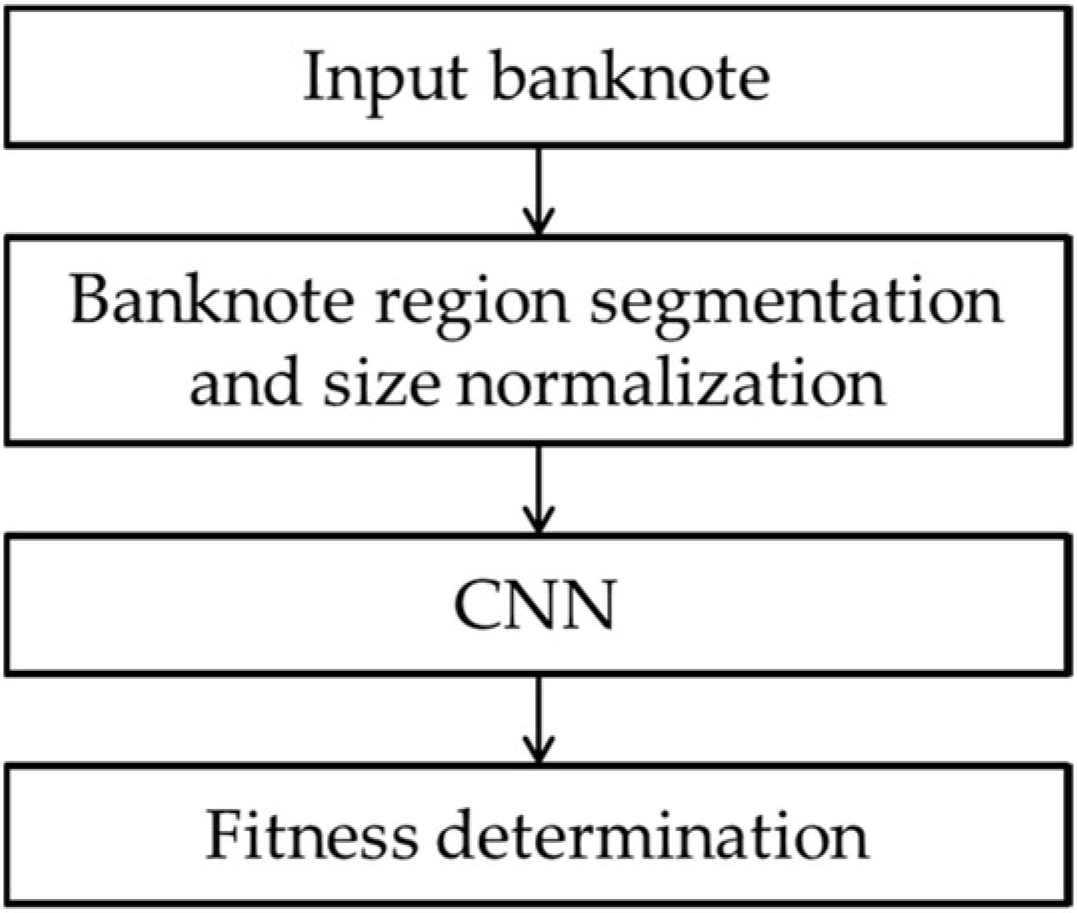
BankNote Authentication

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. 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.

Despite a decrease in the use of currency due to the recent growth in the use of electronic transactions, cash transactions remain very important in the global market. Banknotes are used to carry out financial activities. To continue with smooth cash transactions, entry of forged banknotes in circulation should be preserved. There has been a drastic increase in the rate of fake notes in the 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. The various advancements in the field of scanners and copy machines have led the miscreants to create copies of banknotes. It is difficult for human-eye to recognize a fake note because they are created with great accuracy to look alike a genuine note. Security aspects of banknotes have to be considered and security features are to be introduced to mitigate fake currency. Hence, there is a dire need in banks and ATM machines to implement a system that classifies a note as genuine or fake. In the recent years, Soft computing techniques have been widely used to solve problems that are difficult to solve using conventional mathematical methods. Supervised learning techniques are widely used in classification problems. This paper evaluates supervised machine learning algorithms to classify genuine and fake notes, and compares algorithms on the basis of accuracy, sensitivity, and specificity. Consider someone wants to deposit money in the bank. The notes that are to be deposited are given to a human being to check for their authenticity. As the fake notes are prepared with precision, it is difficult to differentiate them from genuine ones. A recognition system must be installed to detect legitimacy of the note. The system should extract the features of the note using image processing techniques. These features will be given as input to the machine learning algorithm which will predict if the note is true or fake.

FlowChart:



Description of DataSet

Attribute	**Type**	**Description**
Variance of Wavelet Transformed Image	Continuous	Variance finds how each pixel varies from the neighboring pixels and classifies them into different regions
Skewnessof Wavelet Transformed Image	Continuous	Skewness is the measure of the lack of symmetry
Curtosis of Wavelet Transformed Image	Continuous	Curtosis is a measure of whether the data are heavytailed or light-tailed relative to a normal distribution
Entropy of image	Continuous	Image entropy is a quantity which is used to describe the amount of information which must be coded for, by a compression algorithm
Class	Integer	Class contains two values 0 representing genuine note and 1 representing fake note

This problem can be solved by using different machine learning algorithms.Here,we use logistic regression approach and analyze the data to predict the output.

There have been little studies reporting technical and experimental details on how to automatically authenticate currencynotes. Many studies rely on one or two features that also can be duplicated using today’s high end technology.As there are so many security features in the currency notes, analysing only a certain aspect of the note security may not be a good choice. A complete integrated framework has been missing that looks into many aspects like security features in printing, ink, background artwork, watermark, security thread, etc. Another general shortcoming in the existing studies is the use of synthetic data. Many authors generate samples at lab to test their algorithms. Therefore, performances of these algorithms on real forensic samples are yet to be explored.

Result and Analysis:
Performance Measure:
Following measures have been used to measure the performance of the models implemented
Accuracy – The accuracy of the test is its ability to differentiate the genuine and fake note test cases correctly.
Accuracy = (TP +TN)/(TP+TN + FP + FN)
Sensitivity - The sensitivity of a test is its ability to determine the genuine note cases correctly. Sensitivity = TP/(TP + FN)
Specificity - The specificity of a test is its ability to determine the fake note cases correctly. Specificity = TN /(TN + FP)
Precision - The precision of a test is its ability to determine the number of notes that classifier labeled as genuine is actually genuine
Precision = TP/(TP + FP)
F1 Score- It is a measure of model's accuracy on a test data.
F1 Score=2*(Pecision+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

Advantages:
1.Good Accuracy
2.It is very fast at classifying unknown records

Disadvantages:
1.It assumes model to be linear
2.It requires average or no multicollinearity between independent Variables.

Bibliography:

1.Chhotu Kumar and Anil Kumar Dudyala, “Banknote Authentication using Decision Tree rules and Machine Learning Techniques”, International Conference on Advances in Computer Engineering and Applications(ICACEA), 2015.

2. Eugen Gillich and Volker Lohweg, “Banknote Authentication”, 2014.

3. Thirunavukkarasu M, Dinakaran K, Satishkumar E.N and Gnanendra S, “Comparison of support vector machine(svm) and Back propagation network (bpn) methods in predicting the protein Virulence factors”,Jr. of Industrial Polluction Control 33(2)(2017)pp 11-19.

4. Zan Huang, Hsinchun Chen, Chia-Jung-Hsu, Wun-Hwa Chen and Soushan Wuc, “Credit rating analysis with support vector machines and neural network: a market comparative study”, 2004

dataanalysis.java:

```
org1.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 {

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

		  try {
				Table bank_data =
Table.read().csv("C:\\Users\\Admin\\OneDrive\\Desktop\\pro\\data_banknote_aut
hentication.csv");

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

				System.out.println(bank_data.first(5));

				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));



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

 }
```

linearregression.java

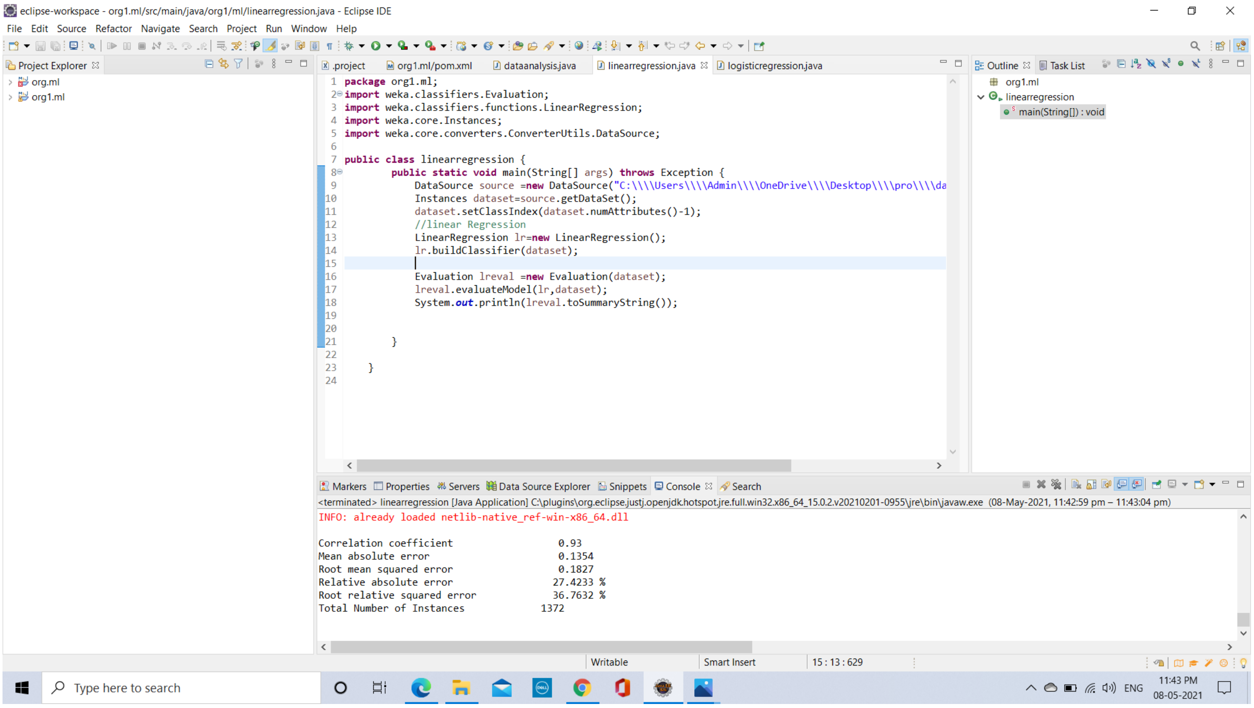
```
package org1.ml;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.LinearRegression;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class linearregression {
                public static void main(String[] args) throws Exception {
                        DataSource source =new
DataSource("C:\\\\Users\\\\Admin\\\\OneDrive\\\\Desktop\\\pro\\\\data_bankn
ote_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());


         }

        }
```

logisticregression.java:

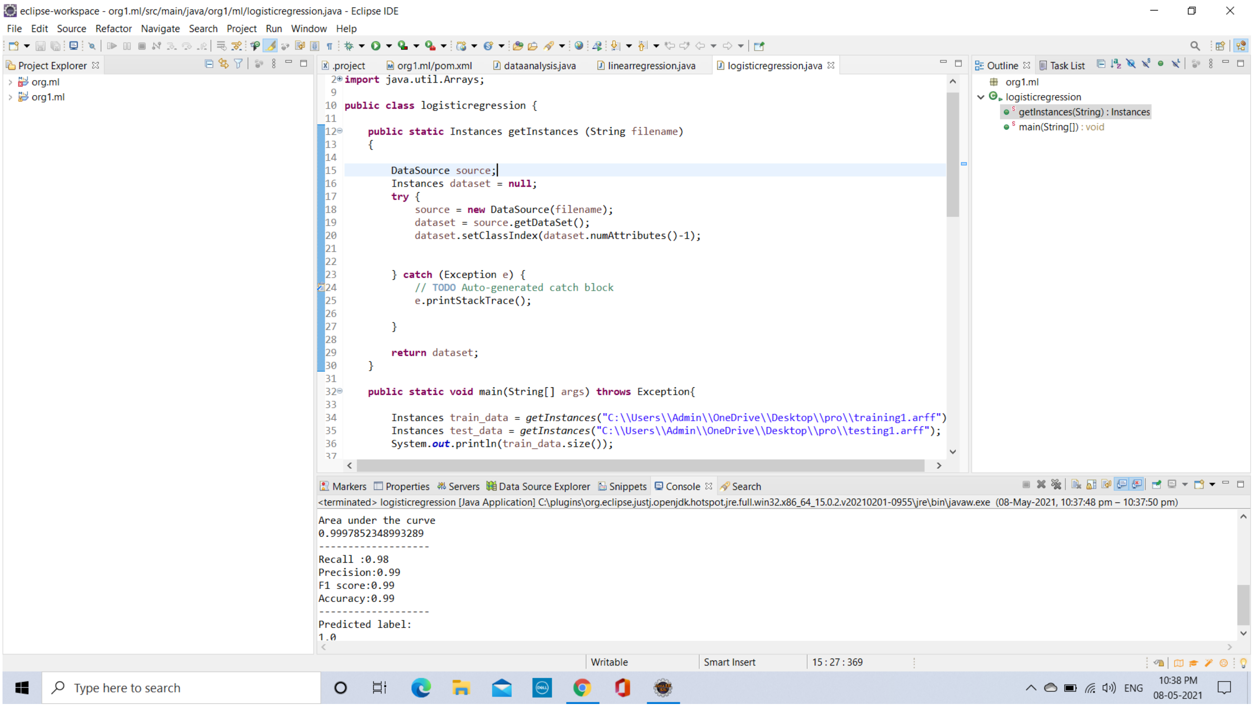
```
package org1.ml;
import java.util.Arrays;

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 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:\\Users\\Admin\\OneDrive\\Desktop\\pro\\training1.arff");
         Instances test_data =
getInstances("C:\\Users\\Admin\\OneDrive\\Desktop\\pro\\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.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);
}
```



```java
package org1.ml;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.LinearRegression;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class linearregression {
        public static void main(String[] args) throws Exception {
            DataSource source =new DataSource("C:\\\\Users\\\\Admin\\\\OneDrive\\\\Desktop\\\\pro\\\\da
            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());


        }

    }

```

```
<terminated> linearregression [Java Application] C:\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe  (08-May-2021, 11:42:59 pm – 11:43:04 pm)
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
```



eclipse-workspace - org1.ml/src/main/java/org1/ml/logisticregression.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
org.ml
org1.ml
.project
org1.ml/pom.xml
dataanalysis.java
linearregression.java
logisticregression.java
Outline
Task List
org1.ml
logisticregression
getInstances(String) : Instances
main(String[]) : void
Markers Properties Servers Data Source Explorer Snippets Console Search
Writable
Smart Insert
15 : 27 : 369
Type here to search
ENG
10:38 PM
08-05-2021

```java
import java.util.Arrays;

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:\\Users\\Admin\\OneDrive\\Desktop\\pro\\training1.arff")
        Instances test_data = getInstances("C:\\Users\\Admin\\OneDrive\\Desktop\\pro\\testing1.arff");
        System.out.println(train_data.size());
```

```
<terminated> logisticregression [Java Application] C:\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe  (08-May-2021, 10:37:48 pm – 10:37:50 pm)
Area under the curve
0.9997852348993289
-------------------
Recall :0.98
Precision:0.99
F1 score:0.99
Accuracy:0.99
-------------------
Predicted label:
1.0
```