A Comparative Study on Credit Card Fraud Detection

Souvik Ghoshal
Institute Of Engineering and Management(IEM)
souvikghosal748@gmail.com

Debkanya Bose
Institute Of Engineering and Management(IEM)
debkanyabose44@gmail.com

Koushik Deb
Institute Of Engineering and Management(IEM)
Koushikdeb2009@gmail.com

Abstract. Credit Card Fraud is increasing rapidly with the development of modern technology. This fraud detection system has been proven essential for banks and financial institution, to minimize their losses. This paper proposes Credit Card Fraud Detection using clustering based on several unsupervised Machine learning and deep learning algorithms. The method we follow to solve our problem is that we are going to plot the points into two dimensional space and some points turns out to be an outliers and some points forms a valid clusters. These outliers are possible number of cheaters which is nothing but the fraudulent transactions and the bank may reject their credit card application. And valid clusters are not cheaters therefore we are going to allocate them the credit card. So as a result we get the explicit list of customers i.e. the potential cheaters who have cheated. Thus, the clustering approach which will give better rating score can be chosen to be one of the best methods to detect fraud. In this paper, we worked with Statlog Australian Credit Card Approval Dataset in which the dependent variables have been removed to maintain the privacy of the customers.

1 Introduction

At the current state of the world, financial organizations expand the availability of financial facilities by employing of innovative services such as credit cards, Automated Teller Machines(ATM), internet and mobile banking services. Credit card is a payment card supplied to customers as a system of payment. There are lots of advantages in using credit cards such as:

- **Ease of purchase**
- **Keep customer credit history**
- **Protection of Purchase**

In spite of all mentioned advantages, the problem of fraud is a serious issue in e-banking services that threaten credit card transactions especially. Since credit card is the most popular mode of payment, the number of fraud cases associated with it is also rising. Fraud detection involves identifying fraud as quickly as possible once it has been done. Fraud detection methods are continuously developed to defend criminals in adapting to their strategies. Fraud detection also involves identifying scarce fraud activities among numerous legitimate transactions as quickly as possible. Fraud detection methods are developing rapidly in order to adapt with new incoming fraudulent strategies across the world. But, development of new fraud detection techniques becomes more difficult due to the severe limitation of the ideas exchange in fraud detection. On the other hand, fraud detection is essentially a rare event problem, which has been variously called outlier analysis, anomaly detection, exception mining, mining rare classes, mining imbalanced data etc. The number of fraudulent transactions is usually a very low fraction of the total transactions. Hence the task of detecting fraud transactions in an accurate and efficient manner is fairly difficult and challengeable.

For this reason ,we study methods for preventing illegal or fraudulent credit card transactions. We learnt that in 2015, [12]Tanmay Kumar and Suvasini Panigrahi in th-eir paper proposed a hybrid approach to credit card fraud detection using fuzzy cluste-ring and neural network. It makes use of two phases. In phase one, they used a c-means clustering algorithm to generate a suspicious score of the transaction and in next phase if a transaction is suspicious it is feed into neural network to determine whether it was really fraudulent or not. In their paper the authors 'Wen-Fang Yu' and 'Na Wang' [13] proposed credit card fraud detection using outlier mining based on distance sum. Outl-ier mining is a field of data mining which is basically used in monetary and internet fields. It deals with detecting objects that are detached from the main system i.e. the transactions that aren't genuine. Sam Maes[15] proposed detecting frauds in credit ca-rd using two machine learning techniques namely Bayesian Netwoks and Artificial Neural Network. The paper discussed that how Bayesian networks after a short traini-ng gave good results and their speed was enhanced by the use of ANN. So, we proposed some clustering techniques using some machine learning algorithms – Simple K-means clustering, K-means clustering along with Principal Component Analysis (PCA), T-distributed Stochastic Neighbor Embedding(t-SNE) and one deep learning Algorithm –Self Organising Map(SOM).

In this paper, the framework we propose will accurately find the cheater who have not followed the norms and regulation of the bank. Here we will propose several unsupervised Machine learning and deep learning algorithms which helps us to plot the clusters within the data and from the clusters we can easily find the outliers within the clusters and we can say that the outliers are nothing but the cheaters and the bank may reject their credit card application. So our objective is to detect the potential fraud wit-hin these applications so that means by the end we will be giving the explicit list of the customers who have cheated.

2 Credit Card Fraud

Illegal use of credit card or its information without the knowledge of the owner is referred to as credit card fraud. Different credit card fraud tricks belong mainly to two groups of application and behavioral fraud [2]. Application fraud takes place when, fraudsters apply new cards from bank or issuing companies using false or other"s information. Multiple applications may be submitted by one user with one set of user details (called duplication fraud) or different user with identical details (called identity fraud).

Based on statistical data stated in [1] in 2012, the high risk countries facing credit card fraud threat is illustrated in Fig.1. Ukraine has the most fraud rate with staggering 19%, which is closely followed by Indonesia at 18.3% fraud rate. After these two, Yugoslavia with the rate of17.8% is the most risky country. The next highest fraud rate belongs to Malaysia (5.9%), Turkey (9%) and finally United States. Other countries that are prune to credit card fraud with the rate below than 1% are not demonstrated in(Fig 1).

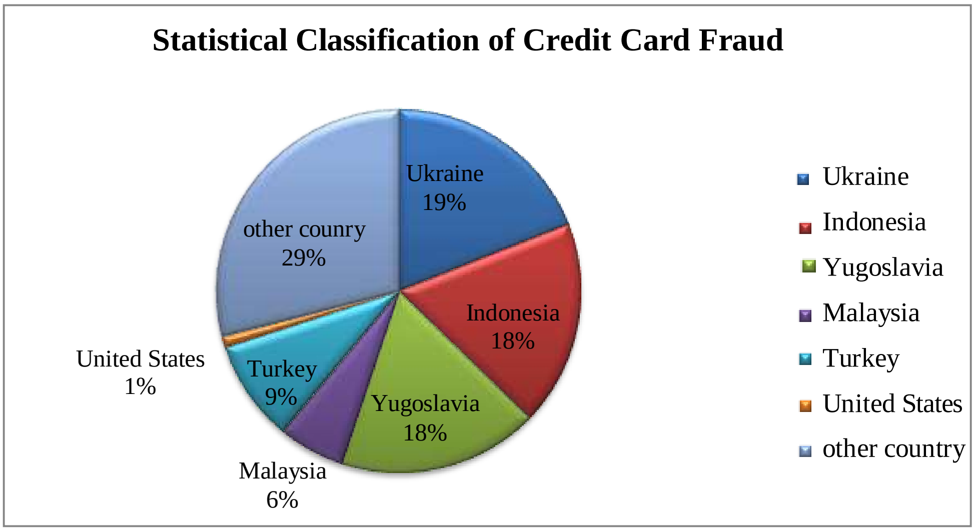


Fig. 1. High risk countries facing credit card fraud threat

The dataset that we are using for fraud detection in this paper is called the Statlog Australian Credit Card Approval Dataset. This dataset is interesting because there is a g-ood mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values. In this dataset, the columns are the attributes i.e. the information of the customers and the rows are the customers itself. The model will identify some patterns i.e. customers. So we will be doing some kind of customer segmentation to identify segments of customers and one of the segments will contain the customers that have potentially cheated.

3 Credit Card Fraud Detection Techniques

In this paper, we will be using four different algorithms on our datasets and will compare the output given by each algorithm for credit card fraud detection. So the number of algorithm we will be using are as follows:

- The simple K-means clustering algorithm
- K-means clustering along with Principal Component Analysis (PCA)
- T- distributed Stochastic Neighbor Embedding (t-SNE)
- Self Organizing Map (SOM)

We will see how these above works and how they find the outliers and the valid clusters from two dimensional data space. The first three algorithms are unsupervised machine learning algorithm and the last algorithm is deep learning algorithm. .In this section we will briefly introduce these fraud detection techniques which are applied to credit card fraud detection tasks, also the drawbacks of each approach will be discussed.

3.1 K-means Clustering

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given dataset through a certain number of clusters (assume k clusters) fixed. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The step is to take each point belonging to a given data set and associate it to the center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more.

3.1.1 Flowchart for K-means-

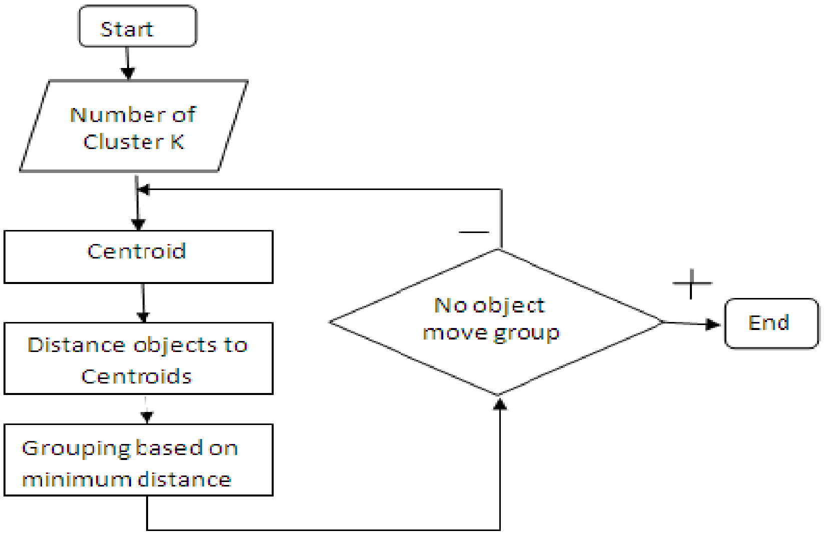

Start
Number of Cluster K
Centroid
Distance objects to Centroids
Grouping based on minimum distance
No object move group
—
+
End

Fig 2. An overview of K-means

3.1.2 Algorithm for K-means-

Step 1: Choose the number K of clusters.
Step 2: Select at random K points, the centroids (not necessarily from our datasets)
Step 3: Assign each data point to the closest centroid from the k clusters .
Step 4: Compute and place the new centroid of each other.
Step 5: Reassign each data point to the new closest centroid . If any reassignment took place then go to step 4 otherwise finish.

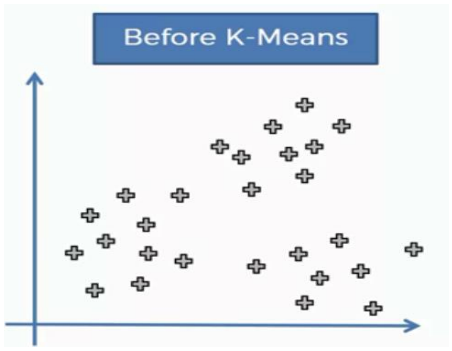


Fig. 3.

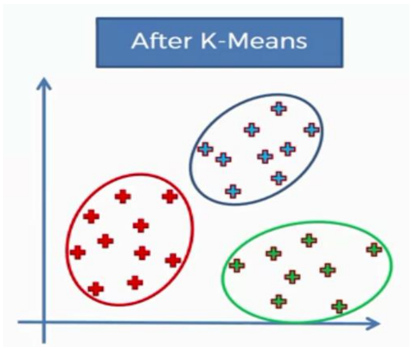


Fig. 4.

So from the above figure we can clearly see that in (Fig. 3), the raw data points are plotted in two dimension space and after applying k-Means algorithm all the points that belongs the same cluster are separated so from the above picture (Fig. 4) we can say that our data points are divided into three clusters which share same centroid.

3.1.3 Drawbacks of K-means-

- It is impossible to find the outliers from the above data.
- From the (Fig.14.) it is very clear that K means algorithm fails for large dimensional datasets.

The problem that K-means suffers is called "curse of Dimensionality"
There are several methods by which we can reduce the dimensions but here we have used three methods to solve this problem.

3.2 K-means Clustering along with Principal Component Analysis(PCA)

As the name suggests we are going to choose some principal component that has more variance than other components. PCA is an algorithm that compresses our dataset's dimensions from a higher to lower dimensionality. It does this based on the Eigen vectors of the variance in our dataset. PCA is extremely used in data compression saving loads in processing time, as well as in visualizations of high dimensional data in 2 or 3 dimensions. That is very helpful when doing cluster analysis. More technically, PCA finds a new set of dimensions such that –

- All the dimensions are orthogonal (and thus linearly independent)
- Ranked according to the variance of data along them. This means the first principal component contains the most information about the data.

Mathematically how PCA is done-:
Let X be a point represented in a specific given dimensional space and we are trying to convert it into another dimensional space(D') with less number of dimensions.
i,e D----→D'

where(D’<D)

Let us consider we have two features f1 and f2 and f1 represents hair colour of indian people and let f2 denotes the height of people if we represent the data in a matrix form
case 1-:

$$\left[\begin{array}{cc} \mathbf{f1} & \mathbf{f2} \\ x1 & y1 \\ x2 & y2 \\ x1 & y3 \\ x2 & y4 \\ x1 & y5 \end{array}\right]$$

Fig. 5.

After plotting the points we get -:

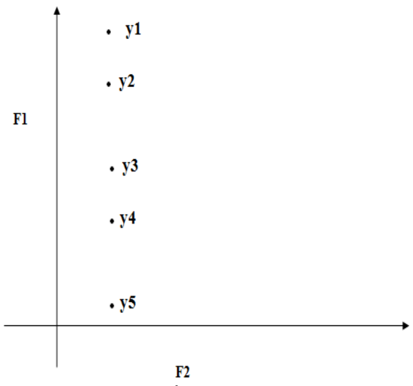


Fig. 6.

Therefore we can say that f1 has very high variance in comaparison to f2 and and we can exclude f2.
Case2-: when there is high variance in both f1 and f2

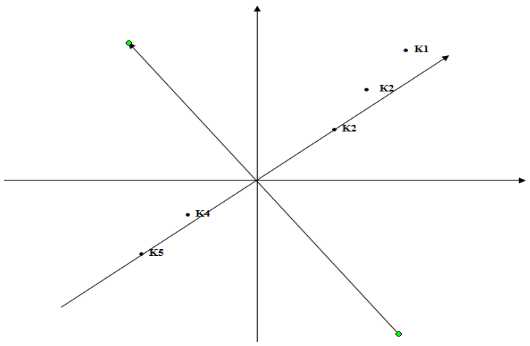


Fig. 7.

We have defined a new dimensional space f1’ and f2’
spread on f1’ >>f2’
So our sole objective is to find out the direction of
Find the direction of f1’ and f2’ because they have maximum variance
Let the number of data points be N and the number of dimension be D

Variance$=\frac{1}{N}\sum_{N=1}^{N}(u_1^T\ x_n - u_1^T\ \overline{x})^2$

where $u_1^T\ x_n$ is the projection of x_n on u1 and $u_1^T\ \overline{x}$ is the mean distance .
$=\frac{1}{N}\sum_{N=1}^{N} u_1^T\ (x_n-\overline{x})(x_n-\overline{x})^T u_1$
$= u_1^T[\ \frac{1}{N}\sum_{N=1}^{N}(x_n-\overline{x})\ (x_n-\overline{x})^T]u_1$
$= u_1^T S\ u_1$
where S is closed form of covariance matrix
We have to find the maximum to find the maximum of $u_1^T S\ u_1$ where the constraint is $u_1^T u_1 =1$
that means u_1 is unit vector
to find this type of maximisation problem we use Lagrange Multipliers .
$\frac{d}{du_1}[\ u_1^T\ S\ u_1\ + \lambda(1- u_1^T u_1\ )\ ]$

$2Su_1- \lambda 2U_1=0$
$S\ u_1= \lambda U_1$
$\lambda= S$
And that means λ is the eigen value and U is the unit vector so to find the maximum variance we have to take the maximum eigen value and the eigen vector corresponding to it contains the maximum variance.
So our problem reduces to find the largest eigen value and the vector corresponding to it.

3.2.1 Flowchart for PCA-

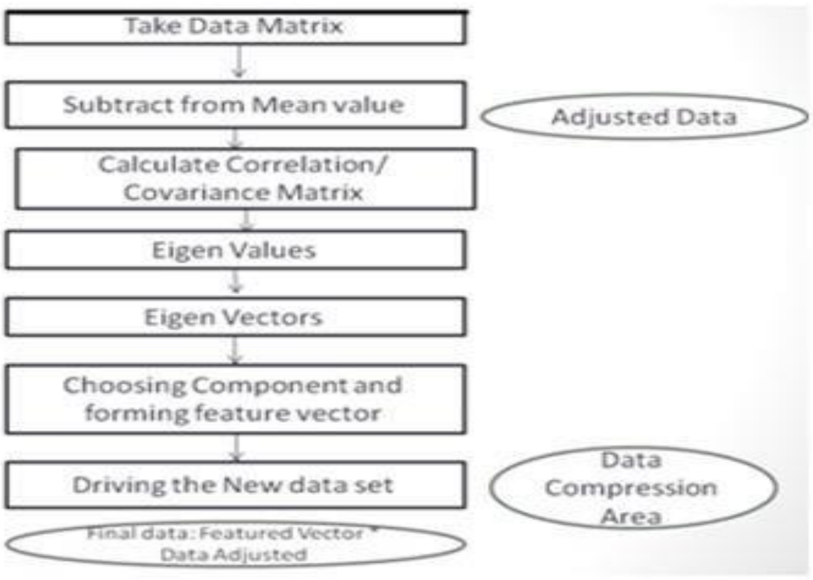

Take Data Matrix
Subtract from Mean value
Adjusted Data
Calculate Correlation/ Covariance Matrix
Eigen Values
Eigen Vectors
Choosing Component and forming feature vector
Driving the New data set
Data Compression Area
Final data: Featured Vector * Data Adjusted

Fig. 8. An overview of PCA

3.2.2 Algorithm for PCA-

Step 1: Standardize the data first.
The input data are standardized or normalized to allow each attribute to fall within the same range. This step helps to ensure that larger domain attributes do not dominate attributes with smaller domains.

Step 2: Calculate data point's covariance matrix X.
The purpose of this step is to understand how the variables of the input data set variables to vary from the mean to each other, or in other words, to see if there is any relationship between them.

Step 3: Calculate the Eigenvectors and their related Eigenvalues.
PCA calculates k-orthogonal vectors (or) eigenvectors which provide basic standard input data. These are unit vectors, each of which points in a direction perpendicular to the others. These vectors are referred to as PCs. The input data is the linear combination of the PCs.

Step 4: Sort the eigenvectors in decreasing order according to their eigenvalues.
The PCs are arranged in the descending order based on its significance (or) strength. The PCs basically provide important information about variance and serve as a new axis for the data.

Step 5: Select the first k eigenvectors and that will be the new k dimensions.
Now select only the first k- PCs with the highest variance and remove the weaker PCs with the lowest variance.

3.2.3 Drawbacks of PCA-

- **Independent variables become less interpretable:** After implementing PCA on the dataset, your original features will turn into Principal Components. Principal Components are the linear combination of your original features. Principal Components are not as readable and inter-pretable as original features.
- **Data standardization is must before PCA:** We must standardize our data before implementing PCA, otherwise PCA will not be able to find the optimal Principal Components. Hence, principal components will be biased towards features with high variance, leading to false results.
- **Information Loss :** Although Principal Components try to cover maximum variance among the features in a dataset, if we don't select the number of Principal Components with care, it may miss some information as compared to the original list of features.

3.3 T- Distributed Stochastic Neighbor Embedding (t-SNE)

The purpose of using t-SNE is same as PCA which is reducing the dimension. It models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. The t-SNE algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar points have an extremely small probability of being picked. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map. Note that while the original algorithm uses the Euclidean distance betw-een objects as the base of its similarity metric, this should be changed as appropriate.

3.3.1 Flowchart of t-SNE –

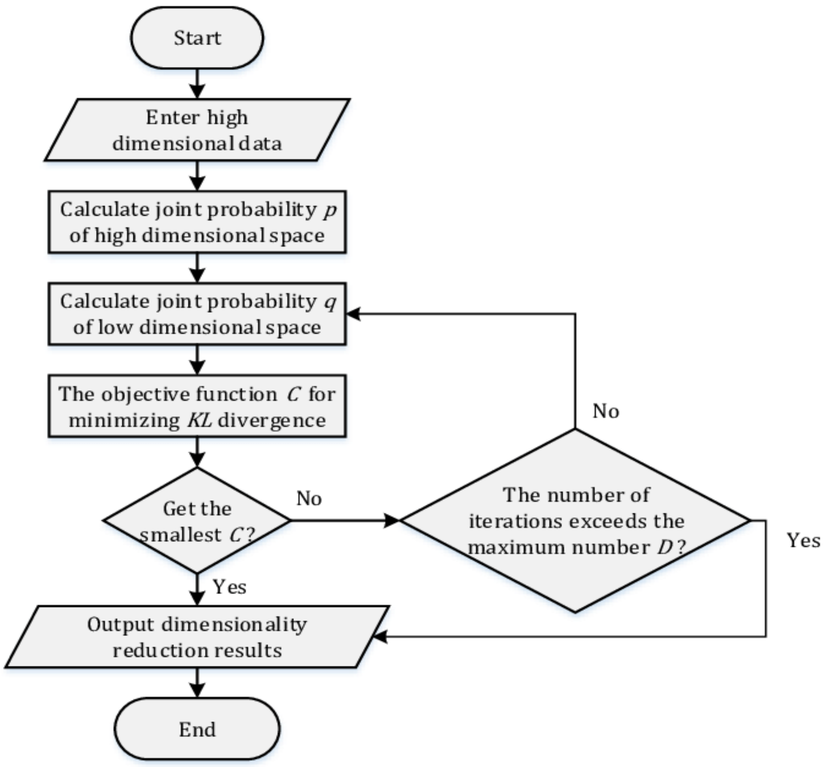

Start
Enter high dimensional data
Calculate joint probability p of high dimensional space
Calculate joint probability q of low dimensional space
The objective function C for minimizing KL divergence
Get the smallest C?
No
Yes
The number of iterations exceeds the maximum number D?
No
Yes
Output dimensionality reduction results
End

Fig. 9. An overview of t-SNE

3.3.2 Algorithm of t-SNE –

Step 1: The algorithms starts by calculating the probability of similarity of points in high-dimensional space and calculating the probability of similarity of points in the corresponding low-dimensional space. The similarity of points is calc-ulated as the conditional probability that a point A would choose point B as its neighbor if neighbors were picked in proportion to their probability density un-der a Gaussian (normal distribution) centered at A.

Step 2: It then tries to minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perf-ect representation of data points in lower-dimensional space.

Step 3: To measure the minimization of the sum of difference of conditional probability t-SNE minimizes the sum of Kullback-Leibler divergence (KL divergence) of overall data points using a gradient descent method.

3.3.3 Drawbacks of t-SNE -

Problems with t-SNE arise when intrinsic dimensions are higher i.e. more than 2-3 dimensions. t-SNE has the tendency to get stuck in local optima like other gradient descent based algorithms. The basic t-SNE algorithm is slow due to nearest neighbor search queries.

3.4 Self Organizing Map (SOM)

Self organizing map (SOM) is a type of Artificial Neural Network (ANN) which is one of the most popular unsupervised neural networks learning which was introduced by [3]. SOM provides a clustering method, which is appropriate for constructing and analyzing customer profiles, in credit card fraud detection, as suggested in [4]. SOM operates in two phase: training and mapping. In the former phase, the map is built and weights of the neurons are updated iteratively, based on input samples [5], in latter, test data is classified automatically into normal and fraudulent classes through the procedure of mapping. As stated in [6], after training the SOM, new unseen transactions are compared to normal and fraud clusters, if it is similar to all normal records, it is classified as normal. New fraudulent transactions are also detected similarly by the outliers.

3.4.1 Flowchart of SOM –

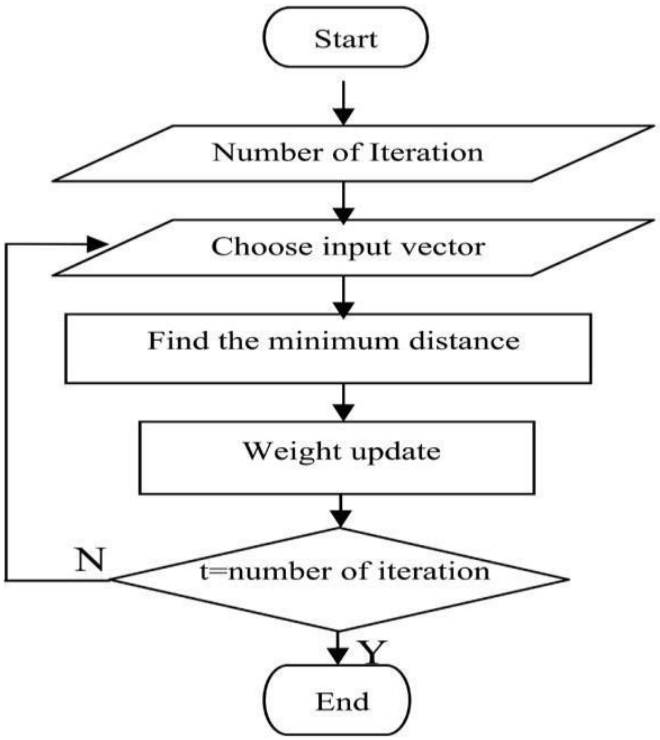


Fig. 10. An overview of SOM

3.4.2 Algorithm of SOM –

The first step in the learning process of self-organizing maps is the initialization of all weights on connections. After that, a random sample from the dataset is used as an input to the network. The network then calculates weights of which neuron are most like the input data (input vector). For this purpose, this formula is used:

$$Distance^2 = \sum_{i=0}^{n}(input_i - weight_i)^2$$

where *n* is the number of connection (weights). The map neuron with the best result is called Best Matching Unit or BMU. In an essence, this means that the input vector can be represented with this mapping neuron. Now, the self-organizing maps are not just calculating this point during the learning process, but they also try to make it "closer" to the received input data.

This means that weights on this connection are updated in a manner that calculated distance is even smaller. Still, that is not the only thing that it is done. The weights of neighbours of BMU are also modified so they are closer to this input vector too. This is how the whole map is 'pulled' toward this point. For this purpose, we have to know the *radius* of the neighbours that will be updated. This radius is initially large, but it is reduced in every iteration (epoch). So, the next step in training self-organizing maps is actually calculating mentioned radius value. Following formula is applied:

$$\sigma(t) = \sigma_0 e^{-\frac{t}{\lambda}}$$

where *t* is the current iteration, *σo* is the radius of the map. The *λ* in the formula is defined like this:

$$\lambda = k/\sigma_0$$

where *k* is the number of iterations. This formula utilizes exponential decay, making radius smaller as the training goes on, which was the initial goal. In a nutshell, this means that every iteration through the data will bring relevant points closer to the input data.

When the radius of the current iteration is calculated weights of all neurons within the radius are updated. The closer the neuron is to the BMU the more its weights are changed. This is achieved by using this formula:

$$weight(t+1) = weight(t) + \Theta(t)L(t)(input(t) - weight(t))$$

This is the main learning formula, and it has a few important points that should be discussed. The first one is *L(t)* which represents the learning rate. Similarly to the radius formula, it is utilizing exponential decay and it is getting smaller in every iteration:

$$L(t) = L_0 e^{-\frac{t}{\lambda}}$$

Apart from that, we mentioned that the weight of the neuron will be more modified if that neuron is closer to the BMU. In the formula, that is handled with the *Θ(t).* This value is calculated like this:

$$\Theta(t) = e^{-distBMU/2\sigma(t)^2}$$

Apparently, if the neuron is closer to the BMU, *distBMU* is smaller, and with that *Θ(t)* value is closer to 1. This means that the value of the weight of such neuron will be more changed. This whole procedure is repeated several times.

To sum it up, these are the most important steps in the self-organizing map learning process:

- Weight initialization
- The input vector is selected from the dataset and used as an input for the network.
- BMU is calculated.
- The radius of neighbours that will be updated is calculated
- Each weight of the neurons within the radius are adjusted to make them more like the input vector
- Steps from 2 to 5 are repeated for each input vector of the dataset

Now, the self-organizing maps are not just calculating this point during the learning process, but they also try to make it "closer" to the received input data. As we can see in the (Fig. 11.), the BMU (the large red circle) is the closest to the data point (the small red circle). As you can also see, as we drag the BMU closer to the data point, the nearby nodes are also pulled closer to that point.

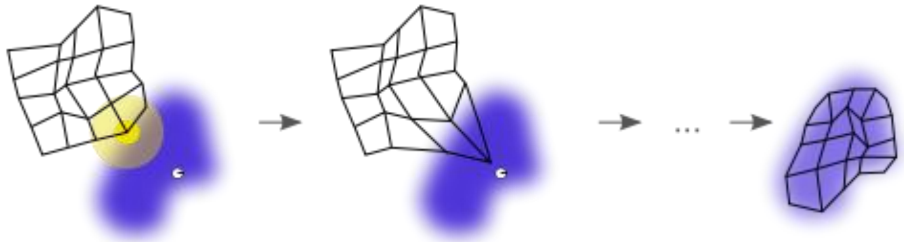


Fig. 11. An illustration of the training of a self-organizing map. The blue blob is the distribution of the training data and the small white disc is the current training data drawn from that distribution. At first (left) the SOM nodes are arbitrarily positioned in the data space. The node (highlighted in yellow) which is nearest to the training data is selected. It is moved towards the training datum, as (to a lesser extent) are its neighbours on the grid. After many iterations the grid tends to approximate the data distribution(right).

3.4.3 Drawbacks of SOM -

- The major disadvantage of a SOM, is that it requires necessary and sufficient data in order to develop meaningful clusters. The weight vectors must be based on data that can successfully group and distinguish inputs. Lack of data or extraneous data in the weight vectors will add randomness to the groupings. Finding the correct data involves determining which factors are relevant and can be a difficult or even impossible task in several problems. The ability to determine a good data set is a deciding factor factor in determining whether to use a SOM or not.
- Another problem with SOMs is that it is often difficult to obtain a perfect mapping where groupings are unique within the map. Instead, anomalies in the map often generate where two similar groupings appear in different areas on the same map. Clusters will often get divided into smaller clusters, creating several areas of similar neurons. This can be prevented by initializing the map well, but that is not an option if the state of the final map is not obvious.

4 Experimental Result

In this paper, we are using these four algorithms which are mentioned in Section3. We have learnt the basic algorithm, flowchart and drawbacks of each techniques. In this section we will see the results of those techniques when they are implemented by using those algorithm.

4.1 Result after applying K-means

Calculations for ‘k’-:
K is calculated based on decreasing value for WCSS (with in cluster sum of square)

$$\mathrm{WCSS} = \sum_{P_i \text{ in Cluster } 1} \mathrm{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster } 2} \mathrm{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster } 3} \mathrm{distance}(P_i, C_3)^2$$

Lower the value of WCSS better will be the result after plotting WCSS value in a graph we will get some structure like elbow and from that we can decide our number of clusters. eg-:

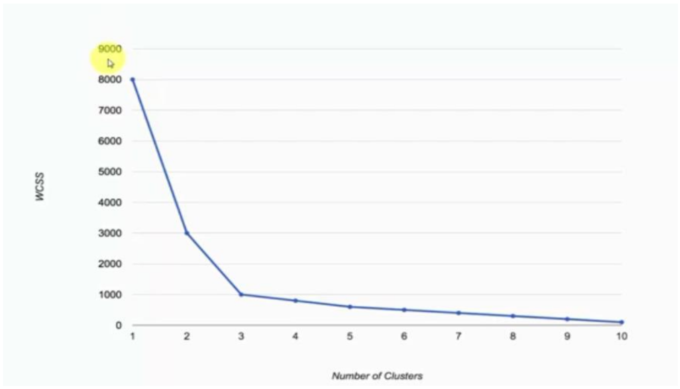


Fig. 12.

After applying elbow method in our dataset-:

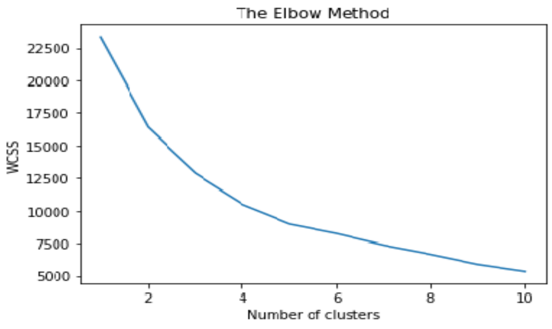


Fig. 13.

So from the above graph we conclude our K value as five.

And our final output-:

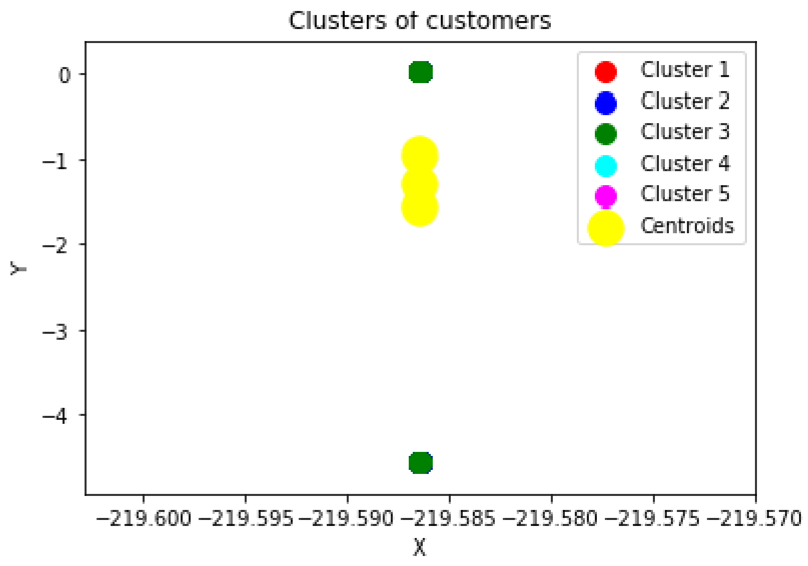


Fig. 14.

4.2 Result after applying PCA

In this graph, we can see the various clusters. Any point which is above the reference line is known as outliers. Therefore, if we identify an outlier in our data, we should examine the observation to understand that it is fraudulent transactions.

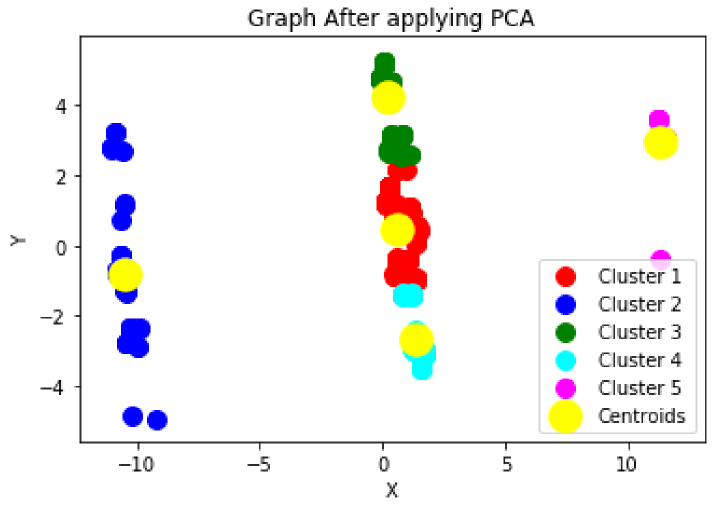


Fig. 15.

4.3 Result after applying t-SNE

In this graph we can see that after many iterations, the clusters get well-formed and can be differentiated.

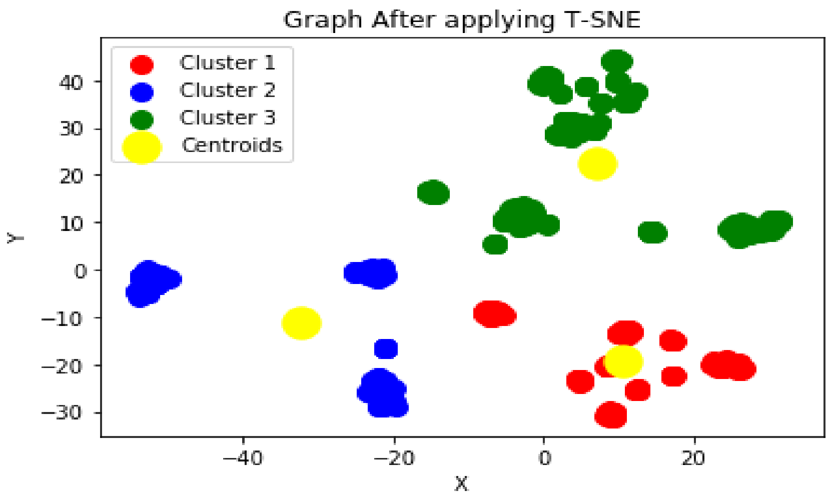


Fig. 16.

4.4 Result after applying SOM

After training the most important thing is to visualize the results i.e. to identify the outline neurons inside the map. So, what we are about to see is clearly a two dimensional grid that will contain all the final winning nodes and for each of these nodes we will get the MID(Mean Interneuron Distance) where MID of a specific winning node is the mean of the distances of all the neurons around the winning node inside a neighbourhood that we defined. And sigma is the radius of the neighbourhood. So basically higher is the MID, then more the winning node will be far away from the neighbours . Therefore higher the MID, the more the winning node is an outlier. So we can detect the frauds by simply taking the winner nodes that have the higher MID. That means the winning nodes will be colored by different colors in such a way that the larger is the MID, the closer to white the color will be.

So for building the map, we are going to add on the map the information of the Mean Interneuron Distance(MID) for all the winning nodes. So we will use pcolor function and we are going to add all the values of the Mean Interneuron Distances for all the winning nodes of our self-organizing map. And to get these mean distances, well we have a specific method for that, it's Distance Map Method and we just need to take the transpose of the MID matrix. So we can see in (Fig. 17.), the self organizing map and legend on the right indicates the range of values of the MID,the Mean Interneuron Distances. But these are normalized values,that means that the values were scaled from zero to one.And therefore now we can clearly see that the highest MIDs,the highest Mean Interneuron Distances, correspond to the white color. And on the other hand, the smallest Mean Interneuron Distances correspond to the dark colors.

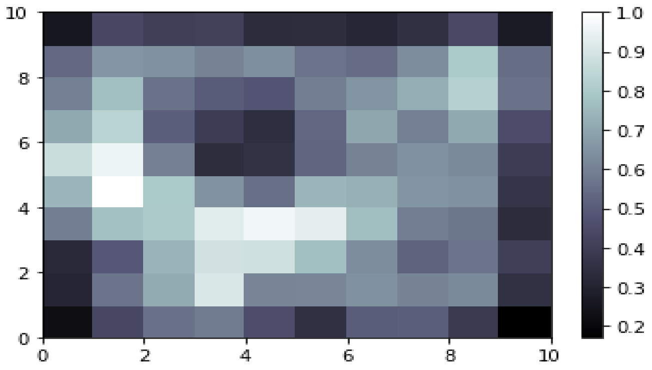


Fig. 17. Self Organizing Map

We can see in (Fig. 17.) all these majority of points here with dark colors are close to each other because their MID is pretty low. So that means that all the winning nodes in the neighborhood of one winning node are close to this winning node at the center and therefore that creates clusters of winning nodes all close to each other. But, these winning nodes here have large MIDs and therefore they're outliers and accordingly potential frauds.

To make it more effective we will add some markers to distinguish between the customers who got approval and the customers who didn't got approval. Because the customers who cheated and got approval are more relevant targets to fraud detection than the customers who didn't get approval and cheated. So to add markers, we're going to create two markers, some red circles and some green squares. The red circles are going to correspond to the customers who didn't get approval. And the green squares will correspond to the customers who got approval. So obviously we can see in (Fig. 18.) outliers that are this winning nodes here which means Mean Interneuron Distance is almost equal to one or perhaps equal to one. Which clearly indicates that there is a high risk of fraud to these customers associated to these two winning nodes. Well we see that we have both cases some customers who got approval and some customers who didn't get approval because we get a green square and also a red circle.

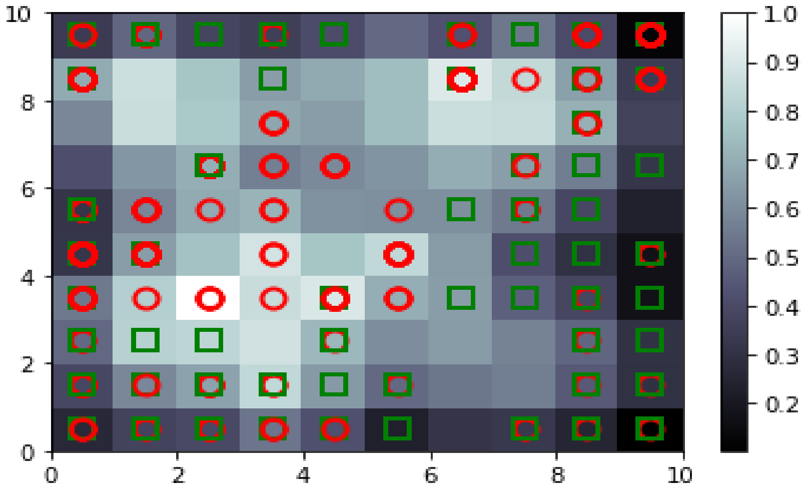
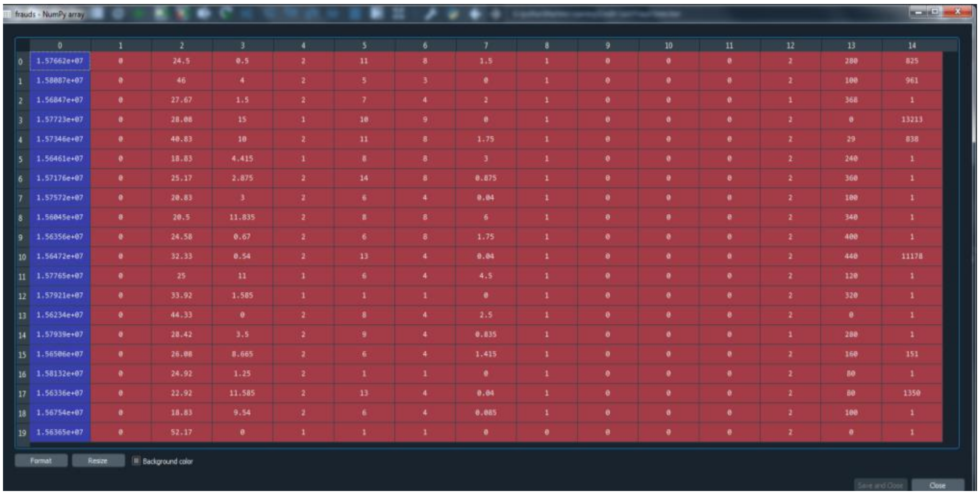


Fig 18. Self Organizing Map with markers

So now, what we have to do is catch these potential cheaters in the winning nodes but in priority those who got approval because it is much more relevant to the bank to catch the cheaters who got away with this. And since we actually identified two outlying winning nodes, we will use the concatenate function to concatenate the two lists of customers so that we can have a whole list of the potential cheaters. So to get explicit list of customers we are doing inverse mapping of the winning nodes to see which customers are associated to this winning nodes. We can see in (Fig. 19.), we have got the list of fraudulent customers but this time with the real values. We get the customer IDs, which are right here and which we can use to identify the potential cheaters.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.57662e+07	0	24.5	0.5	2	11	8	1.5	1	0	0	0	2	280	825
1	1.58087e+07	0	46	4	2	5	3	0	1	0	0	0	2	100	961
2	1.56847e+07	0	27.67	1.5	2	7	4	2	1	0	0	0	1	368	1
3	1.57723e+07	0	28.08	15	1	10	9	0	1	0	0	0	2	0	13213
4	1.57346e+07	0	40.83	10	2	11	8	1.75	1	0	0	0	2	29	838
5	1.56461e+07	0	18.83	4.415	1	8	8	3	1	0	0	0	2	240	1
6	1.57176e+07	0	25.17	2.875	2	14	8	0.875	1	0	0	0	2	360	1
7	1.57572e+07	0	20.83	3	2	6	4	0.04	1	0	0	0	2	100	1
8	1.56045e+07	0	20.5	11.835	2	8	8	6	1	0	0	0	2	340	1
9	1.56356e+07	0	24.58	0.67	2	6	8	1.75	1	0	0	0	2	400	1
10	1.56472e+07	0	32.33	0.54	2	13	4	0.04	1	0	0	0	2	440	11178
11	1.57765e+07	0	25	11	1	6	4	4.5	1	0	0	0	2	120	1
12	1.57921e+07	0	33.92	1.585	1	1	1	0	1	0	0	0	2	320	1
13	1.56234e+07	0	44.33	0	2	8	4	2.5	1	0	0	0	2	0	1
14	1.57939e+07	0	28.42	3.5	2	9	4	0.835	1	0	0	0	1	280	1
15	1.56506e+07	0	26.08	8.665	2	6	4	1.415	1	0	0	0	2	160	151
16	1.58132e+07	0	24.92	1.25	2	1	1	0	1	0	0	0	2	80	1
17	1.56336e+07	0	22.92	11.585	2	13	4	0.04	1	0	0	0	2	80	1350
18	1.56754e+07	0	18.83	9.54	2	6	4	0.085	1	0	0	0	2	100	1
19	1.56365e+07	0	52.17	0	1	1	1	0	0	0	0	0	2	0	1

Fig.19. Customers grouped under the outlier node along with their 14 attributes.

Then we will give the list of potential cheaters to the bank, so now the bank side's to deal with. Their analyst will investigate this list of potential cheaters, what they will probably do is get the values of y for all these customers ID, take in priority the ones that got approved to revise the application, and then by investigating deeper, they will find out if the customer had really cheated.
So the result of Self Organising Map (SOM) are the benchmark of our problem.

5 Comparative Analysis

As per the comparative study, we analysed that between first two methodology, the K-means clustering along with Principal Component Analysis (PCA) is much better than simple K-means clustering algorithm because simple K-means suffers from 'curse of dimensionality' so it does not produce good results. But K-means clustering along with PCA improves the results of clustering by noise reduction.

Although K-means clustering along with Principal Component Analysis (PCA) and T- Distributed Stochastic Neighbour Embedding (t-SNE) both are used for dimensionality reduction. But after analysis through comparative study we get to know that t-SNE is non-linear dimensionality reduction technique whereas PCA is linear dimensionnality reduction technique which means that PCA will fail to find the non-linear (solid-line) path. T-SNE is much better than PCA because PCA gets highly affected by outliers and so it cannot handle outliers whereas t-SNE involves hyperparameters such as perplexity, learning rate and number of steps and it can also handle outliers. The t-SNE algorithm works in a very different way and focuses to preserve the local distances of the high-dimensional data in some mapping to low-dimensional data.

Lastly, when we analysed all the four methodologies we saw that the result of Self Organising Map (SOM) is much better than the other three methodologies. It is because a clustering of a data set will preserve the probability density function of the data set, but not the topological structure of the data set. This makes SOMs especially useful for visualization. By defining a secondary function which transforms a given weight vector into a colour ,we are able to visualize the topology, similarity, and the probability density function of the underlying data set in a lower dimension (usually two dimensions because of the grid). As t-SNE and SOM both are used for non-linear dimensionality reduction , SOM produces better results as it is neural network based algorithm and we also saw that the map obtained in SOM is more clear and the clusters are also better. The SOM method has advantages of data compression. That is, high-dimensional space samples data are mapped into low-dimensional space while keeping the topology unchanged. SOM has clear advantages in this aspect, which other wildly used methods such as PCA do not have. Our model achieved a accuracy of 90% for fraud detection and the affected population (which were considered fraud but were not fraud) was found to be 10%. The results may very if we run the same jupyter notebook because intialization of the weights of the nodes of SOM grid is done by randomly selecting the records/ patterns from the input space i.e randomly selecting the records from the given dataset. Since , we have done training for 100 iterations and weights are randomly intia-lized every time, convergence may vary . We may try with different iterations like 100, 150, 200 etc. to have better convergence. We may also store the weights of the SOM for which we can achieve better accuracy.

6 Conclusion

This is a proposed framework to handle credit card fraud detection specially in banks and in any business industry. We used various techniques to detect fraud in Credit Card transactions but we have found the results of SOM(Self Organising Map) are the benchmark of our problem. This system is capable of providing most of the essential features required to detect fraudulent and legitimate transactions. As technology changes, it becomes difficult to track the behaviour and pattern of fraudulent transactions. We have just detected the fraudulent activity but we have not prevented. Preventing known and unknown fraud in real time is not easy but it is feasible. The proposed architecture is basically designed to detect credit card fraud in online payments, and emphasis is made to provide a fraud prevention system to verify a transaction as fraudulent or legitimate. For implementation purposes it is assumed that issuer and acquirer bank is connected to each other. If this system is to be implemented in real time scenario then exchange of best practices and raising consumer awareness among people can be very helpful in reducing the losses caused by fraudulent transactions.

Further enhancement can be done by making this system secure with the use of certificates for both merchant and customer and as technology changes new checks can be added to understand the pattern of fraudulent transactions and to alert the respective card holders and bankers when fraud activity is identified.

References

[1] KhyatiChaudhary, JyotiYadav, BhawnaMallick, " A review of Fraud Detection Techniques: Credit Card", International Journal of Computer Applications Volume 45– No.1 2012.
[2] Linda Delamaire, Hussein Abdou, John Pointon, "Credit card fraud and detection techniques: a review", Banks and Bank Systems, Volume 4, Issue 2, 2009.
[3] Kohonen, T. "The self-organizing maps". In Proceedings of the IEEE (1990) 78 (9), pp 1464– 1480.
[4] Vladimir Zaslavsky and Anna Strizhak "Credit card fraud detection using self organizing maps". Information & Security. An International Journal, (2006). Vol.18; (48-63).
[5] Vesanto, J., &Alhoniemi, E. (2000). "Clustering of the self-organizing map".IEEE Transactions on Neural Networks, (2009). 11; (586–600).
[6] Serrano-Cinca, C "Self-organizing neural networks for financial diagnosis". Decision Support Systems, (1996). 17; (227–238).
[7] a. k. s.,. m. abhinav srivastava, "credit card fraud detection using hidden markov model," IEEE, vol. 5, no. 1, 2008.
[8] E. D. Yusuf Sahin, "detecting credit card fraud by ann and logistic regression," 2011.
[9] R. M. jamail esmaily, "Intrusion detection system based on multilayer perceptron neural networks and decision tree," in International conference on Information and Knowledge Technology, 2015.
[10] S. J. K. T. J. C. W. Siddhatha Bhattacharya, "Data Mining for credit card fraud: A comparative study," Elsevire, vol. 50, no. 3, pp. 602613, 2011.
[11] "Raghavendra Patidar and Lokesh Sharma," International Journal of soft computing and engineering, vol. 1, no. NCAI2011, 2011
[12] s. p. tanmay kumar behera, "credit card fraud detection: a hybrid approach using fuzzy clustering and neural network," in international conference on advances in computing and communication Engineering, 2015.
[13] N. W. Wen -Fang Yu, "Research on credit card fraud detection model based on distance sum," in International joint conference on artificial intelligence, Hainan Island,China, 2009.
[14] S. k. A. K. M. Ayushi agarwal, "Credit card fraud detection: A case study," in IEEE, New Delhi, India, 2015.
[15] K. T. B. V. Sam Maes, "Credit cards fraud detection using bayesian and neural networks," p. 7, August 2002.