

PROJECT REPORT
TITLE : CREDIT CARD FRAUD PREDICTION

1. INTRODUCTION

1.1 Overview

There are various fraudulent activities detection techniques has implemented in credit card transactions have been kept in researcher minds to methods to develop models based on artificial intelligence , data mining, fuzzy logic and machine learning. Credit card fraud detection is significantly difficult, but also popular problem to solve. In the proposed system I have built the credit card fraud detection using Machine learning. With the advancement of machine learning techniques, Machine learning has been identified as a successful measure for fraud detection. A large amount of data is transferred during online transaction processes, resulting in a binary result: genuine or fraudulent. Within the sample fraudulent datasets, features are constructed. These are data points namely the age and value of the customer account, as well as the origin of the credit card. There are hundreds of features and each contributes, to varying extents, towards the fraud probability. Note, the level in which each feature contributes to the fraud score is generated by the artificial intelligence of the machine which is driven by the training set, but is not determined by a fraud analyst. So, in regards to the card fraud, if the use of cards to commit fraud is proven to be high, the fraud weighting of a transaction that uses a credit card will be equally so. However, if this were to shrink, the contribution level would parallel. Simply make, these models self-learn without explicit programming such as with manual review. Credit card fraud detection using Machine learning is done by deploying the classification and regression algorithms. We use supervised learning algorithm such as Random forest algorithm to classify the fraud card transaction in online or by offline. Random forest is advanced version of Decision tree. Random forest has better efficiency and accuracy than the other machine learning algorithms. Random forest aims to reduce the previously mentioned correlation issue by picking only a subsample of the feature space at each split. Essentially, it aims to make the trees de-correlated and prune the trees by fixing a stopping criteria for node splits.

1.2 Purpose

Billions of dollars of loss are caused every year by the fraudulent credit card transactions. Fraud is old as humanity itself and can take an unlimited variety of different forms. The PwC global economic crime survey of 2017 suggests that approximately 48% of organizations experienced economic crime. Therefore, there is definitely an urge to solve the problem of credit card fraud detection. Moreover, the

development of new technologies provides additional ways in which criminals may commit fraud. The use of credit cards is prevalent in modern day society and credit card fraud has been kept on growing in recent years. Huge Financial losses have been fraudulent affects not only merchants and banks, but also individual person who are using the credits. Fraud may also affect the reputation and image of a merchant causing non-financial losses that, though difficult to quantify in the short term, may become visible in the long period. For example, if a cardholder is victim of fraud with a certain company, he may no longer trust their business and choose a contender.

2. LITERATURE SURVEY

2.1 Existing problem

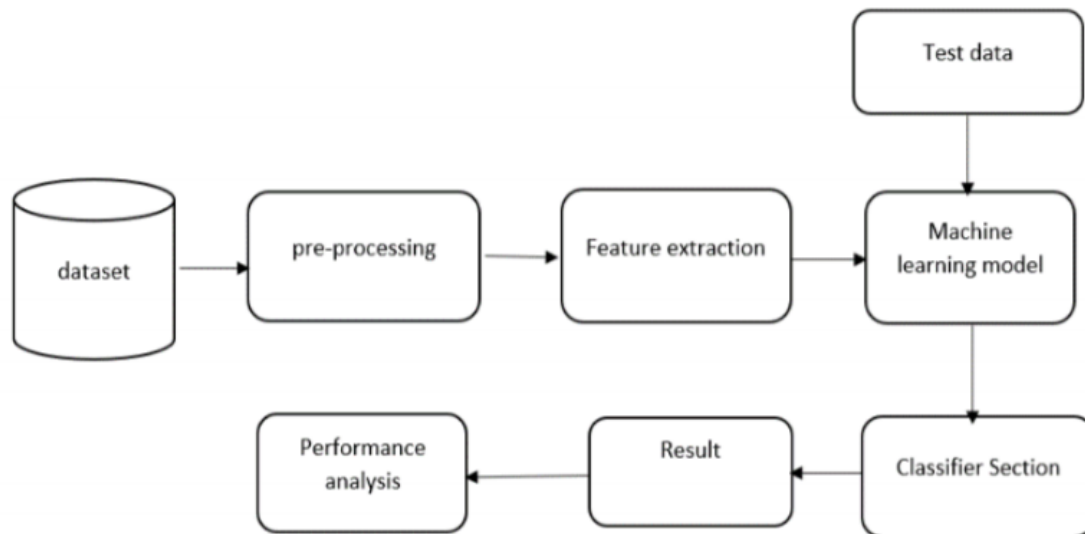
In existing System, a research about a case study involving credit card fraud detection, where data normalization is applied before Cluster Analysis and with results obtained from the use of Cluster Analysis and Artificial Neural Networks on fraud detection has shown that by clustering attributes neuronal inputs can be minimized. And promising results can be obtained by using normalized data and data should be MLP trained. This research was based on unsupervised learning. Significance of this paper was to find new methods for fraud detection and to increase the accuracy of results. The data set for this paper is based on real life transactional data by a large European company and personal details in data is kept confidential. Accuracy of an algorithm is around 50%. Significance of this paper was to find an algorithm and to reduce the cost measure. The result obtained was by 23% and the algorithm they find was Bayes minimum risk.

2.2 Proposed solution

In proposed System, I am applying random forest algorithm for classification of the credit card dataset. Random Forest is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of over fitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. The Random Forest algorithm has been found to provide a good estimate of the generalization error and to be resistant to over fitting.

3. THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software designing

HARDWARE:

Processor : Intel i5 CPU at 1.60 Ghz

RAM : 8GB

System type : 64 bit OS

SOFTWARE:

OS : Windows 7 or above

Programming language : Java

IDE : Eclipse

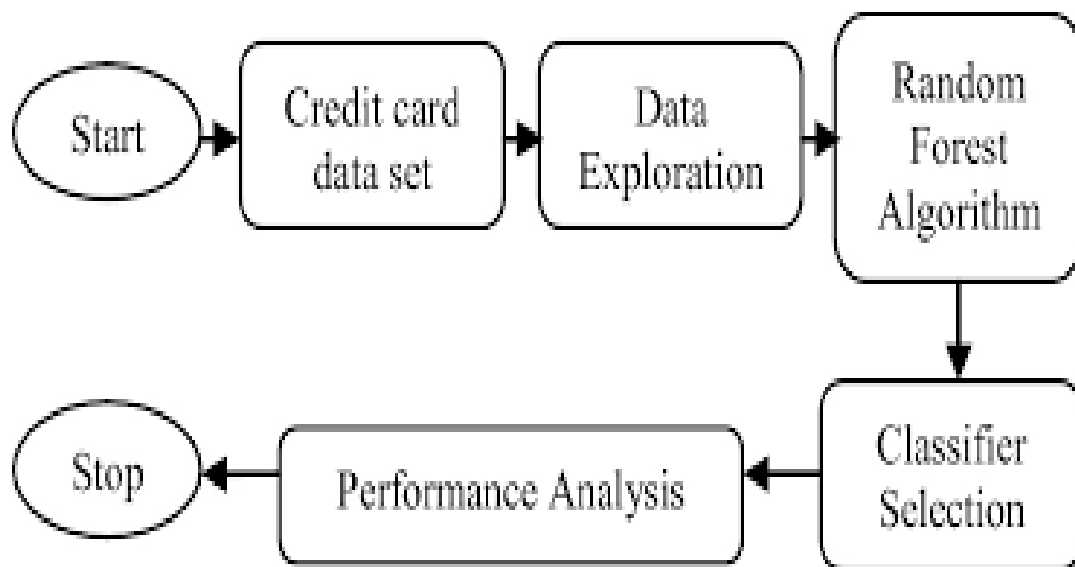
Web browser : Microsoft Internet Explorer/ Mozilla/ Google Chrome

Visualization tool : Weka

4. EXPERIMENTAL INVESTIGATIONS

I have visualized the data from the dataset using weka GUI. Machine learning classifiers were used to check the correlation coefficient and also accuracy so that best classifier could be used for the used dataset. The fraud detection models were trained and tested using Weka. Weka is an abbreviation for “Waikato Environment for Knowledge Analysis”. Weka is a workbench for machine learning that implements the majority of data mining techniques and data pre-processing and filtering techniques. Weka tool was developed in Java language by University of Waikato in New Zealand. We have also used the option 0-fold, 5-fold, 10-fold, 15-fold and 20-fold cross validation, percentage split, using training set in the process of training and testing the different models to find out which option gives us best correlation coefficient. Correlation coefficient lies between -1 to 1, 1 being good model.

5. FLOWCHART



6. RESULT

I have found out that all the features in the dataset are equally important by visualizing each feature and classifying the model using different machine learning algorithms in weka GUI.

Using test option as training set and classifier as linear regression the correlation coefficient is 0.8428

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'LinearRegression' with parameters '-S 0 -R 1.0E-8 -num-decimal-places 4'. The 'Test options' section has 'Use training set' selected. The 'Classifier output' pane displays the following results:

```

Credit_History_Available
Housing
Locality
Fraud_Risk

Test mode: evaluate on training data

=== Classifier model (full training set) ===

Linear Regression Model

Fraud_Risk =

    0.7985 * Married +
    0 * CoapplicantIncome +
    -0.2624 * Credit_History_Available +
    0.4173

Time taken to build model: 0.01 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correlation coefficient      0.8428
Mean absolute error        0.1727
Root mean squared error    0.2659
Relative absolute error     35.3768 %
Root relative squared error 53.827 %
Total Number of Instances  827

```

The 'Result list' on the left shows the following entries:

- 11:17:31 - trees RandomForest
- 11:17:42 - trees RandomForest
- 11:18:13 - functions LinearRegression
- 11:18:18 - functions LinearRegression

Using test option as cross validation with 10 folds and classifier as linear regression the correlation coefficient is 0.8378

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'LinearRegression' with parameters '-S 0 -R 1.0E-8 -num-decimal-places 4'. The 'Test options' section has 'Cross-validation' selected with 'Folds' set to 10. The 'Classifier output' pane displays the following results:

```

CoapplicantIncome
LoanAmount
Loan_Term
Credit_History_Available
Housing
Locality
Fraud_Risk

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

Linear Regression Model

Fraud_Risk =

    0.7985 * Married +
    0 * CoapplicantIncome +
    -0.2624 * Credit_History_Available +
    0.4173

Time taken to build model: 0.25 seconds

=== Cross-validation ===

=== Summary ===

Correlation coefficient      0.8378
Mean absolute error        0.1753
Root mean squared error    0.2698
Relative absolute error     35.8677 %
Root relative squared error 54.5528 %
Total Number of Instances  827

```

The 'Result list' on the left shows the following entries:

- 11:17:31 - trees RandomForest
- 11:17:42 - trees RandomForest
- 11:18:13 - functions LinearRegression
- 11:18:18 - functions LinearRegression

Using test option as training set and classifier as random forest the correlation coefficient is 0.9876

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4'. The 'Test options' section has 'Use training set' selected. The 'Classifier output' pane displays the following summary:

```
=== Summary ===
Correlation coefficient      0.9876
Mean absolute error        0.0352
Root mean squared error    0.0812
Relative absolute error     7.2199 %
Root relative squared error 16.4279 %
Total Number of Instances  827
```

The 'Result list' on the left shows several entries, with '11:17:42 - trees.RandomForest' selected.

Using test option as cross validation with 10 folds and classifier as random forest the correlation coefficient is 0.8931

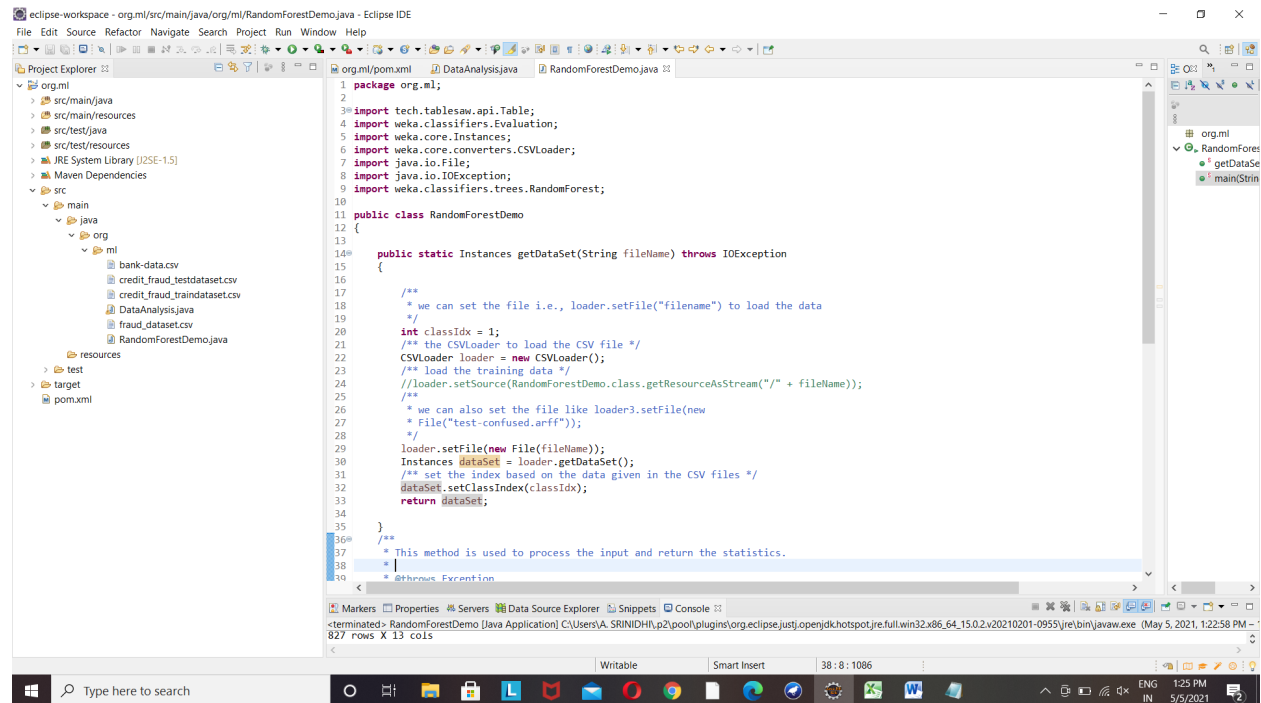
The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4'. The 'Test options' section has 'Cross-validation' selected with 'Folds' set to 10. The 'Classifier output' pane displays the following summary:

```
=== Cross-validation ===
=== Summary ===
Correlation coefficient      0.8931
Mean absolute error        0.0901
Root mean squared error    0.2220
Relative absolute error    20.0821 %
Root relative squared error 45.0433 %
Total Number of Instances  827
```

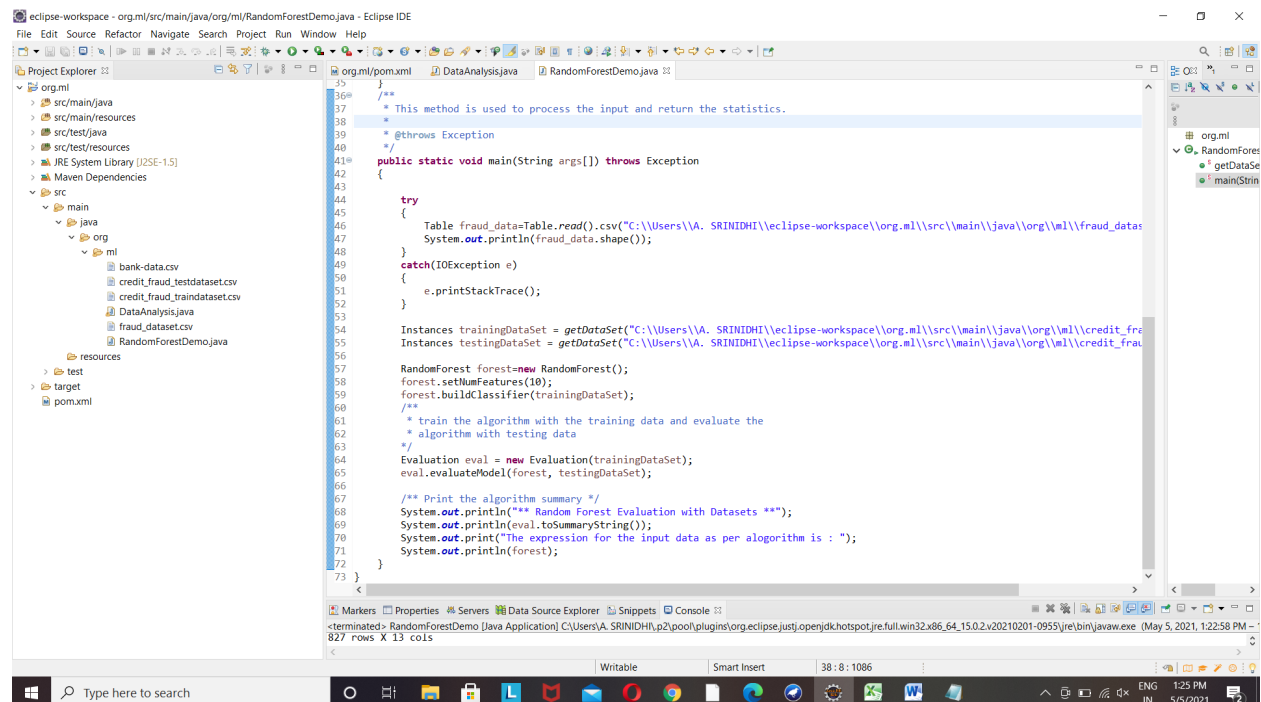
The 'Result list' on the left shows several entries, with '11:17:31 - trees.RandomForest' selected.

Since random forest algorithm gives best results i have coded it in java using weka libraries. Now, i have received correlation coefficient to be 0.9916 which means it is a good model.

Source code:



```
1 package org.ml;
2
3 import tech.tablesaw.api.Table;
4 import weka.classifiers.Evaluation;
5 import weka.core.Instances;
6 import weka.core.converters.CSVLoader;
7 import java.io.File;
8 import java.io.IOException;
9 import weka.classifiers.trees.RandomForest;
10
11 public class RandomForestDemo
12 {
13
14     public static Instances getDataSet(String filename) throws IOException
15     {
16
17         /**
18          * we can set the file i.e., loader.setFile("filename") to load the data
19          */
20
21         int classIdx = 1;
22         /** the CSVLoader to load the CSV file */
23         CSVLoader loader = new CSVLoader();
24         /** load the training data */
25         loader.setSource(RandomForestDemo.class.getResourceAsStream("/" + filename));
26
27         /**
28          * we can also set the file like loader3.setFile(new
29          * File("test-confused.arff"));
30          */
31         loader.setFile(new File(filename));
32         Instances dataSet = loader.getDataSet();
33         /** set the index based on the data given in the CSV files */
34         dataSet.setClassIndex(classIdx);
35         return dataSet;
36     }
37
38     /**
39      * This method is used to process the input and return the statistics.
40      */
41     @throws Exception
42 }
```



```
36
37 /**
38  * This method is used to process the input and return the statistics.
39  */
40 @throws Exception
41 public static void main(String args[]) throws Exception
42 {
43
44     try
45     {
46         Table fraud_data=Table.read().csv("C:\\Users\\A. SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org.ml\\fraud_data.csv");
47         System.out.println(fraud_data.shape());
48     }
49     catch(IOException e)
50     {
51         e.printStackTrace();
52     }
53
54     Instances trainingDataSet = getDataSet("C:\\Users\\A. SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org.ml\\credit_fraud_traindataset.csv");
55     Instances testingDataSet = getDataSet("C:\\Users\\A. SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org.ml\\credit_fraud_testdataset.csv");
56
57     RandomForest forest=new RandomForest();
58     forest.setNumFeatures(10);
59     forest.buildClassifier(trainingDataSet);
60
61     /**
62      * train the algorithm with the training data and evaluate the
63      * algorithm with testing data
64      */
65     Evaluation eval = new Evaluation(trainingDataSet);
66     eval.evaluateModel(forest, testingDataSet);
67
68     /** Print the algorithm summary */
69     System.out.println("** Random Forest Evaluation with Datasets **");
70     System.out.println(eval.toSummaryString());
71     System.out.print("The expression for the input data as per algorithm is : ");
72     System.out.println(forest);
73 }
```

The screenshot displays the Eclipse IDE with a Java project named 'RandomForestDemo'. The main class, 'RandomForestDemo.java', contains the following code:

```

35 }
36 /**
37  * This method is used to process the input and return the statistics.
38  *
39  * @throws Exception
40  */
41 public static void main(String args[]) throws Exception
42 {
43     try
44     {
45         Table fraud_data=Table.read().csv("C:\\Users\\A. SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org\\ml\\fraud_data
46         System.out.println(fraud_data.shape());
47     }
48     catch (IOException e)
49     {
50         e.printStackTrace();
51     }
52 }
53
54 Instances trainingDataSet = getDataSet("C:\\Users\\A. SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org\\ml\\credit_
55 Instances testingDataSet = getDataSet("C:\\Users\\A. SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org\\ml\\credit_f
56
57 RandomForest forest=new RandomForest();
58 forest.setNumFeatures(10);
59

```

The console output shows the results of the evaluation:

```

827 rows X 13 cols
** Random Forest Evaluation with Datasets **

Correlation coefficient      0.9916
Mean absolute error         0.0056
Root mean squared error     0.0386
Relative absolute error     1.334 %
Root relative squared error 9.1558 %
Total Number of Instances   248

The expression for the input data as per algorithm is : RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 10 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

```

Pros of using random forest classifier:

1. The random forest algorithm is not biased, since, there are multiple trees and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd"; therefore, the overall biasedness of the algorithm is reduced.
2. This algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees.
3. The random forest algorithm works well when you have both categorical and numerical features. The random forest algorithm also works well when data has missing values or it has not been scaled well.
4. It is robust to correlated predictors.
5. It is used to solve both regression and classification problems.

6. It can be also used to solve unsupervised ML problems.
7. It can handle thousands of input variables without variable selection.
8. It can be used as a feature selection tool using its variable importance plot.
9. It takes care of missing data internally in an effective manner.

Cons are as follows:

1. The Random Forest model is difficult to interpret.
2. It tends to return erratic predictions for observations out of range of training data. For example, the training data contains two variable x and y. The range of x variable is 30 to 70. If the test data has $x = 200$, random forest would give an unreliable prediction.
3. It can take longer than expected time to computer a large number of trees.

8. APPLICATIONS

- 1) Random forest algorithm is suitable for both classifications and regression task.
- 2) It gives a higher accuracy through cross validation.
- 3) Random forest classifier can handle the missing values and maintain the accuracy of a large proportion of data.
- 4) If there are more trees, it doesn't allow over-fitting trees in the model.
- 5) It has the ability to work upon a large data set with higher dimensionality.

Some major Applications of Random Forest in different sectors:

- **Banking Industry**

- Credit Card Fraud Detection

- Customer Segmentation
- Predicting Loan Defaults on LendingClub.com
- **Healthcare and Medicine**
 - Cardiovascular Disease Prediction
 - Diabetes Prediction
 - Breast Cancer Prediction
- **Stock Market**
 - Stock Market Prediction
 - Stock Market Sentiment Analysis
 - Bitcoin Price Detection
- **E-Commerce**
 - Product Recommendation
 - Price Optimization
 - Search Ranking

9.CONCLUSION

Hence, I have acquired the result of an accurate value(correlation coefficient) of credit card fraud detection i.e. 0.9916 (99.16%) using a random forest algorithm with new enhancements. In comparison to existing modules, this proposed module is applicable for the larger dataset and provides more accurate results. The Random forest algorithm will provide better performance with many training data, but speed during testing and application will still suffer. Usage of more pre-processing techniques would also assist.

10. FUTURE SCOPE

In accordance with future work I will try to represent this into a software application and provide a solution for credit card fraud using the new technologies like Machine Learning, Artificial Intelligence and Deep Learning.

11. BIBLIOGRAPHY

References:

[1] <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>

[2] <https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html>

[3] Gupta, Shalini, and R. Johari. "A New Framework for Credit Card Transactions Involving Mutual Authentication between Cardholder and Merchant." International Conference on Communication Systems and Network Technologies IEEE, 2011:22-26.

[4] Y. GmbH and K. G. Co, "Global online payment methods: the Full year 2016," Tech. Rep., 3 2016.

[5] Bolton, Richard J., and J. H. David. "Unsupervised Profiling Methods for Fraud Detection." Proc Credit Scoring and Credit Control VII (2001):5– 7.

APPENDIX

A. Source Code

package org.ml;

```
import tech.tablesaw.api.Table;
import weka.classifiers.Evaluation;
import weka.core.Instances;
import weka.core.converters.CSVLoader;
import java.io.File;
import java.io.IOException;
import weka.classifiers.trees.RandomForest;
```

```
public class RandomForestDemo
{
```

```

public static Instances getDataSet(String fileName) throws IOException
{

    /**
     * we can set the file i.e., loader.setFile("filename") to load the data
     */
    int classIdx = 1;
    /** the CSVLoader to load the CSV file */
    CSVLoader loader = new CSVLoader();
    /** load the training data */
    //loader.setSource(RandomForestDemo.class.getResourceAsStream("/") +
fileName));
    /**
     * we can also set the file like loader3.setFile(new
     * File("test-confused.arff"));
     */
    loader.setFile(new File(fileName));
    Instances dataSet = loader.getDataSet();
    /** set the index based on the data given in the CSV files */
    dataSet.setClassIndex(classIdx);
    return dataSet;

}
/**
 * This method is used to process the input and return the statistics.
 *
 * @throws Exception
 */
public static void main(String args[]) throws Exception
{

    try
    {
        Table fraud_data=Table.read().csv("C:\\Users\\A.
SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org\\ml\\fraud_dataset.csv");
        System.out.println(fraud_data.shape());
    }
}

```

```

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

```

```

        Instances trainingDataSet = getDataSet("C:\\Users\\A.
SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org\\ml\\credit_fraud_traindat
aset.csv");

```

```

        Instances testingDataSet = getDataSet("C:\\Users\\A.
SRINIDHI\\eclipse-workspace\\org.ml\\src\\main\\java\\org\\ml\\credit_fraud_testdat
aset.csv");

```

```

        RandomForest forest=new RandomForest();
        forest.setNumFeatures(10);
        forest.buildClassifier(trainingDataSet);
        /**
        * train the algorithm with the training data and evaluate the
        * algorithm with testing data
        */
        Evaluation eval = new Evaluation(trainingDataSet);
        eval.evaluateModel(forest, testingDataSet);

```

```

        /** Print the algorithm summary */
        System.out.println("** Random Forest Evaluation with Datasets **");
        System.out.println(eval.toSummaryString());
        System.out.print("The expression for the input data as per alogorithm is :

```

```

");

```

```

        System.out.println(forest);

```

```

    }

```

```

}

```