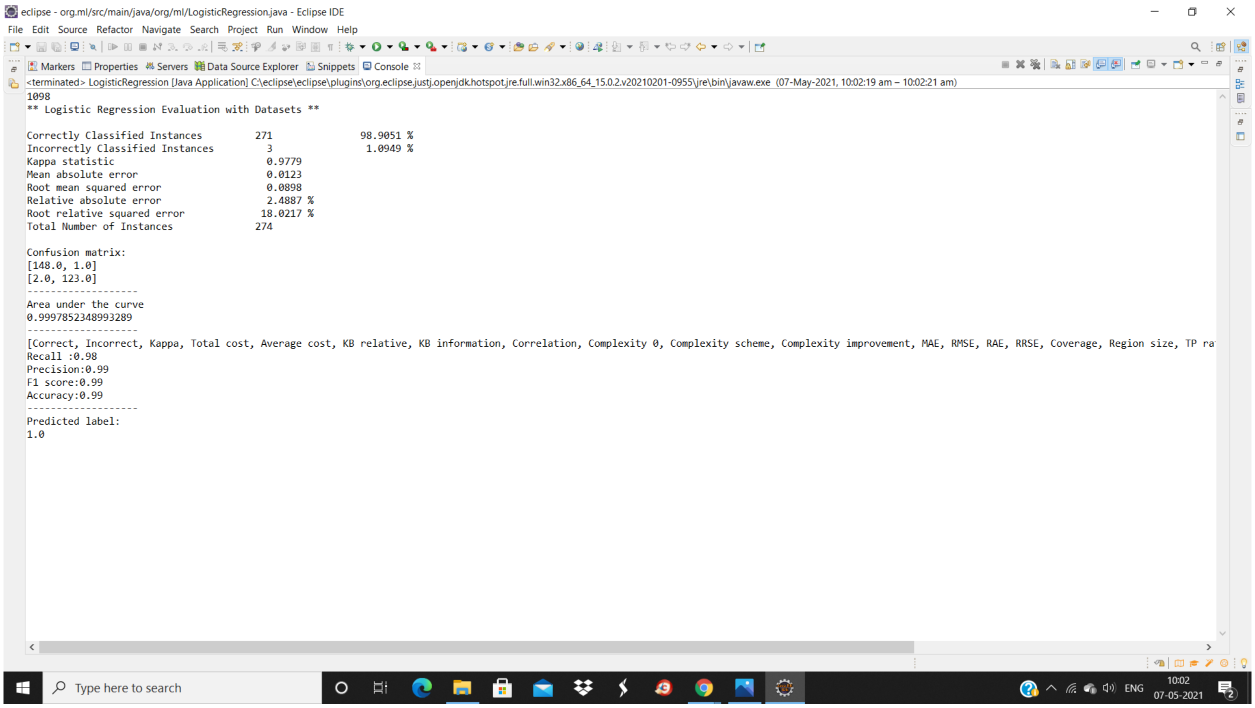
RESULT:


eclipse - org.ml/pom.xml - Eclipse IDE
File Edit Source 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_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
org.ml/pom.xml DataAnalysis.java LinearReg.java LogisticRegression.java
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
<modelVersion>4.0.0</modelVersion>
<groupId>com</groupId>
<artifactId>org.ml</artifactId>
<version>0.0.1-SNAPSHOT</version>
<dependencies>
<dependency>
<groupId>nz.ac.waikato.cms.weka</groupId>
<artifactId>weka-stable</artifactId>
<version>3.8.0</version>
</dependency>
<dependency>
<groupId>tech.tablesaw</groupId>
<artifactId>tablesaw-core</artifactId>
<version>0.38.1</version>
</dependency>
<dependency>
<groupId>tech.tablesaw</groupId>
<artifactId>tablesaw-jsplot</artifactId>
<version>0.38.1</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->
</dependencies>
<properties>
<maven.compiler.source>1.8</maven.compiler.source>
<maven.compiler.target>1.8</maven.compiler.target>
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Outline Task List
project xmlns=http://maven.apache.org/POM/4.0.0
Markers Properties Servers Data Source Explorer Snippets Console
<terminated> C:\eclipse\eclipse\plugins\org.eclipse.justj.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] --
project
Writable
Smart Insert
1 : 1 : 0
Type here to search
ENG
09:56
07-05-2021


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\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (07-May-2021, 9:58:19 am – 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
data_banknote_authentication.csv
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 |
-3.7503 | -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 | 595.0847726999998 | 2637.468481520002 | 1917.5444048900017 | -1634.9527454999989 | 610 |
Mean | 0.4337352570699696 | 1.9223531206414035 | 1.3976271172667638 | -1.1916565200437328 | 0.4446064139941689 5 |
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 | 8.081299121945154 | 34.44570967967512 | 18.576359377624264 | 4.414256203357667 | 0.2471116611696257 |
Std. Dev | 2.842762586278557 7 | 5.86904674369485 | 4.310030090106595 | 2.101013137359609 | 0.4971032701256608 |
Type here to search
ENG
09:59
07-05-2021


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_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
org.ml/pom.xml
DataAnalysis.java
LinearReg.java
LogisticRegression.java
package org.ml;
import java.io.IOException;
public class LinearReg {
public static void main(String[] args) throws Exception {
DataSource source =new DataSource("C:\\\\eclipse\\\\org.ml\\\\src\\\\main\\\\java\\\\org\\\\
Instances dataset=source.getDataSet();
dataset.setClassIndex(dataset.numAttributes()-1);
//linear Regression
LinearRegression lr=new LinearRegression();
lr.buildClassifier(dataset);
Evaluation lreval =new Evaluation(dataset);
lreval.evaluateModel(lr,dataset);
System.out.println(lreval.toSummaryString());
}
}
Outline
Task List
org.ml
LinearReg
main(String[]) : void
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
<terminated> LinearReg [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.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
Writable
Smart Insert
25 : 1 : 815
Type here to search
ENG
10:01
07-05-2021

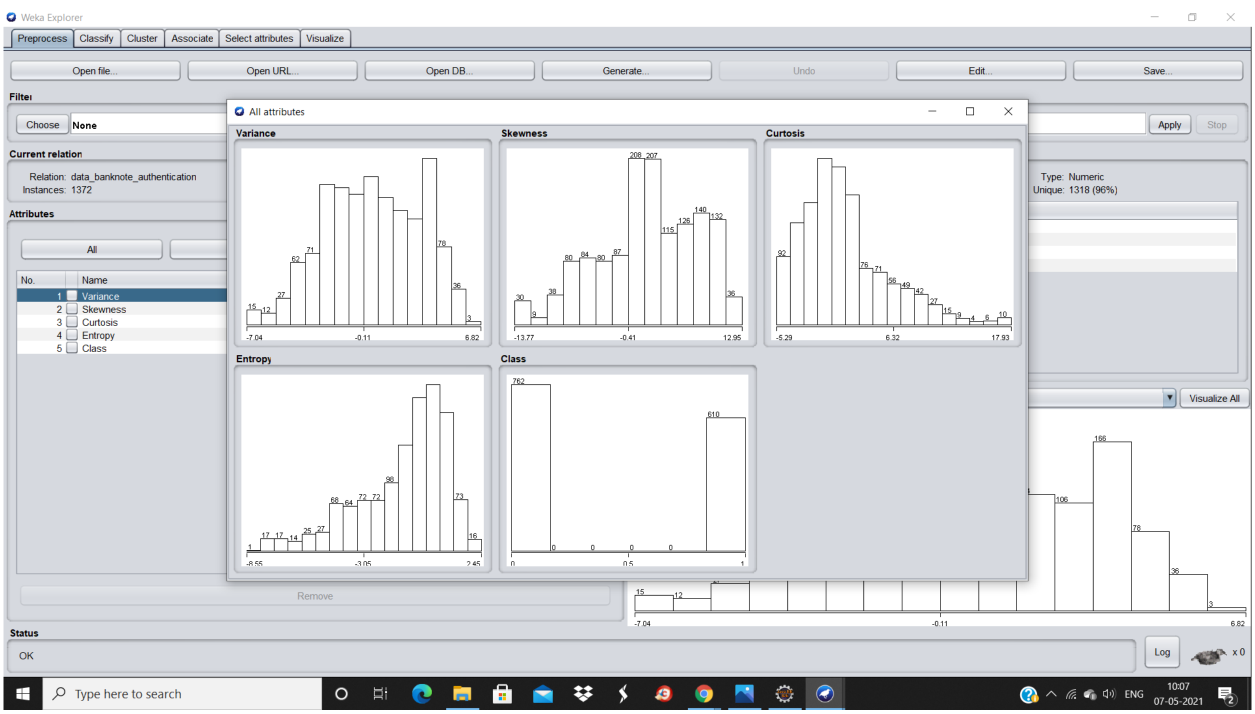

File | C:/eclipse/org.ml/testoutput/outputbe3e0885-4b29-440f-b9b4-ff1e7bb7fe59.html
Distribution of VARIANCE
100
80
60
40
20
0
−6
−4
−2
0
2
4
6
Type here to search
ENG
10:25
07-05-2021

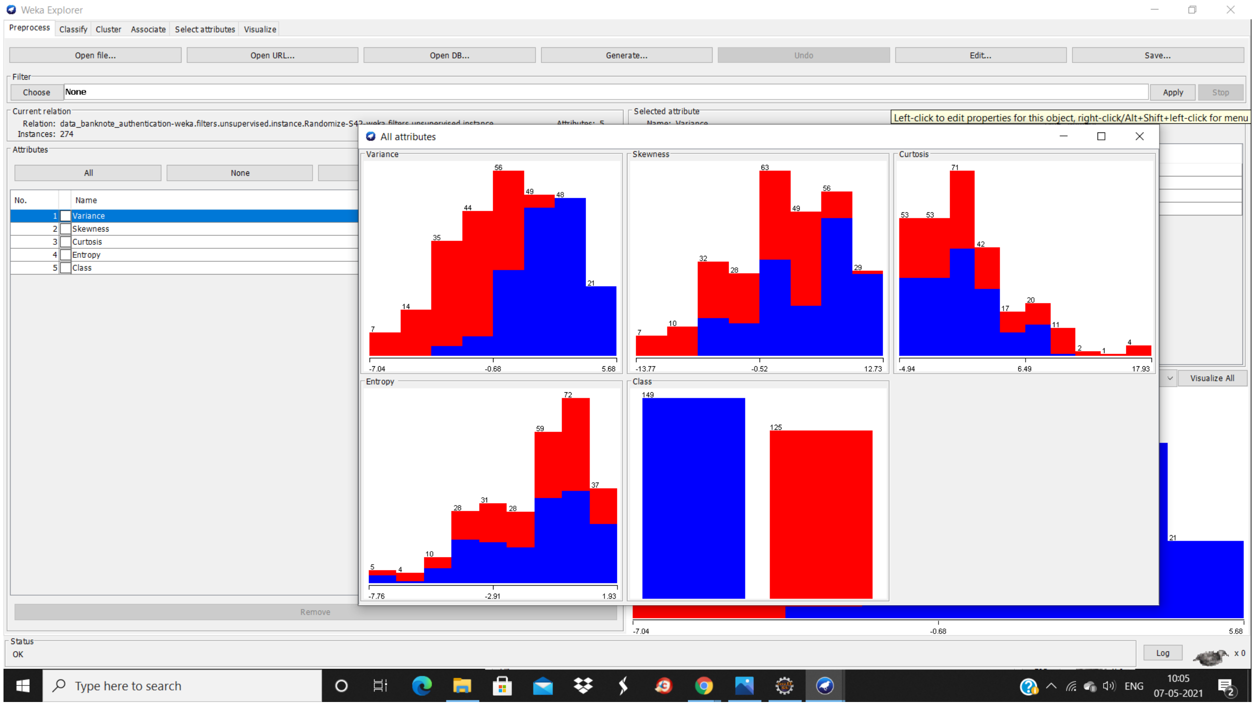

eclipse - org.ml/src/main/java/org/ml/LogisticRegression.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Markers Properties Servers Data Source Explorer Snippets Console
<terminated> LogisticRegression [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (07-May-2021, 10:02:19 am – 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]

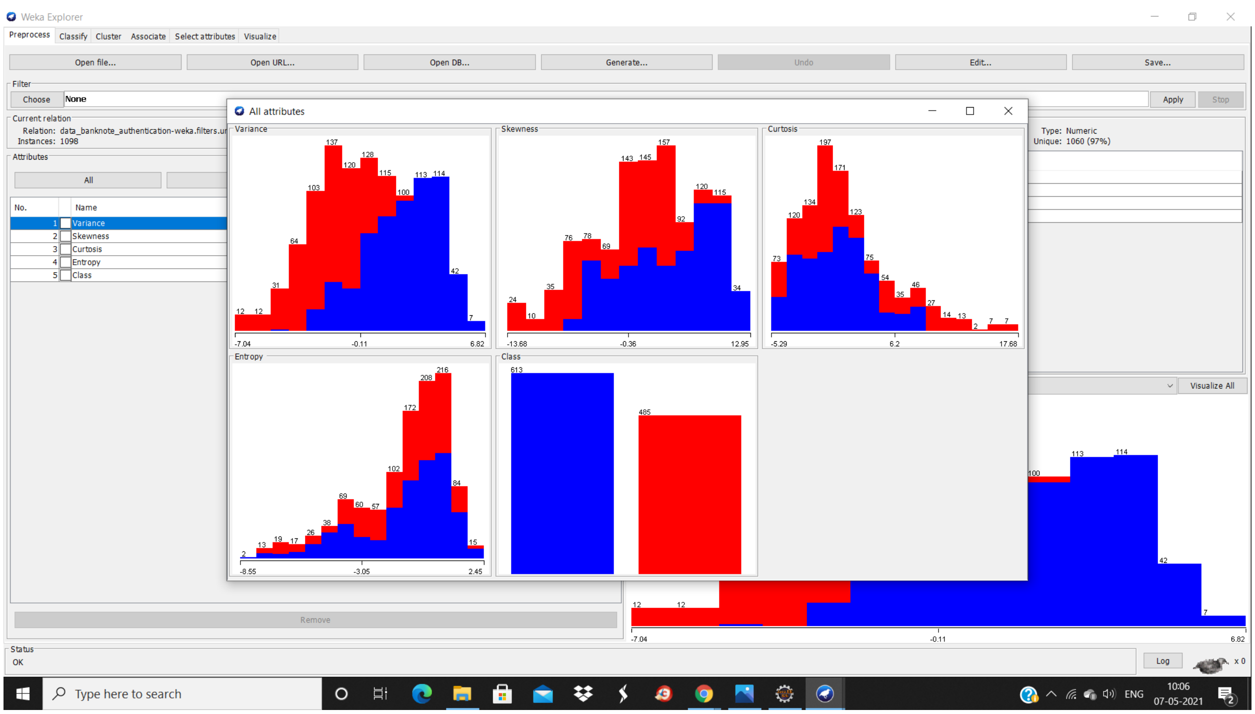
Area under the curve
0.9997852348993289

[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
10:02
07-05-2021


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose None
Apply Stop
Current relation
Relation: data_banknote_authentication
Instances: 1372
Type: Numeric
Unique: 1318 (96%)
Attributes
All
No. Name
1 Variance
2 Skewness
3 Curtosis
4 Entropy
5 Class
All attributes
Variance
Skewness
Curtosis
Entropy
Class
Visualize All
Remove
Status
OK
Log
x 0
Type here to search
10:07
07-05-2021


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: data_banknote_authentication-weka.filters.unsupervised.instance.Randomize-S42
Instances: 274
Selected attribute
Left-click to edit properties for this object, right-click/Alt+Shift+left-click for menu
Attributes
All
None
No.
Name
1 Variance
2 Skewness
3 Curtosis
4 Entropy
5 Class
Remove
All attributes
Variance
Skewness
Curtosis
Entropy
Class
Visualize All
Status
OK
Log
Type here to search
ENG
10:05
07-05-2021


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: data_banknote_authentication-weka.filters.u
Instances: 1098
Type: Numeric
Unique: 1060 (97%)
Attributes
All
No.
Name
1 Variance
2 Skewness
3 Curtosis
4 Entropy
5 Class
Remove
All attributes
Variance
Skewness
Curtosis
Entropy
Class
Visualize All
Status
OK
Log
Type here to search
ENG
10:06
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 date 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.

BIBILOGRAPHY:

- 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:

Data Analysis:

```java
package org.ml;
import java.io.IOException;
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.HistogramTrace;

public class DataAnalysis {
```

```java

    public static void main(String args[])
    {
        System.out.println("DataAnalysis");

        try {
            Table bank_data = Table.read().csv("C:\\eclipse\\org.ml\\src\\main\\java\\org\\ml\\data_banknote_authentication.csv");
            System.out.println(bank_data.shape());

            System.out.println(bank_data.first(4));
            System.out.println(bank_data.last(4));

            System.out.println(bank_data.structure());

            System.out.println(bank_data.summary());

            Layout layout1 = Layout.builder().title("Distribution of VARIANCE").build();
            HistogramTrace trace1= HistogramTrace.builder(bank_data.nCol("Variance")).build();
            Plot.show(new Figure(layout1,trace1));

            ///Layout layout3 = Layout.builder().title("").build();
            ///BoxTrace trace3= BoxTrace.builder(bank_data.categoricalColumn("Skewness"),bank_data.nCol("Variance")).build();
            ///Plot.show(new Figure(layout3,trace3));


        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
}
```

Linear Regression:

```
    package org.ml;
import java.io.IOException;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.LinearRegression;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class LinearReg {
          public static void main(String[] args) throws Exception {
               DataSource source =new DataSource("C:\\\\eclipse\\\\org.ml\\\\src\\\\main\\\\java\\\\org\\\\ml\\\\data_banknote_authentication.csv");
               Instances dataset=source.getDataSet();
               dataset.setClassIndex(dataset.numAttributes()-1);
               //linear Regression
               LinearRegression lr=new LinearRegression();
               lr.buildClassifier(dataset);

               Evaluation lreval =new Evaluation(dataset);
              lreval.evaluateModel(lr,dataset);
               System.out.println(lreval.toSummaryString());


          }

    }
```

Logistic Regression:

```
    package org.ml;
import java.util.Arrays;
```

```java

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 LogisticRegression{
         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("C:\\eclipse\\org.ml\\src\\main\\java\\org\\ml\\data_banknote_authentication_training1.arff");
              Instances test_data = getInstances("C:\\eclipse\\org.ml\\src\\main\\java\\org\\ml\\data_banknote_authentication_testing1.arff");
              System.out.println(train_data.size());

              /** Classifier here is Linear Regression */
```

```
                Classifier classifier = new weka.classifiers.functions.Logistic();
                /** */
                classifier.buildClassifier(train_data);


                /**
                 * train the algorithm with the training data and evaluate the
                 * algorithm with testing data
                 */
                Evaluation eval = new Evaluation(train_data);
                eval.evaluateModel(classifier, test_data);
                /** Print the algorithm summary */
                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("--------------------");
                Instance predicationDataSet = test_data.get(2);
                double value = classifier.classifyInstance(predicationDataSet);
                /** Prediction Output */
                System.out.println("Predicted label:");
                System.out.print(value);
            }
}
```