Fountain Journal of Natural and Applied Sciences 2019; 8(1): 21-33

DOI: 10.53704/fujnas.v8i1.315

A publication of College of Natural and Applied Sciences, Fountain University, Osogbo, Nigeria.
Journal homepage: www.fountainjournals.com
ISSN: 2354-337X(Online),2350-1863(Print)

Comparative Analysis of Back-propagation Neural Network and K-Means Clustering Algorithm in Fraud Detection in Online Credit Card Transactions

***Abdulsalami, B. A., Kolawole, A. A., Ogunrinde, M. A., Lawal, M., Azeez, R. A. and Afolabi, A. Z.**

Department of Mathematical and Computer Sciences, Fountain University, Osogbo, Nigeria

Abstract

The ubiquitous nature of the internet had been a major driving force of the digital transformation in our world today. It has necessarily become the main medium for conducting electronic commerce (e-commerce) and online transactions. With this development, various means of possible payment methods have also emerged, such as electronic cash/ cheques, debit/credit cards, and electronic wallets. However, debit/credit cards are by far the most common payment methods employed. As a result, different credit card fraud activities have rapidly increased all over the world and are still evolving. This menace has drawn a lot of research interest and a number of techniques, with special emphasis on Data Mining, Expert System and Machine Learning (ML), as a means of identifying fraudulent behaviors. This paper examines and investigates two ML algorithms trained on public online credit card datasets, to analyze and identify fraudulent transactions. The BPNN and the K-means clustering ML algorithms were designed and implemented using Python Programming Languages. It was determined that the BPNN has a much higher accuracy of 93.1% as compared to the K-means which has an accuracy of 79.9%. Other metrics used to evaluate their performance also shows that the BPNN algorithm outperformed K-means algorithm, while the low prediction time of K-means gave it an advantage over the BPNN.

Keywords: Credit card, Fraud detection, Back-Propagation neural network, Clustering algorithm, Machine learning, Security.

Introduction

The ubiquitous nature of the internet had been a major driving force of the digital transformation in our world today. It has necessarily become the main medium for conducting electronic commerce (e-commerce). Products are advertised, browsed through and sold over the Internet. As a result, credit card transactions have become the standard for internet e-commerce. Online banking and e-commerce organizations have experienced rapid increase in credit card transaction and other modes of online transactions. Consequently, the usage of credit cards for online transactions has increased in a geometric progression. According to MasterCard Incorporation, 180 million cards were created in 2017 which led to total credit card purchases of N162 billion in the same year. With respect to the previous year, there was an increase of 11% on the overall credit card usage. However, the use of credit card is accompanied by a large number of deceptive transactions called fraud. As credit card had become

*Corresponding author: +2348034300543
Email address: basiratabdusalam@gmail.com

a popular tool for transactions in many countries lately, the growing number of credit card transactions provides more opportunity for thieves to steal credit card numbers and subsequently commit fraud. Fraud is an intentional deception with the purpose of obtaining financial gain or causing loss by implicit or explicit trick (Samaneh *et al*., 2016). Credit card fraud affects the organization by financial losses and individual user is also affected if the information of credit card gets stolen. The problem of fraud has been reported to be a serious issue in e-banking services that specially threaten credit card transactions (Samaneh *et al.*, 2016). Despite significant efforts by merchants, card issuers and law enforcement to curb fraud, online fraud continues to plague electronic commerce web sites (Aderounmu *et al*., 2012). Different credit card fraud activities have rapidly increased all over the world and are still very evolving with different techniques used by the fraudsters to perpetrate fraud. According to Akshata & Sheetal (2015), fraudulent transactions cost hundreds of millions of dollars annually.

For many years, the credit card companies have studied computing models for automating fraud detection systems. These models have been the subjects of academic research, especially with respect to e-commerce. Currently, these companies attempted to predict the legitimacy of a purchase by analyzing anomalies in various fields such as purchase location, transaction amount, and user purchase history (Chan *et al*., 1999; Banerjee *et al.*, 2018). However, with the recent increase in cases of credit card fraud, it is crucial for these companies to optimize their algorithmic solutions. Thus, improvements in fraud detection practices have become essential to maintain existence of payment system (Khan *et al.*, 2014).

This paper compares two machine learning algorithmic models, Back Propagation Neural Network (BPNN) and K-means Clustering, to explore and investigate which algorithm provides the most accurate method of classifying a credit-card transaction as fraudulent or non-fraudulent. In addition, a two-way ANOVA was used to determine if there is a significant difference between the two algorithms.

General Problems of Machine Learning in Credit Card Fraud Detection Techniques

Many machine learning techniques have been applied to the field of fraud detection ranging from supervised learning and unsupervised learning. The advantage of unsupervised methods is that previously undiscovered types of fraud may be detected. Supervised methods require accurate identification of fraudulent transactions and are only trained to discriminate between legitimate transactions and previously known fraud.

Navanshu *et al.* (2018) stated that Fraud detection in credit card is the process of identifying and classifying transactions into two classes of legit class and fraud class transactions. Several techniques are designed and implemented to solve CCFD such as Genetic algorithm (GA), Artificial neural network (ANN), frequent item set mining, Support vector machine (SVM), Hidden Markov model (HMM), Decision tree etc. CCFD is very popular but also a difficult problem to solve.

One of the major problems is the issue of having only a limited amount of data; credit card makes it challenging to match a pattern for the credit card transactions dataset. There can be many entries in the dataset with truncations of fraudsters which will fit a pattern of legitimate behavior. Also, the problem has many constraints, for example, datasets are not easily accessible for the public and the results of researches are often hidden and censored, making the results inaccessible.

Moreover, the improvement of existing methods is more difficult as a result of the associated security issues. Consequently, this imposes a limitation on exchange of ideas and methods in fraud detection, most especially in the domain of credit card.

Lastly, the credit card transaction datasets are continuously evolving and changing, which thereafter results into differences and uncertainties in the profiles of normal and fraudulent behaviors. This could result to lack of detection accuracy most especially when the dataset used to train is incorrect. Thus, it may take time and much resource for machine learning to yield the needed tangible results.

Related Works

A good amount of practices are available in the literatures for the detection of credit card fraud in an online system. Lately, this has enjoyed massive attention and interest among researchers owing to its expansive application in safeguarding online financial transactions. Recently, researchers have developed more interest in this application area, and this apparently has been employed in safeguarding online transactions and their associated components.

Pouramirarsalani *et al.* (2017), proposed a fraud detection method which used a hybrid of Feature selection and evolutionary algorithms They observed the salient features of the credit card transactions and used the same while detecting any unusual feature and flagging it to be the fraud one. The GA was used in the optimization and search problems. In the same year, Raj & Portia (2011), proposed a paper that represents a research about a case study involving CCFD, where data normalization was applied prior to Cluster analysis. The results obtained shows that neuronal inputs can be minimized using clustering attributes. They concluded that promising results could be obtained by using normalized data, while ensuring that the data should be Multilayer perceptron trained. This work was based on unsupervised learning. The significance of their paper was finding new methods for fraud detection and increasing the accuracy of detection results.

Falaki *et al*. (2012), developed a probabilistic Credit card fraud detection system (CCFDS) in online transactions. The developed probabilistic based model served as a basis for mathematical derivation of adaptive threshold algorithm for detecting anomalies in transactions. Their experimental result shows the performance and effectiveness of new systematic approach, and demonstrates the usefulness of learning the spending profile of the cardholders. The optimization of parameters, posterior-viterbi cum new detection model performed better than viterbi cum old detection model. The results obtained from the evaluation showed the overall average of accuracy and precision are about 84% and about 86% respectively.

Also, Dhanapal & Gayathiri (2012) implemented CCFDS using Decision tree and Hunts algorithm. His work detects fraudulent customers/merchants by tracing fake mails and IP address. Customers/merchants are tagged suspicious if the mail is fake, and thereafter trace the owner/sender through IP Address. Their work was termed "Tracing Email and IP fraud detection".

Avinash & Thool (2013) used the Hidden Markov Model (HMM) and combined it with the Clustering algorithm in CCFD. They were able to build a system that can detect fraud using customer spending profile. The system checks for the past transaction history of the customer and make decision from it. Its limitation was True Positive (TP) and False positive (FP) issues.

Patel & Gond (2014) proposed SVM learning for CCFD. The SVM based method with multiple kernel involvement, which also includes several fields of user profile instead of only spending profile. The simulation result shows improvement in TP (true positive), TN (true negative) rate, and also decreases the FP (false positive) and FN (false positive) rate. In their work, they looked into the effectiveness of hybridizing two techniques, using both the users profile and spending profile in the detection of anomalies in online transaction, with the goal of improving on false rejections and prediction accuracy, and compared it with the standalone units of the algorithms.

Also, Devaki *et al.* (2014), proposed a CCFD using time series analysis. The parameters considered are transaction amount and transaction time. They used the periodic pattern in the spending behaviour of a cardholder to detect the anomalies in the transaction with respect to the analyses of the past history of transactions belonging to an individual cardholder. Two levels of detection methods were used. The first level detected fraud by analyzing whether the new incoming transaction is fraud or not, using distance-based method. In the second level, the next transaction was predicted by means of label-prediction methodology and compared with the actual transaction, if there is deviation; it is detected to be a fraudulent transaction. If the particular transaction is considered as a fraud, the cardholder is asked to

continue the transaction by asking a secret question. However, if the cardholder does not give correct answer, the transaction is halted and discontinued. The approach decreases the FP situation. Hence, non-rejection of genuine transaction was ensured.

In another work by Pooja *et al.* (2015), they proposed simple K-means and Simple GA for fraud detection. In their work, they discussed how K-means algorithm grouped the transactions based on the distinct attribute values. GA was used for implementing optimization because the increase in size of the input K-means algorithm produced outliers. Basically K-means algorithm produced clusters which were then optimized by the GA.

Behera & Panigrahi (2015), proposed a hybrid approach to CCFD using Fuzzy clustering and Neural Network (NN). Their work was implemented in two phases. In phase one, they used K-means clustering algorithm to generate a suspicious score of the transaction, while in the second phase, a suspicious transaction was fed into NN to confirm whether it was fraudulent or otherwise.

In another work by Esmaily *et al*. (2015), they proposed a hybrid of ANN and Decision tree. In their model, they used a two-phase approach. In the first phase, the classification results of Decision tree and Multilayer perceptron were used to generate a new dataset, while in the second phase, data was fed into Multilayer perceptron to finally classify the data. This model promises reliability by giving very low false detection rate.

Agrawal *et al.* (2015), proposed testing credit card transaction using HMM, Behavior based technique and GA. They used the HMM to maintain the record of previous transactions, behavior based technique for grouping the datasets and GA for optimization, i.e. calculating the threshold value.

In another work proposed by Demla & Agrawal (2016), it was gathered that various modern techniques based on Sequence Alignment, ML, AI, Genetic Programming (GP), Data mining etc., has evolved and is still evolving, in detecting fraudulent transactions in credit cards. However, a sound and clear understanding of all these approaches is needed, which will certainly lead to an efficient CCFDS. Survey of various techniques used in CCFD mechanisms was shown in this work along with evaluation of each methodology based on certain design criteria. The survey was purely to detect the efficiency and transparency of each method. The aim of their work was to conduct a survey, compare different CCFD algorithms, and find the most suitable algorithm to solve the problem.

In another work by Fashoto *et al*. (2016), they proposed a hybrid of K-means clustering with Multilayer Perceptron (MLP) and HMM. They used K-means clustering to group together the suspected fraudulent transactions into similar clusters. The output of this was used to train the HMM and the MLP, which thereafter classifies the incoming transactions. They found in their results that the detection accuracy of "MLP with K-means Clustering" is higher than the "HMM with K-means clustering" but the result was reversed for 10-fold cross-validation.

Navanshu *et al.* (2018), proposed a new collative comparison measure that reasonably represents the gains and losses due to fraud detection. The significance of their paper was to find an algorithm and to reduce the cost measure. A cost sensitive method which was based on Bayes minimum risk was presented using the proposed cost measure. An improvement up to 23% was obtained when this method was compared with other state of art algorithms. The data set for this paper was based on real life transactional data by a large European company and the personal details in the data was kept confidential; with the accuracy of the algorithm to be around 50%. The result obtained was by 23% and the algorithm they find was Bayes minimum risk.

Research Methodology

Discussed in this section is the methodology adopted in implementing the proposed CCFDS in an online credit card transaction dataset. The algorithmic approach, their key concepts, test framework, and datasets for implementation (real credit card transactions are major points /focus here.

As shown in Figure 1 below, the fraud detection process comprises of Four (4) phases; the data Acquisition, data pre-processing, feature extraction, and the classification phase (which

involves Training and testing). The credit card transactions data used as inputs were acquired from Kaggle.com, Data preprocessing phase helps in data normalization, it includes getting rid of unwanted data that would have been extracted as features. The feature extraction phase extracts the features that will be used inputs to train the algorithm. The classification phase classifies the credit card data transactions into legal or illegal.

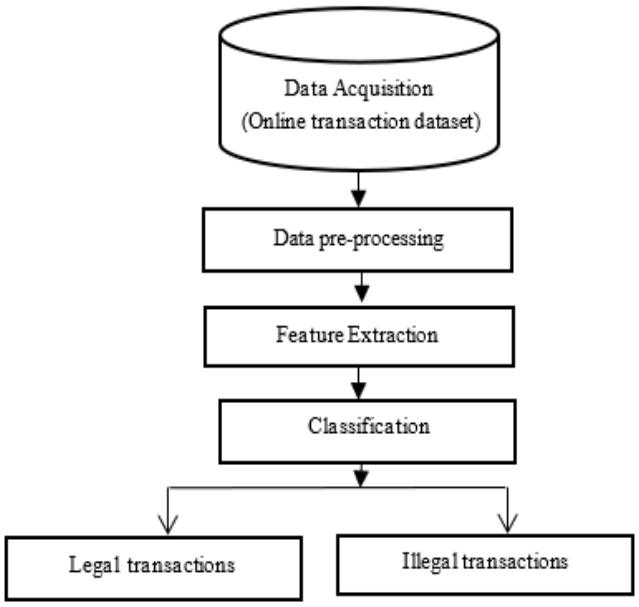

Figure 1: Credit Card Fraud Detection Model Architecture

Data Acquisition

The acquisition of dataset to test the model was a difficult task mainly because financial institutions do not generally agree to share their data with researchers for security reasons. All efforts prove abortive; a real credit card transaction of Europeans cardholders that was provided for CCFD on Kaggle.com was acquired as alternative. A total number of Two hundred and eighty-four thousand, eighty hundred and seven (284,807) real credit card transactions was used to train and validate the two models to detect if the transaction was fraudulent or valid

Data pre-processing

The obtained dataset was already pre-processed in such a way that the variables were acceptable to be used in training the models. The dataset provided was already labeled i.e. classified into Fraudulent and Valid transaction. The total number of valid transactions was Two hundred and eighty-four thousand, three hundred and fifteen (284,315) and the total number of fraudulent transaction was Four hundred and ninety-two (492). Eighty percent (80%) of the transaction dataset, and twenty percentages (20%) was used in training and testing the models respectively.

Feature Extraction

In this work, the credit card features (information) that was used for training and classification were extracted using Principal Component Analysis (PCA), except for the "Time" and "Amount" column. The main idea of the PCA is to reduce the dimensionality of a dataset consisting of many variables correlated with each other. According to Asiedu *et al.* (2016), it is an effective way to suppress redundant information and provide only one or two composite data from most of the information from the initial data. Hence, the transaction variables were reduced into the PCA variables, which are the sensitive information of the dataset or cardholder such as the name, address, sex, location, marital status, etc. This sensitive information was transformed using the PCA for confidentiality issues.

Classification

This is the last stage of the credit card detection process. This stage comprises of the training stage and testing stage. The training stage comprises of 80% (Two hundred and twenty-seven thousand, eight hundred and forty-six) of the dataset and the other 20% (Fifty-six thousand, nine hundred and sixty-one) was used in testing. The features extracted (PCA variables) were fed into the BPNN and K-means classifier and the transactions were trained and classified to the corresponding transactions (legal or illegal).

The Proposed Algorithm

Back Propagation Neural Network (BPNN)

This is the most popular learning algorithm to train the NN. It is a multi-stage dynamic system optimization method that minimizes the objective function. It is a supervised learning method and is a generalization of the delta rule. It is most useful

for the feed-forward network which is network that has no feedback. It consists of three layers, input; hidden; and output layers. The data are passed from input layer through hidden layer and then to the output layer. This is also known as forward propagation. The input data is repeatedly feed to the neural network. With each presentation the output of the NN is compared to the desired output and an error is computed. This error is then feed-back (back propagated) to the NN and used to adjust the weights such that the error decreases with each iteration and the NN gets closer to producing the desired output.

How BPNN works

The BPNN model consists of an Input layer, five (5) Hidden layers and an Output layer. In this model, each of the neurons in each layer was connected with each other without making any loop (Multilayer feed forward neural network), i.e., the link was in a forward direction to the output layer. The connection between each neuron was used to store the weights. These weights are initialized with some random values and change at every iteration in the training process. The BPNN required the dataset to be labeled so that the desired outputs can be compared to achieve the system outputs, and the system was tuned by adjusting connection weights to narrow the difference between the two outputs as much as possible. The weights are updated backwards, from the output layer back to the input layer, until the actual result was same with the expected value.

Figure 2 shows the flowchart for the steps, taking by the programme in implementing the CCFDS using BPNN algorithm.

K-means Clustering Algorithm: The clustering algorithm finds the centroid of a group of the datasets. To determine cluster membership, the algorithms evaluate the distance between a point and the cluster centroids. The output from the clustering algorithm is basically a statistical description of the cluster centroids with the number of components in each cluster. The K-means algorithm used in this work is one of the most non-hierarchical methods used for data clustering. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k-clusters).

How K-means Clustering works

The main idea was to partition the raw dataset into *k* clusters based on the initial value of k which in this case, k is initialized to be two (2) i.e. Fraudulent and valid transaction. Each new instance was compared with existing ones by using a distance metric, and the closest existing instance was used to assign the class to the new on.

Basic K-Mean Clustering Technique

$$\sum_{i,j=1}^{n} [\min d\text{^}2(xi, mj)] \quad (1.1)$$

where d(xi, mj) denotes the Euclidean distance between xi and mj. The points {mj} (j=1, 2…, k) are known as cluster centroids.

Figure 3 shows the flowchart of the steps taking by the programme, in implementing the CCFDS using K-Means clustering algorithm.

Pseudocode 1: BPNN Pseudo Code

// The following describes the how the Back-propagation algorithm works.
Assign all network inputs and output
Initialize all weights with small random numbers
repeat
for every pattern in the training set
Present the pattern to the network
// Propagated the input forward through the network:
for each layer in the network
for every node in the layer
1. *Calculate the weight sum of the inputs to the node*
2. *Add the threshold to the sum*
3. *Calculate the activation for the node*

End
End
// Propagate the errors backward through the network
for every node in the output layer
calculate the error signal
end

for all hidden layers
for every node in the layer
1. *Calculate the node's signal error*
2. *Update each node's weight in the network*

End
End
// Calculate the Error Function
End
while ((maximum number of iterations < = specified)

Pseudocode 2: K-means Pseudocode

Input: k (The number of clusters),
D (set of elements)
Output: a set of k-clusters
Method:
Arbitrarily choose k-object from D as the initial cluster centers;
repeat
Assign each item to the cluster which has the closest mean;
Calculate new mean for each cluster;
***Until** all criteria is met*

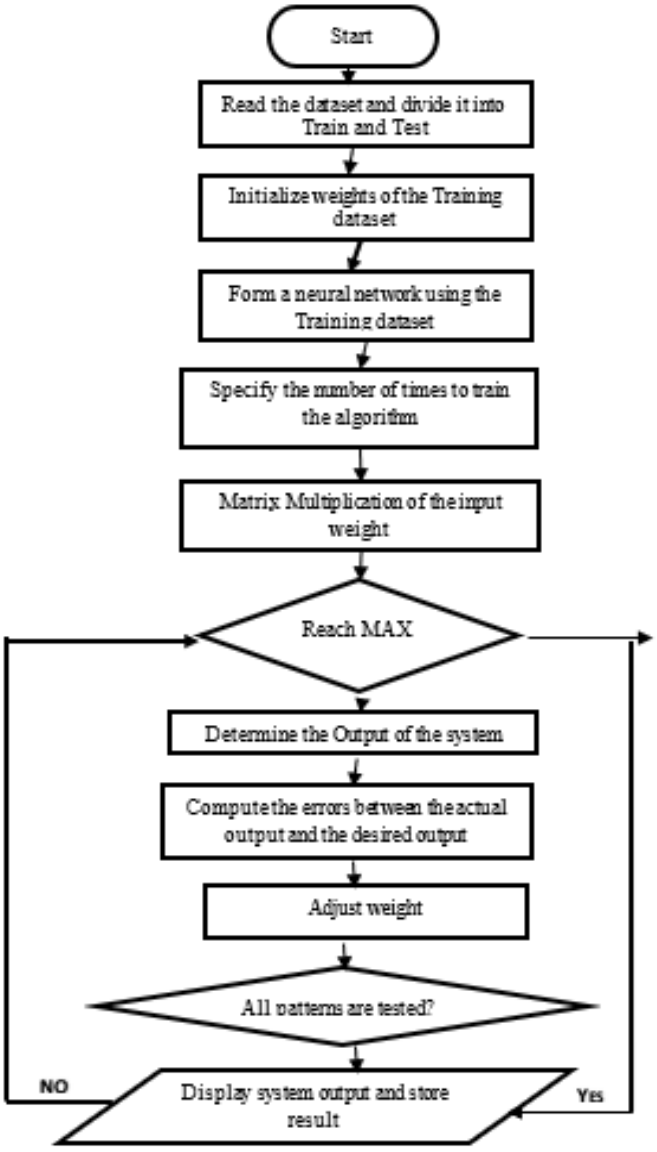

Figure 2: The BPNN Flowchart for Credit Card Fraud Detection System

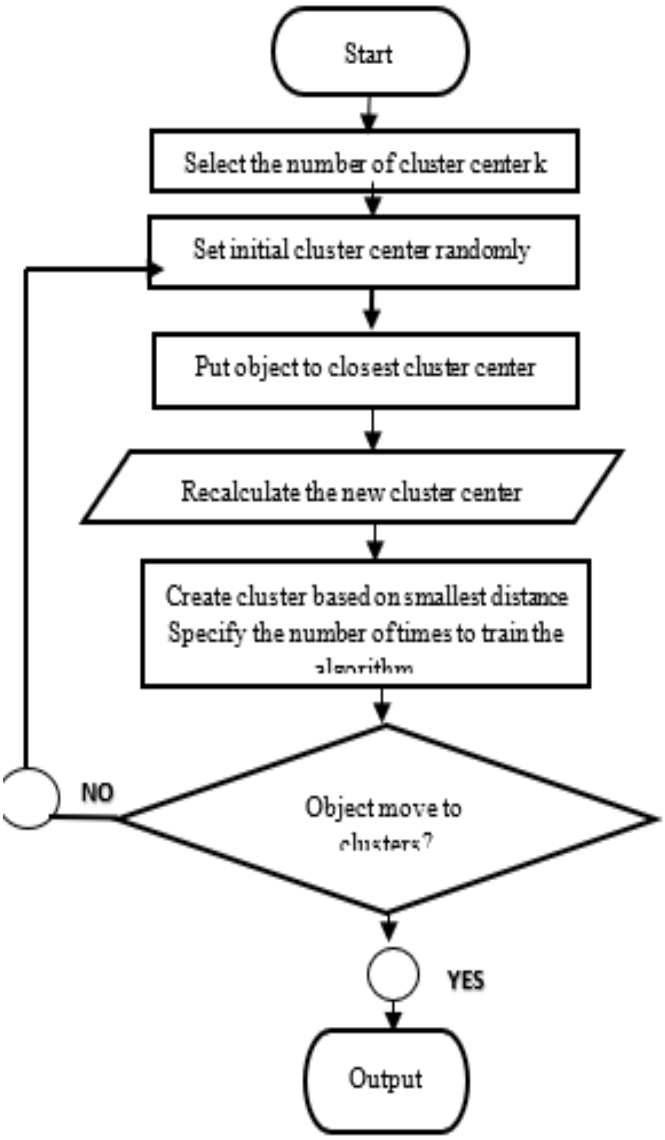

Figure 3: K-Means Flowchart for Credit Card Fraud Detection System

Implementation, Findings and Results

The developed systems were implemented in two stages namely, the training stage and the testing stage. A total of two hundred and eighty-four thousand, eighty hundred and seven (284,807) real credit card transactions each was used to train and also test the two models to detect if a transaction was fraudulent or valid. Figure 4a and 4b shows the training and testing process of BPNN respectively. At the training phase (as shown in Figure 4a), the experiments were performed using Two hundred and twenty-seven thousand, eight hundred and forty-six (80%) of the dataset, while fifty-six thousand, nine hundred and sixty-one (20%) was used in testing the model (as shown in Figure 4b). The performance evaluation metrics considered are System accuracy, Precision and Recall (Sensitivity), Error rate, False positive rate (Specificity), Prediction accuracy, Hit rate and Miss rate, which were calculated with indices True positive (TP), True negative (TN), False negative (FN) and False positive (FP) using equations 1.2, 1.3, 1.4, 1.5, and 1.6 and respectively.

$$Precision = \frac{TP}{(TP+FP)} \quad (1.2)$$

$$Recall = \frac{TP}{(TP+FN)} \quad (1.3)$$

$$Error\ Rate = \frac{FN+FP}{TN+FP+TP+FN} \quad (1.4)$$

$$False\ Positive\ Rate\ (FPR) = \frac{TN}{FN+TN} \quad (1.5)$$

$$Predictive\ Accuracy = \frac{TP+TN}{TN+FP+TP+FN} \quad (1.6)$$

Figure 5 shows the snapshot of the dataset used.

The BPNN Model Implementation

The network consists one input layer, five (4) hidden layers and one output layer (fraud or valid transaction). The system weight was randomly assigned using python package. The dataset was trained using the back propagation training technique and it was trained two thousand times (2000). The training section took a lot of time to process in those 2000 times therefore, the training section was incremented in fifty (50) steps so as to increase the rate at which it trains. The "Time" and "Amount" column was dropped because its dataset variables weren't corresponding with the rest of the dataset variables.

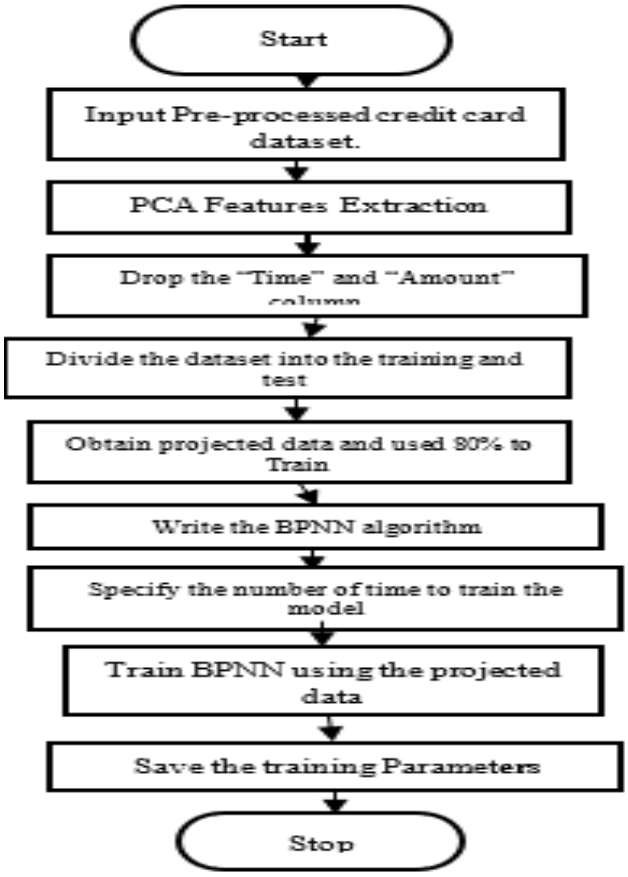

Figure 4a: Training process of BPNN

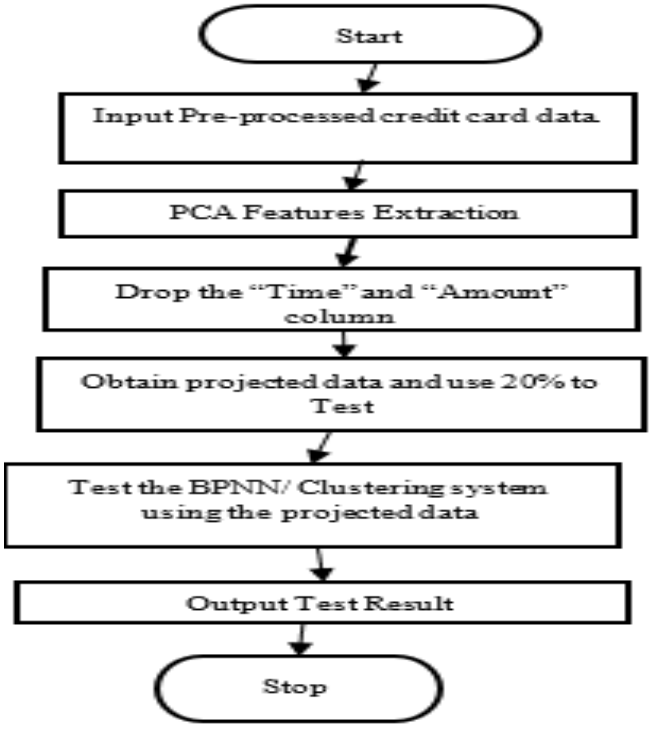

Figure 4b: Testing process of BPNN

The system accuracy changes with the incremental steps, the accuracy gotten when the step was increased to fifty is different from the accuracy gotten when trained without using the steps. Whenever the model was run, the weight initialized changes, therefore, so does the accuracy of the system changes which led to same changes in the test accuracy. The system was designed in such a way that the train accuracy was shown together with the test accuracy per every 50 steps.

The input layer was multiplied to all the hidden layers in a matrix form using Tensorflow python package. All input layer was multiplied to the hidden layers and from the hidden layer down to the output layer which determines whether it is fraudulent or valid. Figure 6 below shows the snapshot of BPNN training process

Figure 7a shows the training and testing accuracy per each 50 steps of the sections. The loss values of training and testing section is shown in Fig. 7b. The higher the training accuracy, the better the model. The loss value implies how well or poorly a certain model behaves after each iteration, the lower the loss, the better the model.

This model was developed on the already prepared dataset. The k-cluster was defined to be two (2), that is, $k= 2$, fraud → 1 and valid transactions → 0. The model checked each transaction and its closeness to the defined cluster, and then assigned each transaction to the closet clusters. The K-means algorithm is an unsupervised learning algorithm, it doesn't need the transaction to be labelled or weight should be assigned. Therefore, the "Time" and "Class" column was dropped. The model was able to separate the fraudulent transaction away from the valid transactions all by itself but the accuracy was low because the dataset used was highly positively skewed which resulted to a lot outlier which doesn't fit into any of the defined clusters.

```
[14]: data
```

:[14]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V22	V23	V24	V2
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	0.277838	-0.110474	0.066928	0.1285
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.638672	0.101288	-0.339846	0.1671
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.771679	0.909412	-0.689281	-0.3276
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	0.005274	-0.190321	-1.175575	0.6473
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	0.798278	-0.137458	0.141267	-0.2060
5	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	-0.371407	...	-0.559825	-0.026398	-0.371427	-0.2327
6	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	-0.099254	...	-0.270710	-0.154104	-0.780055	0.7501
7	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	1.249376	...	-1.015455	0.057504	-0.649709	-0.4152
8	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	-0.410430	...	-0.268092	-0.204233	1.011592	0.3732
9	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	-0.366846	...	-0.633753	-0.120794	-0.385050	-0.0697
10	1.449044	-1.176339	0.913860	-1.375667	-1.971383	-0.629152	-1.423236	0.048456	-1.720408	1.626659	...	0.313894	0.027740	0.500512	0.2513
11	0.384978	0.616109	-0.874300	-0.094019	2.924584	3.317027	0.470455	0.538247	-0.558895	0.309755	...	0.238422	0.009130	0.996710	-0.7673
12	1.249999	-1.221637	0.383930	-1.234899	-1.485419	-0.753230	-0.689405	-0.227487	-2.094011	1.323729	...	-0.483285	0.084668	0.392831	0.1611
13	1.069374	0.287722	0.828613	2.712520	-0.178398	0.337544	-0.096717	0.115982	-0.221083	0.460230	...	0.074412	-0.071407	0.104744	0.5482
14	-2.791855	-0.327771	1.641750	1.767473	-0.136588	0.807596	-0.422911	-1.907107	0.755713	1.151087	...	0.222182	1.020586	0.028317	-0.2327

Figure 5: The credit card transaction dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import tensorflow as tf
import matplotlib.gridspec as gridspec
import math
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

feature_num = 29
hidden1_unit = 10
hidden2_unit = 10
hidden3_unit = 10
hidden4_unit = 10
hidden5_unit = 10

weights1 = tf.Variable(tf.truncated_normal([29,hidden1_unit],stddev=1.0/math.sqr
biases = tf.Variable(tf.zeros(hidden1_unit),name = 'biases')
hidden1 = tf.nn.sigmoid(tf.matmul(x,weights1)+biases)
hidden1 = tf.nn.dropout(hidden1, 0.5)

weights2 = tf.Variable(tf.truncated_normal([hidden1_unit,hidden2_unit],stddev=1.
biases = tf.Variable(tf.zeros(hidden2_unit),name = 'biases')
hidden2 = tf.nn.sigmoid(tf.matmul(hidden1,weights2)+biases)
```

Figure 6: Snapshot of BPNN training process

```
In [20]: data=data.sample(frac=1)#randomize the whole dataset
         full_features=data.drop(["Time","Class"],axis=1)
         full_labels=pd.DataFrame(data[["Class"]])
         full_features_array=full_features.values
         full_labels_array=full_labels.values
         train_features,test_features,train_labels,test_labels=train_test_split(full_features_array,full_labels_array,train_size
         train_features=normalize(train_features)
         test_features=normalize(test_features)

         C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2179: FutureWarning: From version 0.21,
         test_size will always complement train_size unless both are specified.
           FutureWarning)

In [10]: #k_means_classification --> k_means_clustering
         kmeans=KMeans(n_clusters=2,random_state=0,algorithm="elkan",max_iter=2000)
         kmeans.fit(train_features)
         kmeans_predicted_train_labels=kmeans.predict(train_features)
```

Figure 7 Snapshot of K-means Clustering training process

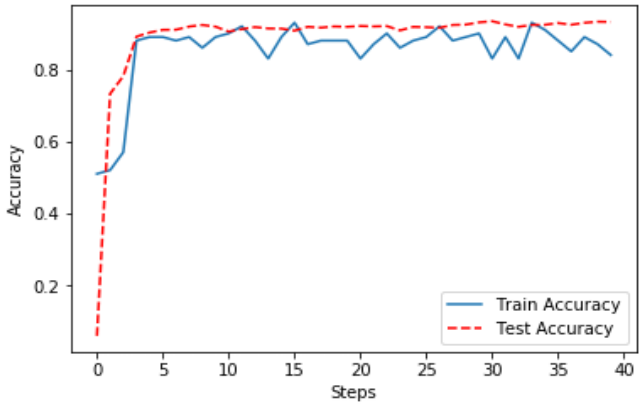

Figure 7a: The Training and Testing Loss

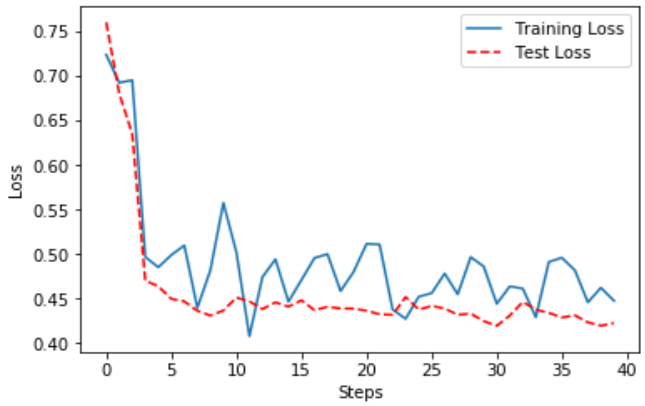

Figure 7b: The Training and Testing Loss

The K-Means model

Results and Discussion

Table 1, as shown below represents the results of the systems developed, namely K-Means base system, and BPNN based system. On evaluation using the considered metrics, it shows that the developed BPNN system was more accurate than the K-mean algorithm due to the nature of the dataset used. The BPNN had a system accuracy of 93.1% while the K-Means had 79.9% system accuracy. BPNN based system had an Error rate of 0.07 compared with the K-means based system which had 0.20. This implies that the developed BPNN system is least tolerant in falsely accepting fraudulent transactions as valid. The BPNN model had a Specificity value of 0.98% compared to the K-means system which had 0.99. This implies that the BPNN system is least tolerant in causing false alarm rate.

Table 1: Performance Summary of BPNN and K-means

Evaluation Metrics	BPNN	K-means
Precision	0.84	0.01
Recall (Sensitivity)	0.95	0.64
Error Rate	0.07	0.20
False Positive Rate (FPR)	0.98	0.99
Prediction Accuracy	0.93	0.80
Prediction Accuracy	93.1%	79.9%

In order to evaluate the performance of the two algorithms, i.e. to determine if there is a significant difference between the two algorithms, the two-way ANOVA was used to determine this using the result of the evaluation metrics shown in Table 1. The summary of the analysis is shown in Tables 2 and 3 respectively.

From Table 2, the sum, average and variance was calculated for each rows and columns. The Rows represent the evaluating metrics, that is, The Precision, Recall, Error rate, False positive (FPR), Prediction accuracy and the System accuracy. The column represents the algorithms, that is, The BPNN and the K-means.

Table 3 uses the results: sum, average and variance from Table 2 to calculate the sum of squares (SS), mean of square (MS), degree of freedom (df), Fisher's transformation (*F)*, P-value ad F-critical for rows and columns. Also, the corresponding errors and total were calculated.

To evaluate the performance, we draw two hypotheses H0 and H1 and a decision rule for both rows and columns as shown below:

H0: There is significant difference between BPNN and K-means.

H1: There is no significant difference between BPNN and K-means

Decision rule: Reject H0 if F-critical < F, otherwise, do not reject.

From Table 3, considering the row, *F*-critical < *F*, therefore we reject H0 and accept H1. Considering the column, F-critical > F, therefore we do not reject H0. In conclusion, Since F-critical > F for the column, we can conclude that there is significant difference between BPNN and K-means.

Table 2: Two-way ANOVA of BPNN and K-means

SUMMARY	Count	Sum	Average	Variance
Precision	2	0.85	0.425	0.34445
Recall (Sensitivity)	2	1.59	0.795	0.04805
Error Rate	2	0.27	0.135	0.00845
False Positive Rate (FPR)	2	1.97	0.985	0.00005
Prediction Accuracy	2	1.73	0.865	0.00845
Accuracy	2	173	86.5	87.12
BPNN	6	96.87	16.145	1421.416
K-means	6	82.54	13.75667	1050.121

Table 3: Summary of ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Rows	12287.27	5	2457.454	174.4928	0.0000132	5.05032
Columns	17.11241	1	17.11241	1.215076	0.3205386	6.60789
Error	70.4170 4	5	14.08341			
Total	12374.8	11				

Summary and Conclusion

This paper presents the application of BPNN and K-means algorithm to fraud detection in an online credit card transaction. These are algorithms that are efficient and they are vital for machine learning. Based on the results of the ANOVA two-way factor, it can be concluded that there is a significance difference between BPNN and K-means in fraud detection in an online credit card transaction,. The BPNN model is of greater accuracy and has least tolerant for raising false alarms compared to K-means model. Hence, this work has contributed to the body of knowledge by successfully demonstrating the effectiveness of BPNN and K-means for fraud detection in online credit card transactions. In addition, we were able to compare the result of the detection model, which indicated show that BPNN performs better than K-means.

Recommendation for Further Works

It is recommended that future work can be carried out by comparing the effect of combing these two models together so as to optimize the system accuracy.

References

Aderounmu, G.A., Adewale, O.S., Alese, B.K., Ismaila, W.O. & Omidiora, E.O. (2012). Investigating the effects of Threshold in Credit Card Fraud Detection System. *International Journal of Engineering and Technology.* 2(7), 328-1332.

Agrawal, A., Kumar, S. & Mishra, A.K. (2015). Credit Card Fraud Detection: A case study. 2nd *International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi* 5-7.

Akshata, H. & Sheetal, Y. (2015). Online Credit Card Fraud Detection. *International Journal for Research in Engineering Application and Management* 1(2), 1-3.

Asiedu, L., Adebanji, A.O., Oduro, F.T. & Mettle, F.O. (2016). Statistical Assessment of PCA/SVD and FFT-PCA/SVD on Variable Facial Expressions. *British Journal of Mathematics & Computer Science* 12(6), ISSN: 2231-0851.

Avinash, I. & Thool, R. C. (2013). Credit Card Fraud Detection Using Hidden Markov Model and Its Performance. *International Journal of Advanced Research in Computer Science and Software Engineering,* 3(6), 24-31.

Banerjee, R., Bourla, G., Chen, S., Kashyap, M., Purohit S. & Battipaglia, J. (2018). Comparative Analysis of Machine Learning Algorithms through Credit Card Fraud Detection. *New Jersey's Governor's School of Engineering and Technology.* Retrieved from https://www.soe.rutgers.edu on 6th February, 2019.

Behera, T.K. & Panigrahi, S. (2015). Credit Card Fraud Detection: A Hybrid Approach Using Fuzzy Clustering & Neural Network. *In Proceedings of the 2015 Second International Conference on Advances in Computing and Communication Engineering (ICACCE)*, Dehradun, India, 494–499.

Chan, P.K., Fan, W., Prodromidis, A. L. & Stolfo, S. J. (1999). Distributed data mining in credit card fraud detection, *Intelligent Systems and their Applications, IEEE* 14(60), 67-74.

Demla, N. and Agrawal, A.N. (2016). Credit card fraud detection using SVM and Reduction of false alarms, *International Journal of Innovations in Engineering and Technology* 7(2). 176-182.

Devaki, R., Kathiresan, V. & Gunasekaran, S. (2014). Credit Card Fraud Detection using Time Series Analysis. *International Journal of Computer Applications Proceedings on International Conference on Simulations in Computing Nexus ICSCN*(3)*,* 8-10.

Dhanapal, R. & Gayathiri, P. (2012). Credit Card Fraud Detection Using Decision Tree for Tracing Email and IP, *International Journal of Computer Science* 9(5), 406-412.

Esmaily, J., Moradinezhad, R. & Ghasemi, J. (2015). Intrusion detection system based on Multi-Layer Perceptron Neural Networks and Decision Tree. *7th Conference on Information and Knowledge Technology (IKT)*, *Urmia* 1-5, doi: 10.1109/IKT.2015.7288736.

Falaki, S.O, Alese, B.K., Adewale, O.S., Ayeni, J., Aderounmu, G.A. & Ismaila, W.O. (2012). Probabilistic Credit Card Fraud Detection

System in Online Transactions. *International Journal Software Engneering Application* 6(4), 69–78.

Fashoto, S., Adeleye, O. & Wandera, J. (2016). Hybrid Methods for Credit Card Fraud Detection Using K-means Clustering with Hidden Markov Model and Multilayer Perceptron Algorithm. *British Journal of Applied Science & Technology* 13(5), 1-11.

Khan, M.Z., Pathan, J.D., & Ahmed, A. H. (2014), Credit Card Fraud Detection System Using Hidden Markov Model and K-Clustering, *International Journal of Advanced Research in Computer and Communication Engineering* 3(2), 2319-5940.

Navanshu, K. & Saad, Y. S. (2018), Credit Card Fraud Detection using Machine Learning models and collating Machine Learning models. *International Journal of Pure and Applied Mathematics* 118(20), 825-838.

Patel, S. & Gond, S. (2014). Supervised Machine (SVM) Learning for Credit Card Fraud Detection. *International Journal of Engineering Trends and Technology* 8(3), 137-139.

Pooja, C., Thakare, A.D., Prajakta, K., Madhura, G. & Priyanka, N. (2015). Genetic K-means Algorithm for Credit Card Fraud Detection. *International Journal of Computer Science and Informaion Technologies* 6(2), 1724-1727.

Pouramirarsalani, A., Khalilian, M., & Nikravabshalmani, A. (2017). Fraud Detection in E-banking by Using the Hybrid Feature Selection and Evolutionary Algorithms. *International Journal of Computer Science and Network Security* 17(8), 271-279.

Raj, S. B. & Portia, A. A. (2011). Analysis on Credit Card Fraud Detection Methods. *International Conference on Conference on Computer Communication and Electrical Technology* https://doi.org/10.1109/ICCCET.2011.5762457.

Samaneh, S., Zahra, Z., Ebrahimi, A., & Amir, H. M. (2016). A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective. *arXiv preprint arXiv:1611.06439*.