Credit Card Fraud Detection in Transaction By Using Decision Tree and Random Forest Algorithm

Vibhooti Jain
M.tech Scholar
Department of Information Technology
LNCT
Bhopal, India

Abhishek Dwivedi
Assistant Professor
Department of Information Technology
LNCT
Bhopal, India

***Abstract*— Due to the theatrical increase of fraud which results in loss of dollars worldwide each year, several modern techniques in detecting fraud are persistently evolved and applied to many business fields. Fraud detection involves monitoring the activities of populations of users in order to estimate, perceive or avoid undesirable behaviour. Undesirable behaviour is a broad term including delinquency, fraud, intrusion, and account defaulting. This paper we proposed an classification techniques which takes dataset of credit card transactions containing 284,807 transactions records and these techniques has capability to process the numerical and categorical datasets. In these we can take decision tree and random forest classifier and the classification result of these model is then compared the performance measure of these models .**

Keywords: data mining, random forest, decision tree, machine learning, classification, comparative analysis, credit card fraud.

I.INTRODUCTION

Credit card fraud can be defined as "Unauthorized account activity by a person for which the account was not intended. Operationally, this is an event for which action can be taken to stop the abuse in progress and incorporate risk management practices to protect against similar actions in the future".

Types of Frauds : Various types of frauds in this article include credit card frauds, telecommunication frauds, computer intrusions, Bankruptcy fraud, Theft fraud/counterfeit fraud, Application fraud, Behavioral fraud

Credit Card Fraud: Credit card fraud has been divided into two types:

· Offline fraud is committed by using a stolen physical card at call center or any other place .

· On-line fraud is committed via internet, phone, shopping, web, or in absence of card holder.

Brief of Dataset

Context It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Content The datasets contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

II. LITERATURE REVIEW

According to [1], There is a huge growth in the financial frauds , So data mining plays an important role to scan all the financial transaction. So it is a difficult task for detecting master card fraud, because the transaction dataset in sampling, choice of variables and which detection technique(s) used [4]. The datasets which they taken are from European credit card holders which contains 284,807 transaction records. First these raw data is pre-processed by various mining techniques and then splits these data into train or test datasets. After that they trained the classifier on the training datasets[5] and then test these classifiers on test datasets and computes the various performance measures such as accuracy, precision, sensitivity and specificity, And also compares these classifiers based on their performance measures.

Their is various fraud detection technologies is present through which they can identify the credit card frauds. In [2] they discussed about various problem challenges and issues they are facing in detecting frauds. Such as application frauds on which thier is identity frauds is useful for North American nation to unravel the matter of mastercard fraud. The add [2] , Application fraud when the customer apply for the credit card and there is data mismatch problems like multiple applications are submitted by one user with only one set of user details is known as duplication fraud[7]. According to [9] they used hidden markoff model for detecting the credit card frauds and its very effective also on detecting attacks [8]. In these there is a profile analyser who continuously scans or analyse the group of transactions sequence and detect if there is any fraud sequence is coming based on the cardholder's past behaviour.

III PROBLEM DEFINITION

An effective fraud detection technique have faces these kind of difficulties to achieve best performance.

- **Imbalanced data**: The Dataset of credit card transaction has imbalanced nature.
- **Different misclassification importance:** There are misclassification errors have has an different importance.
- **Overlapping data**: In dataset there are many number of transactions are considered fraudulent means they are normal (false positive) and vice verso also. So it is very difficult to maintaining a low false positive and false negative rate.

IV PROPOSED WORK

In this paper ,we proposed an network IDS based on Decision Tree and Random Forest classifier. We performed our experiments analysis on NSL KDD Data set which is mostly used for testing intrusion detection system (IDS).

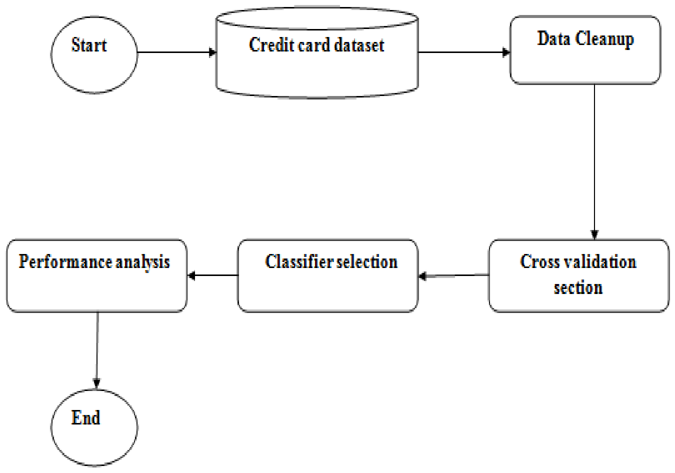


Figure 1. Proposed model

Decision Tree Algorithm

Input:
Data partition, D, which is a set of training tuples and their related class labels;
Attribute_list;
Attribute_selection_method, to determine the splitting criterion that "best" partitions.

Output: A decision tree.

```
Step 1- CREATING A ROOT NODE
1. Create a root node N
2. If tuples in D are all of the similar class, C then
3.      Return N as a leaf node label with the class C;
4. If attribute list is empty then
5.      Return N as a leaf node label with the majority class in D

Step 2- ATTRIBUTE SELECTION
6. Apply attribute_selection_method(D, attribute_list) to discover the "best " splitting _criterion attribute;
7. Label node N with splitting _criterion;
8. Update the attribute_list
```

```
Step 3- SPLIT THE TREE
9. for each outcome j of splitting_criterion
    //partition the tuples and produce subtrees for each partition
10.Based on splitting_criterion attribute
            Split the tree into two part
12.  attach a leaf labeled with the majority class D in node N:
13.  else attach the node returned by Generate_decision_tree(Dj:attribute_list)to node N:
    end for
14. return N,
```

Random Forest Algorithm

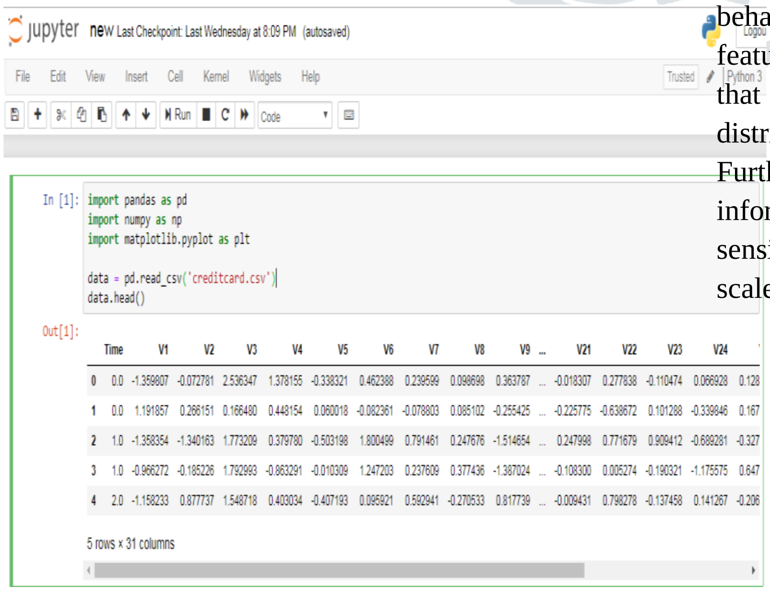
```
for S = 0, .., W do
Hi ← Bootstrap sample from H
Ti ← Contruct tree using Hi
for node = 1, .., No.Nodes do
node$_i$ ← choose random subset m of all features.
 end for
 end for
X ← take the majority vote for all trees
```

Tools & Technology used:

1. Python

V EXPERIMENTAL & RESULT ANALYSIS

For experiment we can take credit card data set contains 284807 observations of 31 variables and loaded into python , Figure 2 shows some observation details.


jupyter new Last Checkpoint: Last Wednesday at 8:09 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        data = pd.read_csv('creditcard.csv')
        data.head()

Out[1]:
   Time        V1        V2        V3        V4        V5        V6        V7        V8        V9  ...       V21       V22       V23       V24
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206

5 rows × 31 columns
```

Figure 2. Data set details

Feature Engineering

In the broadest sense correlation is any statistical association, though in common usage it most often refers to how close two variables are to having a linear relationship with each other. Familiar examples of dependent phenomena include the correlation between the physical statures of parents and their offspring, and the correlation between the demand for a limited supply product and its price. Correlation with respect to the prediction independent variable which is shown in figure 3.

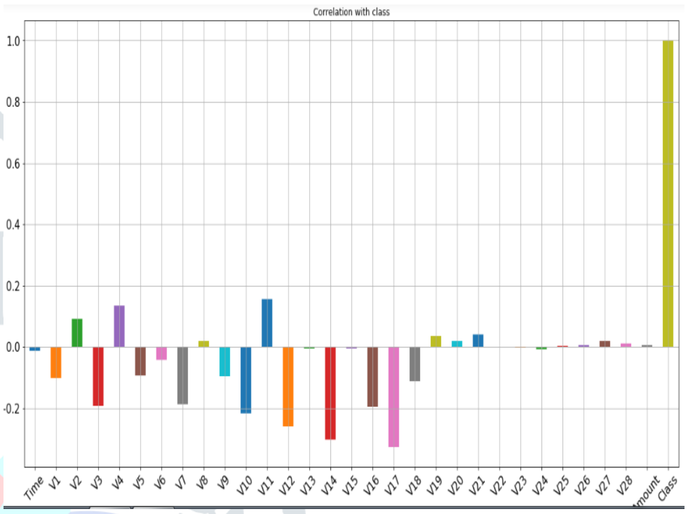
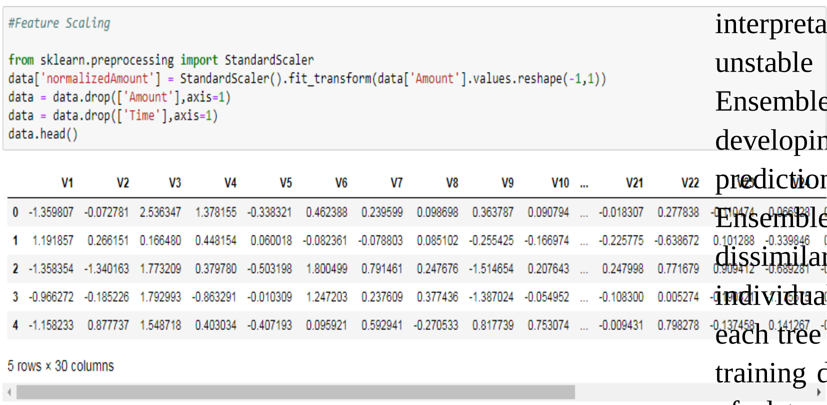


Figure 3. Correlation with Class

Feature Scaling

Bringing features onto the same scale Feature scaling is a crucial step in our pre-processing pipeline that can easily be forgotten. Decision trees and random forests are one of the very few machine learning algorithms where we don’t need to worry about feature scaling. However, the majority of machine learning and optimization algorithms behave much better Using standardization, we centre the feature columns at mean 0 with standard deviation 1 so that the feature columns take the form of a normal distribution, which makes it easier to learn the weights. Furthermore, standardization maintains useful information about outliers and makes the algorithm less sensitive to them in contrast to min-max scaling, which scales the data to a limited range of values.



```
#Feature Scaling

from sklearn.preprocessing import StandardScaler
data['normalizedAmount'] = StandardScaler().fit_transform(data['Amount'].values.reshape(-1,1))
data = data.drop(['Amount'],axis=1)
data = data.drop(['Time'],axis=1)
data.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.066928
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288	-0.339846
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	[illegible]	[illegible]
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267

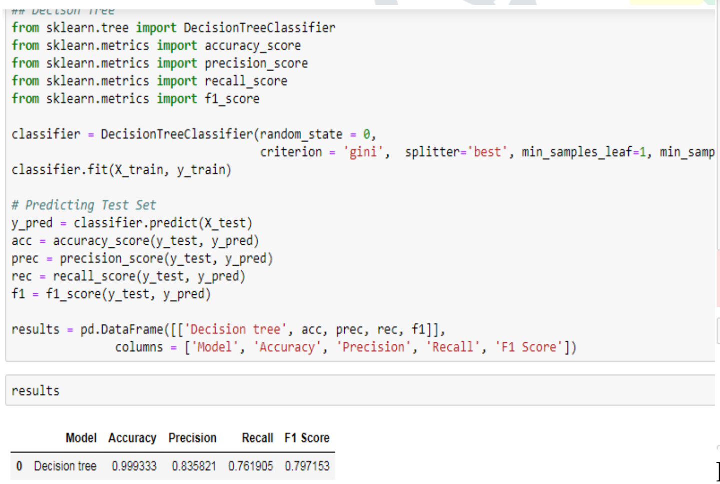
5 rows × 30 columns

Figure 4. Feature Scaling

Model Training

Decision tree

Decision trees are statistical data mining technique that express independent attributes and a dependent attributes logically AND in a tree shaped structure. Classification rules, extracted from decision trees, are IF-THEN expressions and all the tests have to succeed if each rule is to be generated. Decision tree usually separates the complex problem into many simple ones and resolves the sub problems through repeatedly using .Decision trees are predictive decision support tools that create mapping from observations to possible consequences. Figure 5 shows training and testing of the classifier model and their performance measure.



```
## Decison Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

classifier = DecisionTreeClassifier(random_state = 0,
                                    criterion = 'gini',  splitter='best', min_samples_leaf=1, min_samp
classifier.fit(X_train, y_train)

# Predicting Test Set
y_pred = classifier.predict(X_test)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

results = pd.DataFrame([['Decision tree', acc, prec, rec, f1]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])
```

```
results
```

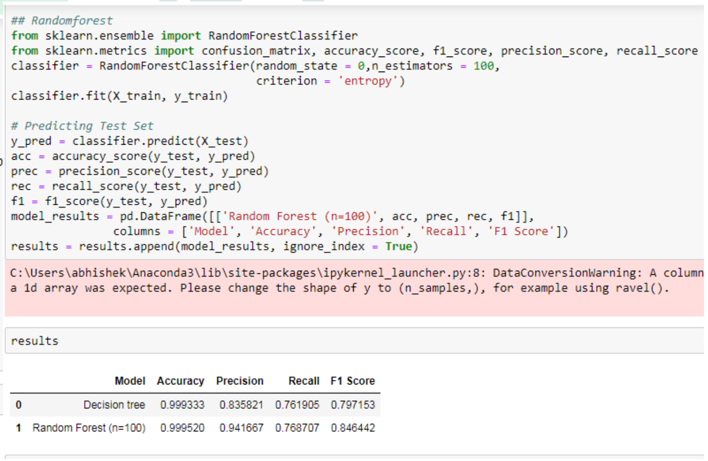
	Model	Accuracy	Precision	Recall	F1 Score
0	Decision tree	0.999333	0.835821	0.761905	0.797153

Figure 5. Performance measure of Decision Tree

Random forests

Random forest model is an ensemble of classification (or regression) trees. The popularity of decision tree models in data mining arises from their ease of use, flexibility in terms of handling various data attribute types, and interpretability. Single tree models, however, can be unstable and overly sensitive to specific training data. Ensemble methods seek to address this problem by developing a set of models and aggregating their predictions in determining the class label for a data point. Ensembles perform well when individual members are dissimilar, and random forests obtain variation among individual trees using two sources for randomness: first, each tree is built on separate bootstrapped samples of the training data; secondly, only a randomly selected subset of data is considered at each node in building the individual trees. Random forests thus combine the concepts of bagging, where individual models in an ensemble are developed through sampling with replacement from the training data, and the random subspace method, where each tree in an ensemble is built from a random subset of attributes. Given a training data set of N cases described by B attributes, each tree in the ensemble is developed as follows:

Obtain a bootstrap sample of N cases.
At each node, randomly select a subset of b attributes. Determine the best split at the node from this reduced set of b attributes
Grow the full tree without pruning
Random forests are computationally efficient since each tree is built independently of the others. With large number of trees in the ensemble, they are also noted to be robust to over fitting and noise in the data.



```
## Randomforest
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score, recall_score
classifier = RandomForestClassifier(random_state = 0,n_estimators = 100,
                                    criterion = 'entropy')
classifier.fit(X_train, y_train)

# Predicting Test Set
y_pred = classifier.predict(X_test)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
model_results = pd.DataFrame([['Random Forest (n=100)', acc, prec, rec, f1]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

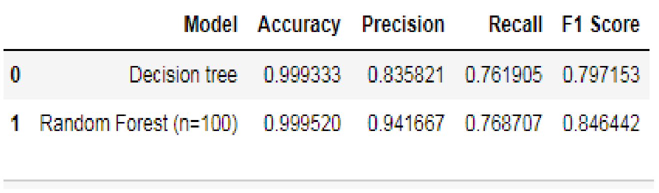
```
C:\Users\abhishek\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: DataConversionWarning: A column
a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

```
results
```

	Model	Accuracy	Precision	Recall	F1 Score
0	Decision tree	0.999333	0.835821	0.761905	0.797153
1	Random Forest (n=100)	0.999520	0.941667	0.768707	0.846442

Figure 6. Performance measure of Random Forest

Table-1. Accuracy of Models



	Model	Accuracy	Precision	Recall	F1 Score
0	Decision tree	0.999333	0.835821	0.761905	0.797153
1	Random Forest (n=100)	0.999520	0.941667	0.768707	0.846442

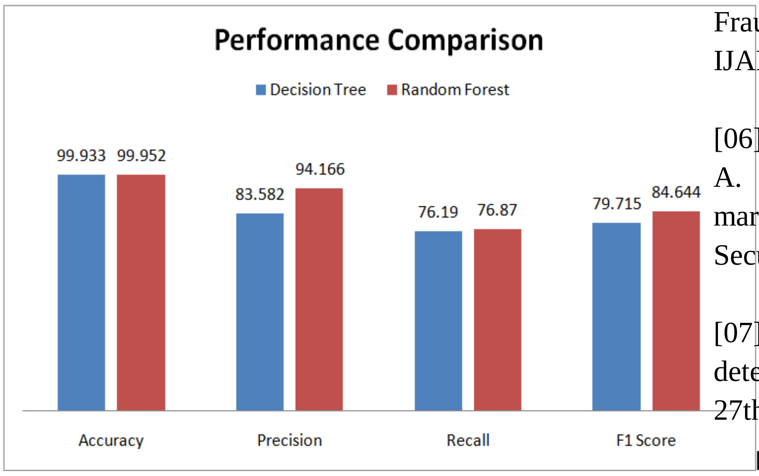


Figure-6. Performance of the Models

VI CONCLUSION

As usage of credit cards become more and more common in every field of the daily life, credit card fraud has become much more rampant. This paper investigates the Decision Tree and random forest model in binary classification of imbalanced credit card fraud data and also compare performance with each classification models respectively. The comparative results show that random forest performs better than other Decision tree.

REFERENCES

[01] John O. Awoyemi, Adebayo O. Adetunmbi, Samuel A. Oluwadare, " Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis" in 2017 IEEE .

[02] Bolton, R. J., and Hand, D. J. (2002). Statistical fraud detection: a review. Statistical Science, 17(3), 235-249.

[03] V.Dheepa and Dr. R.Dhanapal, et al. "Analysis of Credit Card Fraud Detection Methods", "IJRTE", Vol 2, No. 3, November 2009
.

[04] P Chan, W Fan, A Prodromidis & S Stolfo. 1999. Distributed data mining in credit card fraud detection, IEEE Intelligent Systems, 14(6): 67–74.

[05] N.Sivakumar, Dr.R.Balasubramanian "Credit Card Fraud Detection: Incidents, Challenges And Solutions" in IJARCSA.

[06] Srivastava, A., Kundu, A., Sural, S., and Majumdar, A. (2008). Credit card fraud detection using hidden markov model. IEEE Transactions on Dependable and Secure Computing, 5(1), 37-48.

[07] Ghosh, S. and Reilly D. L. 1994. Credit card fraud detection with a neural network. In Proceedings of the 27th Hawaii International Conference on system Science.

[08] Ju, W. H., and Vardi, Y. (2001). A hybrid high-order markov chain model for computer intrusion detection. Journal of Computational and Graphical Statistics, 10(2), 277-295.

[09] Chen, R.-C., Luo, S.-T., Liang, X. and Lee, V. C. S.. Personalized approach based on SVM and ANN for detecting credit card fraud. In Proceedings of the IEEE 2005.