Bayesian Method with Clustering Algorithm for Credit Card Transaction Fraud Detection

Luis Jose S. Santos (lixsnx@gmail.com)
De La Salle University, Philippines

Shirlee R. Ocampo (shirlee.ocampo@dlsu.edu.ph)
De La Salle University, Manila, Philippines

ABSTRACT

Card transaction fraud is prevalent anywhere. Even with current preventive measures like the Europay, MasterCard and Visa (EMV) chips, possible weaknesses or loopholes can be exploited by fraudsters. This paper explores Naïve Bayes Classifier and clustering algorithms to detect fraud in credit card transactions. Data on fraud labels and arrival times of transactions were simulated by the Markov Modulated Poisson Process. Amounts of transactions for genuine and fraud transactions were simulated based on two Gaussian distributions. Kinds of spenders and types of fraudsters serve as the bases for the parameters used in the simulation of the data. Using the simulated data, EM clustering algorithm with three different initializations and K-means were applied to cluster transaction amounts into high, medium and low. The Naïve Bayes classifier algorithm was then applied to classify the transactions as good or fraud for the simulated data of 9 types of fraudsters across all clustering algorithms. Simulations and analyses were done using R software. Results include comparisons of true positive rates, false positive rates, and detection accuracies among the nine types of fraudsters across all clustering algorithms. For 3 clusters, (high, medium, low transaction amounts), the Naïve Bayes Method with clustering algorithms resulted to an average of 76% true positive (TP) detection, 18% false positive (FP) detection, with an overall accuracy of 81%. The same averages of TP, FP, and overall accuracy were obtained using 2 clusters (high, and low). EM clustering algorithm generated TP, FP, and overall accuracy of 80%, 16%, and 83% respectively.

Keywords: *credit card transaction, fraud detection, Naïve Bayes Classifier, clustering algorithms, Markov Modulated Poisson Process data simulation*

JEL Classification*: C11, C39*

1. INTRODUCTION

E-commerce is basically any form of transaction between two parties electronically without the need for physical contact. A popular medium of payment for such is the use of credit cards. The steady growth of the use of credit cards looks favorable, but there are possible adverse effects when it comes to the security of such cards. Although credit cards possess numerous measures of security to prevent fraudulent use, they are not completely immune to attacks. These attacks can be categorized as either a physical attack on the card or card-holder, or a virtual attack such as phishing. The numerous ways of committing fraud are alarming because large amounts of money were lost by banks. According to Panigrahi, et al (2009), card fraud is still an ongoing problem despite all the security measures available for each card. Fraudsters are adaptive, and given time, will eventually be able to avoid further security, and hence, having a detective measure for fraud is highly recommended to bypass financial losses.

Methods for card fraud detection are mostly based on neural networks, machine learning, and data-mining. For instance, a detection method based on Bayesian Networks (BN) was proposed by Mukhanov (2008) using Naïve Bayesian Classifier. The Naïve Bayes is a form of BN which uses conditional independence of attributes which is better in terms of predictive power as compared to constraint-based and score-based algorithms which focuses on the structure of the Bayesian Network.

A fusion approach including Bayesian learning was proposed as a novel approach to fraud detection (Panigrahi, et al, 2009). This recently devised method uses Dempster-Shafer theory in conjunction with the Bayesian Learner for determining whether a new transaction is genuine or not. The data used was simulated, and the results produced more accurate detections of fraud. Hidden Markov Model (HMM) was used by Srivastava et al (2005). Using only transaction amounts, this method produced accuracy of over 80%.

This paper explores Naïve Bayes Classifier and clustering algorithms to detect fraud in credit card transactions. It aims to introduce a simple fraud detection system that can be used by credit card companies as an initial measure of fraud detection.

Within the fraud detection system (FDS), the objectives include the comparison of the Expectation-Maximization (EM) and K-means clustering algorithms. K-means clustering has been the common choice in profiling the spending behavior of credit card holders. In this study, the EM Algorithm is explored because of its status as model-based clustering algorithm.

2. DATA AND METHODOLOGY

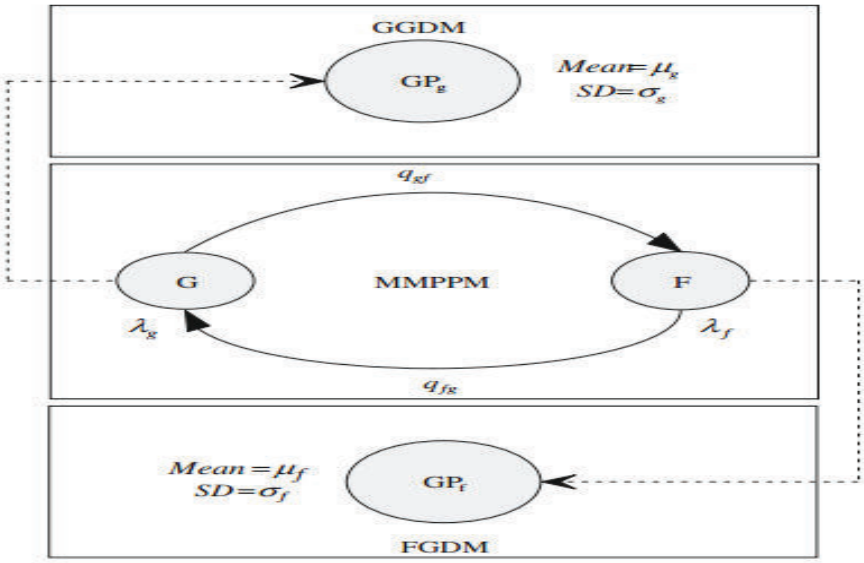
2.1 Data Simulation

Data simulation procedure was adapted from Panigrahi et al (2009). Data simulation is composed of three components: Markov Modulated Poisson Process (MMPP), Genuine Gaussian Distribution Module (GGDM), and Fraud Gaussian Distribution Module (FGDM).

MMPP is a Poisson arrival process that has its parameter λ controlled by an underlying Markov process. It has two states, genuine (g) and fraud (f), with transaction arrival times λ_g and λ_f, respectively. Amounts of transactions for genuine and fraud transactions were simulated based on two Gaussian distributions. GGDM was used to generate transaction amounts for different genuine customers by varying mean μ and standard deviation σ. This is a Gaussian process with mean μ_g and standard deviation σ_g. FDGM was used to generate transaction amounts for fraudsters. This Gaussian process can simulate different categories of fraudsters by varying mean μ_f and standard deviation σ_f.The basic flow of this algorithm is shown in Figure 1. All genuine transactions are initially recorded under a single database but will be classified later based on their fraud labels.

Data Simulation

Figure 1



(Panigrahi, et al, 2009)

Kinds of spenders and types of fraudsters serve as the bases for the simulation of different sets of data using R software. Srivastava, et al (2005) generalized spenders into the usual range of their spending i.e. high spender, low spender. For fraudsters, expert knowledge suggests two kinds of fraudsters. The first kind maximizes their benefit by making several low value purchases with common items. The second kind would do the opposite by making one high value purchase, perhaps with electronics. Although cited in Panigrahi et al (2009), the simulator settings were not described specifically. However, these settings produced varied results among each other. Table 1 summarizes these settings and the parameters are defined accordingly: λ_g and λ_f are Poisson rates for genuine and fraudulent distributions respectively; and q_{gf} and q_{fg} are the transition probabilities from good to fraud and fraud to good respectively. The values of these parameters are to be inputted for the MMPP, implemented using the *HiddenMarkov* R package to simulate data. The means of the two Gaussian distributions, denoted by μ_g and μ_f, are to be used following the MMPP, and are defined to be percentages of an unknown credit limit.

Data Simulator Parameter Settings

Table 1

Settings	λ_g	λ_f	q_{gf}	q_{fg}	μ_g	μ_f
SS1	2	8	0.8	0.2	10	50
SS2	4	8	0.8	0.5	20	50
SS3	2	6	0.5	0.2	10	30
SS4	6	8	0.8	0.8	30	50
SS5	4	6	0.5	0.5	20	30
SS6	2	4	0.2	0.8	10	20
SS7	6	4	0.2	0.8	30	20
SS8	6	6	0.5	0.8	30	30
SS9	4	4	0.2	0.5	10	20

(Panigrahi, et al, 2009)

2.2 Clustering Algorithms

Using the simulated data, Expectation-Maximization (EM) with three initializations and K-means algorithms were applied to cluster transaction amounts into three clusters (high, medium and low) in comparison with two clusters (high and low).

2.2.1. Expectation-Maximization Clustering

Expectation-Maximization (EM) clustering is a type of probabilistic model-based clustering algorithm as described below.

Expectation (E-Step): Construct a conditional distribution Q_i of z_i over x_i, that is,

$$Q_i = p(z_i|x_i;\ \theta) \quad (1)$$

where x_i are *m* observed variables from a data set with *i* = 1, 2, 3, …, *m'*

z_i, are *m* latent random variables obtained from the observed variables x_i, and

θ is the parameter for the conditional distribution Q_i.

Maximization (M-Step): Maximize the expression from the E-Step with respect to its parameters, that is,

$$\theta = arg\ \max_{\theta} \sum_{i} \sum_{z_i} Q_i\ log\ \frac{p(x_i, z_i;\ \theta)}{Q_i} \quad (2)$$

Repeat E and M step until convergence.

Used as a clustering algorithm, EM assumes a finite mixture of Gaussian distributions. This assumption is also used in the *EMCluster* R package used in this research. In this case, Q_i is of the form

$$f(x|\theta) = \sum_{k=1}^{K} \pi_k \cdot\ \varphi(x|\mu_k, \Sigma_k) \quad (3)$$

where x is a p-dimensional observation,

$\theta = \{\pi_1, \pi_2, \dots, \pi_{K-1},\ \mu_1, \mu_2, \dots, \mu_K, \Sigma_1, \Sigma_2, \dots, \Sigma_K\}$,

$\pi_1, \pi_2, \dots, \pi_K$, are the mixing proportions for the *K* clusters such that $\sum_{k=1}^{K} \pi_k = 1$, and $0 < \pi_k < 1\ \forall\ k = 1, 2, \dots, K$, and

$\varphi(x|\mu_k, \Sigma_k)$ are multivariate Gaussian distributions with mean vector μ_K and variance-covariance matrix Σ_K.

The data is then partitioned into K clusters by the maximum posterior

$$\arg\max_{k} \hat{\pi}_k \cdot\ \varphi(x_i|\hat{\mu}_k, \hat{\Sigma}_k) \quad (4)$$

Although the EM clustering algorithm is efficient, its effectivity is largely dependent on its starting point; especially at a high number of clusters (Hu, 2015). This paper explored 3 initialization methods available in the R program, namely, *RndEM*, *emEM,* and *svd*.

The three initializations found in the *EMCluste*r package guide of the R program are briefly described as follows. The first initialization, *RndEM*, selects *K* centers from the data, and the other data are grouped to the closest

center based on Euclidean distance. The initial with the highest log likelihood is chosen as the initial value for the EM algorithm until convergence. The second initialization, *emEM*, also starts by selecting *K* centers, but this time directly applying a shorter version of the EM algorithm. Once the algorithm converges to a specified value, that value is used as the start of the regular EM algorithm. The third initialization singular value decomposition *svd* selects centers from a major component space. The data are then grouped around the centers by K-means, which also generates the initial. This initial is then used for the EM algorithm.

2.2.2. K-means Clustering

K-means clustering is a type of partition-based clustering, which aims to partition the points into *k* groups such that the sum of squares from points to the assigned cluster centers are minimized. The Within Cluster Sum of Squares (WCSS) used in *kmeans()* function in R is given by

$$WCSS = \sum_{i=1}^{n}\sum_{j=1}^{p}\left(x_{ij} - \overline{x_{kj}}\right)^2 \tag{5}$$

where x_{ij} is the ij[th] object,
n is the number of objects, *i* = 1, 2, …, *n,*
p is the number of variables from *j* = 1, 2, …, *p*, and
$\overline{x_{kj}}$ is the mean variable j of all elements in group *k*.

The algorithm of this K-means clustering is as follows:

1. Allocate objects to clusters with the nearest cluster mean.
2. Search for the *K*-partition with locally optimal WCSS by moving objects from one cluster to another.
3. Repeat step 2 until convergence.

Each clustering algorithm was scored using the silhouette coefficient (SC) given by

$$s = \frac{b - a}{max(a, b)} \tag{6}$$

where a is a mean distance between a sample and all other points in the same class, and

b is a mean distance between a sample and all other points in the next nearest cluster.

High values of SC (closer to 1) indicate highly dense clustering. Values less than zero would indicate either overlapping clusters or incorrect clustering. All outcomes of these clustering algorithms were inputted to Naïve Bayes algorithms.

2.3 Naïve Bayes Classifier

The Naïve Bayes classifier is a set of learning algorithms based on applying the Bayes' rule given by

$$P(B_r|A) = \frac{P(B_r \cap A)}{P(A)} \tag{7}$$

$$= \frac{P(B_r)P(A|B_r)}{P(A)} \tag{8}$$

where A and B_i are events from a sample space, $i = 1, 2, \dots, r, \dots, k$ and

$P(A) = \sum_{i=1}^{k} P(B_i)P(A|B_i)$, following the rule of total probability

The Naïve Bayes' Classifier has the 'naïve' assumption of independence between pair of attributes and can be derived as follows:
Let A be any event, and B_i independent attributes of event A. Construct a conditional likelihood table as shown in Table 2.

Conditional Likelihood Table

Table 2

	B_1		B_2		...	B_i		Total
A	$P(B_1\|A)$	$P(\neg B_1\|A)$	$P(B_2\|A)$	$P(\neg B_2\|A)$	...	$P(B_i\|A)$	$P(\neg B_i\|A)$	$P(A)$
$\neg A$	$P(B_1\|\neg A)$	$P(\neg B_1\|\neg A)$	$P(B_2\|\neg A)$	$P(\neg B_2\|\neg A)$	...	$P(B_i\|\neg A)$	$P(\neg B_i\|\neg A)$	$P(\neg A)$

Consider computing for $P(A|B_1 \cap \neg B_2 \cap \neg B_3 \cap B_4)$. Applying Bayes' rule from eqn [5],

$$P(A|B_1 \cap \neg B_2 \cap \neg B_3 \cap B_4) = \frac{P(B_1 \cap \neg B_2 \cap \neg B_3 \cap B_4|A)P(A)}{P(B_1 \cap \neg B_2 \cap \neg B_3 \cap B_4)} \tag{9}$$

This becomes more and more computationally difficult as the number of B_i's increase. Therefore, the naïve independence assumption of Naïve Bayes is applied resulting to

$$P(A|B_1 \cap \neg B_2 \cap \neg B_3 \cap B_4) = \frac{P(B_1|A)P(\neg B_2|A)P(\neg B_3|A)P(B_4|A)P(A)}{P(B_1)P(\neg B_2)P(\neg B_3)P(B_4)} \quad (11)$$

This can be generalized into

$$P(A_L|B_1, \ldots, B_N) = \frac{1}{Z}P(A_L)\prod_{i=1}^{n} P(B_i|A_L) \quad (12)$$

where A_L is a class A at level L,
B_i, are the N features of the classes, $i = 1, 2, \ldots, N$, and
$Z = \prod_{i=1}^{n} P(B_i)$ is the scaling factor.

The distribution of $P(B_i|A_L)$ is not specified, and as such can take on different distributions. For this paper, $P(B_i|A_L)$ assumes a multinomial distribution since the features of the Naïve Bayes Classifier will have two levels (for the dummy binary variables, and clustering for K = 2), and three levels (for clustering K = 3. This incarnation of the Naïve Bayes classifier is called the Multinomial Naïve Bayes.

Note that the conditional likelihood table is in the form of a probability mass function (PMF), and the expression $\prod_{i=1}^{n} P(B_i|A_L)$ from eqn [11], the product of all conditional probabilities dependent on A_L, can be treated as a likelihood function. Likewise, the expression $\frac{1}{Z}P(A_L)$ also from eqn [11] can be treated as a prior probability. Multiplying those two expressions results to a posterior probability following the Bayesian Method.

2.4. Proposed Fraud Detection System

The typical flow of transaction fraud detection systems (FDS) starts with the classification of transaction amounts. This is to acquire the spending behavior of the credit card holder. After which, the data acquired from the clustering is incorporated into whatever model is proposed in order to detect fraud. The proposed FDS is derived from Srivastava et al (2008), Mukahnov (2008), and Panigrahi et al (2009). It is described as follows.

1. Record past transactions classified separately as genuine or fraud along with transaction amounts (m), and arrival time of transaction (a). This is called the initial transaction history.
2. Classify transactions separately into either genuine or fraud.
3. Compute the interarrival times (i) from the arrival times for all genuine transactions using $i = a_t - a_{t-1}$
4. Create four dummy binary variables (T_1, T_2, T_3, T_4) based on the following rules:

$$T_1 = \begin{cases} 1 & if\ i < 0.333 \\ 0 & otherwise \end{cases}$$

$$T_2 = \begin{cases} 1 & if\ 0.333 < i < 0.667 \\ 0 & otherwise \end{cases}$$

$$T_3 = \begin{cases} 1 & if\ 0.667 < i < 1 \\ 0 & otherwise \end{cases}$$

$$T_4 = \begin{cases} 1 & if\ i > 1 \\ 0 & otherwise \end{cases}$$

5. Cluster transaction amounts (m) into K clusters ($K = 2$ or $K = 3$).
6. Repeat steps 3 to 5 for fraudulent transactions.
7. Include the created variables into the initial transaction history and call this the secondary transaction history.
8. Apply the Naïve Bayes algorithm to the secondary transaction history to classify new transactions as either genuine or fraudulent.

The created dummy variables T_{ij} and a set of clustered transaction amounts m_i were used along with the fraud labels to train the Naïve Bayes, implemented by the *bnlearn* package. Note that T_{ij} are the created dummy binary variables in the proposed FDS, $i = 1,2,3,4,\ j = 0,1$, and m_i refers to the clustered transaction amounts, $i = 1,2,3$ for 3 clusters, $i = 1,2$ for 2 clusters. Naïve Bayes conditional likelihood is of the form shown in Tables 3 and 4 for clusters 3 and 2, respectively.

Naïve Bayes Likelihood Table for $K = 3$ Clusters

Table 3

	T_{1j}		T_{2j}		T_{3j}		T_{4j}		m_i		
	Yes (1)	No (0)	Yes (1)	No (0)	Yes (1)	No (0)	Yes (1)	No (0)	Low (1)	Medium (2)	High (3)
Fraud (F)	$P(T_{11}\|F)$	$P(T_{10}\|F)$	$P(T_{21}\|F)$	$P(T_{20}\|F)$	$P(T_{31}\|F)$	$P(T_{30}\|F)$	$P(T_{41}\|F)$	$P(T_{40}\|F)$	$P(m_1\|F)$	$P(m_2\|F)$	$P(m_3\|F)$
Genuine (¬F)	$P(T_{11}\|\neg F)$	$P(T_{10}\|\neg F)$	$P(T_{21}\|\neg F)$	$P(T_{20}\|\neg F)$	$P(T_{31}\|\neg F)$	$P(T_{30}\|\neg F)$	$P(T_{41}\|\neg F)$	$P(T_{40}\|\neg F)$	$P(m_1\|\neg F)$	$P(m_2\|\neg F)$	$P(m_3\|\neg F)$

Naïve Bayes Likelihood Table for $K = 2$ Clusters

Table 4

	T_{1j}		T_{2j}		T_{3j}		T_{4j}		m_i	
	Yes (1)	No (0)	Yes (1)	No (0)	Yes (1)	No (0)	Yes (1)	No (0)	Low(1)	High (2)
Fraud (F)	$P(T_{11}\|F)$	$P(T_{10}\|F)$	$P(T_{21}\|F)$	$P(T_{20}\|F)$	$P(T_{31}\|F)$	$P(T_{30}\|F)$	$P(T_{41}\|F)$	$P(T_{40}\|F)$	$P(m_1\|F)$	$P(m_2\|F)$
Genuine (¬F)	$P(T_{11}\|\neg F)$	$P(T_{10}\|\neg F)$	$P(T_{21}\|\neg F)$	$P(T_{20}\|\neg F)$	$P(T_{31}\|\neg F)$	$P(T_{30}\|\neg F)$	$P(T_{41}\|\neg F)$	$P(T_{40}\|\neg F)$	$P(m_1\|\neg F)$	$P(m_2\|\neg F)$

For evaluating the performance of the FDS, three common metrics were taken. The true positive rate (TPR), false positive rate (FPR), and the overall accuracy are often used in fraud detection studies, and these were used by Panigrahi et al (2009), Srivastava et al (2008), and Maes et al (1993). A good fraud detection system should typically have high TPR and overall accuracy, and low FPR. They are computed as follows:

$$TPR = \frac{Number\ of\ True\ Positive\ (TP)\ Cases}{Total\ Number\ of\ Positive\ Cases} \quad (12)$$

$$FPR = \frac{Number\ of\ False\ Positive\ (FP)\ Cases}{Total\ Number\ of\ Negative\ Cases} \quad (13)$$

$$Overall\ Accuracy = \frac{Number\ of\ TP\ Cases + Number\ of\ FP\ Cases}{Total\ Number\ of\ Cases} \quad (14)$$

Note that in this case, positive will refer to fraud transactions, and negative will refer to genuine transactions

3. RESULTS AND DISCUSSION

Nine initial transaction histories were generated, each containing 300 transactions with different ratios of fraud. Of the nine data sets generated, the highest amount of fraudulent transactions amounted to over ninety percent of transactions (SS6). The lowest prevalence of fraud is from SS1 followed by SS2. A summary of the simulated data sets is shown in Table 5.

Summary of Simulated Data Sets

Table 5

	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9
Genuine	282 94.0%	235 78.3%	225 75.0%	109 36.3%	136 45.3%	28 9.3%	147 49.3%	108 36.0%	110 36.7%
Fraud	18 6.0%	65 21.7%	75 25.0%	191 63.7%	164 54.7%	272 90.7%	153 51.0%	192 64.0%	190 63.3%

The EM clustering algorithm and its initializations produced results close to each other, while the K-means algorithm result was somewhat different. Distributions of cases falling in both 3 clusters and 2 clusters for all simulated data sets were generated along with corresponding Silhouette Coefficients (SC). On the average, $K = 2$ clusters exhibited denser clusters compared to $K = 3$ clusters. Different clustering algorithms produced different results across the nine parameter settings. The highest SC belongs to initialization *emEM* of the EM algorithm at both $K = 3$ and $K = 2$, with values

of 0.830851 and 0.508284, respectively. K-means consistently produced low values of SC, all with values below 0.3. A summary of these coefficients is presented in Tables 6 and 7.

Silhouette Coefficient for K = 3 Clusters

Table 6

	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9
RndEM	-0.0322	0.0721	0.3394	-0.0337	0.0074	0.1052	0.4822	0.1146	0.2834
emEM	-0.0162	0.0935	-0.084	-0.0726	0.2602	0.4524	-0.0327	0.5082	-0.0304
svd	-0.1409	0.7778	0.2569	0.214	0.1646	0.1613	0.3681	0.132	0.2755
K-means	0.0622	0.0283	-0.0479	-0.032	-0.0029	0.1604	0.01665	0.2247	0.0875
average	-0.0318	0.2429	0.116	0.0189	0.1073	0.2199	0.2085	0.2449	0.1539

Silhouette Coefficient for K = 2 Clusters

Table 7

	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9
RndEM	0.2222	0.6422	0.2222	0.5586	0.2420	0.1724	0.3184	0.6323	0.0039
emEM	0.2122	0.6422	0.8308	0.7714	0.1374	0.4125	0.3115	0.6376	0.0039
svd	0.2301	0.6298	0.1980	0.3103	0.7262	0.5662	0.7514	0.6443	0.1659
K-means	0.1386	0.0663	0.0759	0.0444	0.0761	0.3037	0.0604	0.5534	0.0413
Average	0.2008	0.4951	0.3317	0.4212	0.2954	0.3637	0.3604	0.6169	0.0538

Naïve Bayes was then applied to the different simulated data sets with the integrated clusters from EM and K-means algorithms. The performance metrics TPR, FPR and overall accuracy were computed. The overall accuracies of the nine simulated data sets with 3 clusters in Table 8 show above 90% in SS1 and SS6. With accuracies of above 80% are those of SS2 and SS3, while the rest register an accuracy of above 70%.

Naïve Bayes Overall Accuracy for *K* = 3 Clusters

Table 8

Algorithm	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9
emEM	95.3%	89.0%	75.7%	64.0%	92.3%	96.3%	58.7%	65.0%	88.7%
RndEM	94.7%	88.0%	96.0%	64.0%	61.0%	94.0%	93.7%	65.0%	71.7%
Svd	94.3%	98.0%	88.3%	88.3%	79.0%	93.3%	90.3%	88.0%	90.7%
K-means	91.7%	78.0%	74.7%	65.3%	66.0%	90.7%	59.3%	67.7%	63.3%
Average	94.0%	88.3%	83.7%	70.2%	74.6%	93.6%	75.5%	71.4%	78.6%

For Naïve Bayes with $K = 2$ clusters, the results do not differ from the accuracies of the case $K = 3$ clusters. SS4 has a significant increase in the average accuracy of 8%, with its accuracy registering at 78%. Parameter settings SS5, SS7, and SS8 had the opposite outcome and produced lower overall accuracy than its counterpart. Settings 7 and 8 had over 7% decrease in accuracy,

while setting 5 had a 5% decrease in accuracy. Table 9 contains the summary of the overall accuracies of the nine simulated data sets with 2 clusters.

Naïve Bayes Overall Accuracy for $K = 2$ Clusters

Table 9

Algorithm	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9
emEM	94.7%	92.7%	96.3%	94.7%	61.0%	93.7%	60.7%	63.7%	83.7%
RndEM	95.0%	92.7%	75.7%	82.3%	62.0%	91.0%	61.3%	64.7%	80.1%
Svd	94.3%	92.3%	76.7%	71.0%	91.3%	92.0%	93.3%	64.7%	84.0%
K-means	92.7%	78.0%	75.7%	64.0%	60.7%	89.3%	57.3%	64.7%	71.7%
Average	94.2%	88.9%	81.1%	78.0%	68.8%	91.5%	68.2%	64.4%	79.9%

While results across the simulated data sets are varied, most of the overall accuracies of the fraud detection system are higher than 70%. For $K = 3$ clusters, two initializations of the EM algorithm registered an accuracy of 80% (*emEM* and *RndEM*), with another initialization capping at 90% (*svd).* At $K = 2$, initialization *RndEM* from EM clustering, and K-means made virtually similar results, while the highest accuracies belonged to *emEM* and *svd.* The results of the three initializations of EM Algorithm across the settings are more alike.

Both the true positive rate (TPR) and the false positive rate (FPR) also registered higher at the EM algorithm initializations, with *RndEM* scoring both the highest TPR and the lowest FPR among all the clustering algorithms. K-means clustering algorithm scored 72.96% at the overall accuracy with TPR and FPR of 64% and 26%, respectively. The case for high and low clusters produced different results, with *emEM* having the highest TPR. The two other initializations of EM obtained the lowest FPR. Furthermore, the averages of the initializations indicate that EM is superior to K-means in accuracy, as well as TPR and FPR for 2 and 3 clusters. Table 10 summarizes the TPR and FPR of the fraud detection system across all the clustering algorithms and initializations used. On the average, EM clustering algorithm has an overall accuracy of 83.83% with TPR of 80.14% and FPR of 15. 34%. Figures 2 and 3 are visual supplements to Tables 10 and 11.

Algorithm and Initialization Accuracy for $K = 3$ Clusters

Table 10

	emEM	*RndEM*	*svd*	K-means
TPR	73.80%	**85.71%**	80.93%	64.52%
FPR	18.35%	**12.10%**	15.56%	26.34%
Accuracy	80.56%	80.89%	**90.04%**	72.96%

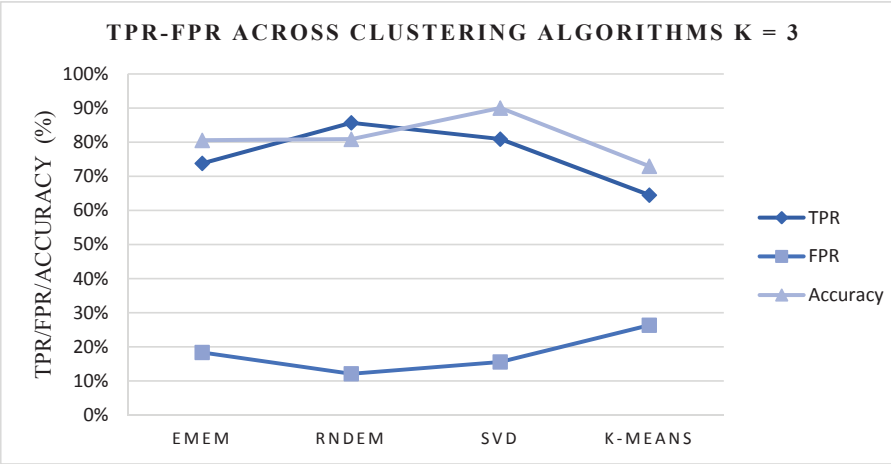
Algorithm and Initialization Accuracy for *K* = 2 Clusters

Table 11

	emEM	*RndEM*	*svd*	K-means
TPR	**81.07%**	75.91%	79.35%	65.75%
FPR	19.36%	**15.46%**	**15.12%**	23.64%
Accuracy	**83.70%**	80.07%	**83.96%**	71.74%

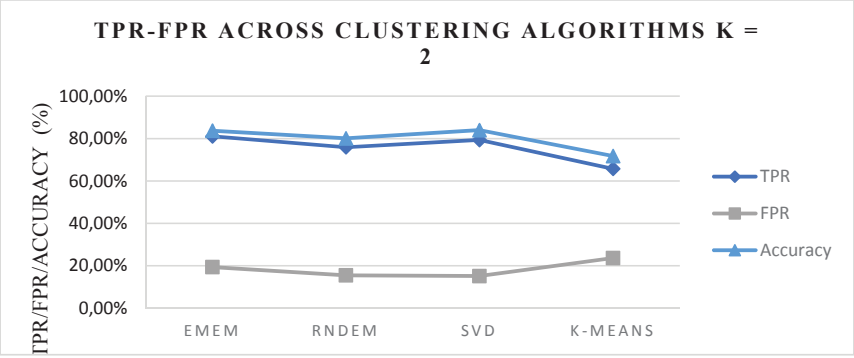
Over-all TPR-FPR across the Clustering Algorithms for K = 3 Clusters

Figure 2



Over-all TPR-FPR across the Clustering Algorithms for K = 2 Clus-ters

Figure 3

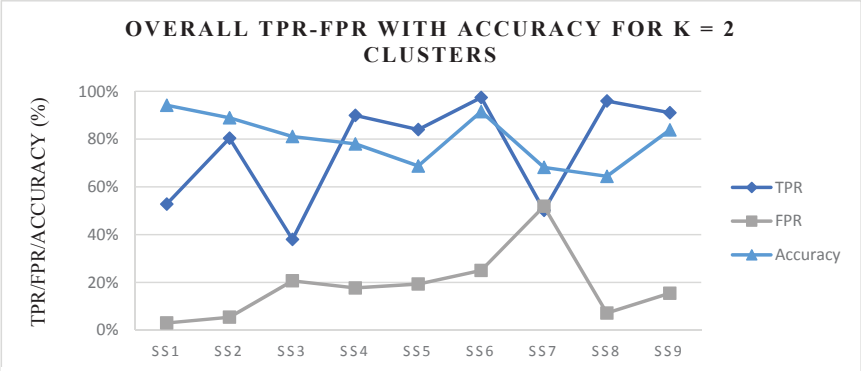


The spread of TPR and FPR with overall accuracy across the 9 simulated data sets are shown in Figures 4 and 5. SS6 got the highest TPR, and

overall accuracy, while SS3 has the lowest TPR for $K = 3$. At $K = 2$, SS6 had the highest overall accuracy, and SS7 seems to have performed worst with both the highest FPR, as well as the second lowest TPR.

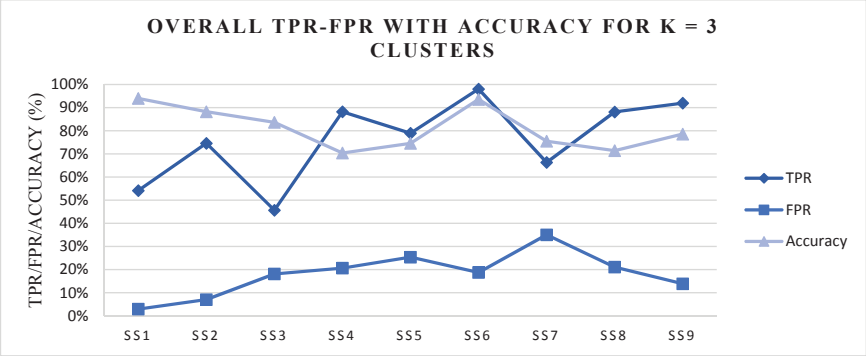
Over-all TPR-FPR across the Simulated Data for K = 2 Clusters

Figure 4



Over-all TPR-FPR across the Simulated Data for K = 3 Clusters

Figure 5



5. CONCLUSIONS AND RECOMMENDATIONS

The proposed Naïve Bayes method with clustering is generally effective with an overall accuracy of 81% with 3 clusters. At 2 clusters, the overall accuracy does not deviate much with an average accuracy of 79%. The true positive rate of this method averages at 80% and can be further improved. Both cases of clusters produced the same averages. Results also show that the

accuracy is dependent on the type of fraudsters where fraudsters with large transactions are easier to detect with high TPR and overall accuracy.

Among the three performance metrics, the EM clustering algorithm is better than the K-means algorithm. Increasing the number of clusters is recommended that may help determine the best initialization for the EM clustering algorithm, and profile the spending patterns of both credit card holders and fraudsters. More specifically, use $K = 5$ for, very low, low, medium, high, and very high transaction amounts. It is also recommended to test this method with other clustering algorithms to achieve better silhouette coefficients.

For the simulator, it is suggested that the mean of the Gaussian Distributions should be turned into an actual amount rather than a percentage of a credit limit. Since the credit limit should be unknown to the fraudster, it is less realistic to include the credit limit in the simulator. Furthermore, the proposed FDS algorithm could also be revised by combining it with other Bayesian Network methods. Integration with other fraud detection methods can also be explored. It is highly recommended to apply the proposed method to real-life data if available.

References

1. **Barlongo, C.**, 2015, *"CCAP expects double-digit growth of credit-card holders in next few years"*, Business Mirror. March 22, 2015. Retrieved from http://www.businessmirror.com.ph/ccap-expects-double-digit-growth-of-credit-card-holders-in-next-few-years/, February 2017.
2. **Hu, Z.**, 2015, *"Initializing the EM Algorithm for Data Clustering and Sub-population Detection"*. (Electronic Dissertation). The Ohio State University, USA. Retrieved from https://etd.ohiolink.edu/, June 2017.
3. **Lantz, B.,** 2013, *Machine learning with R*. Birmingham: PACKT Publishing.
4. **Maes, S., Tuyls, K., Vanschoenwinkel, B. & Manderick, B.**, 1993, *"Credit Card Fraud Detection Using Bayesian and Neural Networks"*. Retrieved from https://www.researchgate.net/profile/Karl_Tuyls/publication/248809471_Credit_Card_Fraud_Detection_Applying_Bayesian_and_Neural_neworks/links/0deec52519708c5f7a000000 .pdf, February 2017.
5. **Maitra, R.**, 2001, *"Clustering Massive Datasets with Applications in Software Metrics and Tomography"*, Technometrics, 43(3). Retrieved from http://www.jstor.org/stable/1271221, June 2017.
6. **Mukhanov, L.**, 2008, *"Using Bayesian Belief Networks for Credit Card fraud Detection"*, Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications, February 11-13, 2008, Innsbruck, Austria, ACTA Press Anaheim, CA, USA, 221-225. https://www.researchgate.net/publication/262175453_Using_Bayesian_Belief_Networks_for_credit_card_fraud_detection, June 2017.
7. **Panigrahi, S., Kundu, A., Sural, S., & Majumdar, A**., 2009, *"Credit card fraud detection: A Fusion Approach Using Dempster–Shafer theory and Bayesian Learning"*, Information Fusion, 10(4), 354-363. doi:10.1016/j.inffus.2008.04.001, February 2017.
8. **Srivastava, A., Kundu, A., & Sural, S**., 2005, *"Credit Card Fraud Detection Using Hidden Markov Model"*, IEEE Transactions on Dependable and Secure Computing, 5(1), 37-48. doi:10.1109/TDSC.2007.70228, February 2017.

Appendix

R codes:

```
#Simulation:
#package Hidden Markov, xlsx
Q <- matrix(c(-.2, .2, .8, -.8), byrow=TRUE, nrow=2)
x <- mmpp(NULL, Q, delta=c(.99, 0.01), lambda=c(8, 2))
y <- simulate(x, nsim=299, seed = 10)
at <- y$tau #arrival times
lab <- y$ys
m <- list()
for (i in 1:length(lab)){
m[i] = (lab[i] - 1)
}
label <- unlist(m) #fraud labels (0, 1)
set.seed(123)
n <-list()
for (i in 1:length(label)){
if (label[i] == 0){n[i] = rnorm(1, 10, )}
else if (label[i] == 1){ n[i] = rnorm(1, 50, )}
else {NULL}
}
amt <- unlist(n)
out<- data.frame(at, amt, label)
colnames(out) <- c("arrival.times", "amount", "fraud.label")
write.xlsx(out, "SS1.xlsx")

#Stage 1:
ml <- read.xlsx("SS1.xlsx", sheetName = "Sheet1", header = TRUE)
ch <- ml[ml$fraud.label == 0, ]
at1 <- ch$arrival.times
l1 <- list(0)
for (i in 2:length(at1)){
l1[i] = at1[i] - at1[i-1]
}
iat.ch <- unlist(l1)
ch$interarrival.times <- iat.ch
ch <- ch[, c(1, 2, 5, 3, 4)]
d1 <- list()
d2 <- list()
d3 <- list()
d4 <- list()
for (i in 1:nrow (ch)){
if(iat.ch[i]<=.33){
d1[i] = 1
} else{d1[i] = 0}
}
for (i in 1:nrow (ch)){
if((iat.ch[i] > .33)&(iat.ch[i]<=.66)){
d2[i] = 1
} else{d2[i] = 0}
}
for (i in 1:nrow (ch)){
if((iat.ch[i] > .66)&(iat.ch[i]<=1)){
d3[i] = 1
} else{d3[i] = 0}
}
for (i in 1:nrow (ch)){
if(iat.ch[i] > 1){
d4[i] = 1
} else{d4[i] = 0}
}
D1 <- unlist(d1)
D2 <- unlist(d2)
D3 <- unlist(d3)
D4 <- unlist(d4)
ch$D1<- D1
ch$D2<- D2
ch$D3<- D3
ch$D4<- D4
m1 <- data.frame(ch$amount)

amt.rnd <- init.EM(m1, nclass = 3, method = "em.EM")
amt.em <- init.EM(m1, nclass = 3, method = "Rnd.EM", EMC = .EMC.Rnd)
amt.svd <- emgroup(m1, nclass = 3)
amt.kmn <- kmeans(m1, 3, iter.max = 10, nstart = 1)
cl1 <- amt.rnd$class
cl2 <- amt.em$class
cl3 <- amt.svd$class
cl4 <- amt.kmn$cluster
ch$cl1 <- cl1
ch$cl2 <- cl2
ch$cl3 <- cl3
ch$cl4 <- cl4
ch$NA. <- NULL


fh <- ml[ml$fraud.label == 1, ]
at2 <- fh$arrival.times
l2 <- list(0)
for (i in 2:length(at2)){
l2[i] = at2[i] - at2[i-1]
}
iat.fh <- unlist(l2)
fh$interarrival.times <- iat.fh
fh <- fh[, c(1, 2, 5, 3, 4)]
d1 <- list()
```

```
d2 <- list()
d3 <- list()
d4 <- list()
for (i in 1:nrow (fh)){
if(iat.fh[i]<=.33){
d1[i] = 1
} else{d1[i] = 0}
}
for (i in 1:nrow (fh)){
if((iat.fh[i] > .33)&(iat.fh[i]<=.66)){
d2[i] = 1
} else{d2[i] = 0}
}
for (i in 1:nrow (fh)){
if((iat.fh[i] > .66)&(iat.fh[i]<=1)){
d3[i] = 1
} else{d3[i] = 0}
}
for (i in 1:nrow (fh)){
if(iat.fh[i] > 1){
d4[i] = 1
} else{d4[i] = 0}
}
D1 <- unlist(d1)
D2 <- unlist(d2)
D3 <- unlist(d3)
D4 <- unlist(d4)
fh$D1<- D1
fh$D2<- D2
fh$D3<- D3
fh$D4<- D4
m2 <- data.frame(fh$amount)
amt.rnd <- init.EM(m2, nclass = 3, method =
"em.EM")
amt.em <- init.EM(m2, nclass = 3, method =
"Rnd.EM", EMC = .EMC.Rnd)
amt.svd <- emgroup(m2, nclass = 3)
amt.kmn <- kmeans(m2, 3, iter.max = 10,
nstart = 1)
cl1 <- amt.rnd$class
cl2 <- amt.em$class
cl3 <- amt.svd$class
cl4 <- amt.kmn$cluster
fh$cl1 <- cl1
fh$cl2 <- cl2
fh$cl3 <- cl3
fh$cl4 <- cl4
fh$NA. <- NULL

new <- rbind(ch, fh)
mlnew <- new[order(new$arrival.times), ]

write.xlsx(mlnew, "SS1_sim.xlsx", sheet-
Name ="ML", append = FALSE, row.names =
FALSE)

write.xlsx(ch, "SS1_sim.xlsx", sheetName
="CH", append = TRUE, row.names = FALSE)
write.xlsx(fh, "SS1_sim.xlsx", sheetName
="FH", append = TRUE, row.names = FALSE)

#cl1: RndEM
#cl2: emEM
#cl3: svd
#cl4: k-means; Hartigan-Wong

#Performance metrics:

ml <- read.xlsx("SS1_sim.xlsx", sheetName
= "ML", header = TRUE)
intCriteria(as.matrix(ml$amount), as.inte-
ger(ml$cl1), c("cal", "sil"))
intCriteria(as.matrix(ml$amount), as.inte-
ger(ml$cl2), c("cal", "sil"))
intCriteria(as.matrix(ml$amount), as.inte-
ger(ml$cl3), c("cal", "sil"))
intCriteria(as.matrix(ml$amount), as.inte-
ger(ml$cl4), c("cal", "sil"))

RRand(ml$cl1, ml$cl2)
RRand(ml$cl1, ml$cl3)
RRand(ml$cl1, ml$cl4)
RRand(ml$cl2, ml$cl3)
RRand(ml$cl2, ml$cl4)
RRand(ml$cl3, ml$cl4)

#Stage 2:
#EMCluster
ml <- read.xlsx("SS1_sim.xlsx", sheetName
= "ML", header = TRUE)
D1 <- factor(ml$D1)
D2 <- factor(ml$D2)
D3 <- factor(ml$D3)
D4 <- factor(ml$D4)
fraud.label <- factor(ml$fraud.label)
cl1 <- factor(ml$cl1)
cl2 <- factor(ml$cl2)
cl3 <- factor(ml$cl3)
cl4 <- factor(ml$cl4)

mat1 <- data.frame(fraud.label, D1, D2, D3,
D4, cl1)
mat2 <- data.frame(fraud.label, D1, D2, D3,
D4, cl2)
mat3 <- data.frame(fraud.label, D1, D2, D3,
D4, cl3)
mat4 <- data.frame(fraud.label, D1, D2, D3,
D4, cl4)

vars1<- c("D1", "D2", "D3", "D4", "cl1")
vars2<- c("D1", "D2", "D3", "D4", "cl2")
```

```
vars3<- c("D1", "D2", "D3", "D4", "cl3")
vars4<- c("D1", "D2", "D3", "D4", "cl4")

nb1 <- naive.bayes(mat1, "fraud.label", vars1)
nb2 <- naive.bayes(mat2, "fraud.label", vars2)
nb3 <- naive.bayes(mat3, "fraud.label", vars3)
nb4 <- naive.bayes(mat4, "fraud.label", vars4)

pred1 <- predict(nb1, mat1, prob = TRUE)
pred2 <- predict(nb2, mat2, prob = TRUE)
pred3 <- predict(nb3, mat3, prob = TRUE)
pred4 <- predict(nb4, mat4, prob = TRUE)

pscore1 <- t(data.frame(attr(pred1, 'prob')))
rownames(pscore1) <- NULL
pscore1 <- unlist(pscore1) # pscore1[, 1] score of 0, pscore1[, 2] score of 1

pscore2 <- t(data.frame(attr(pred2, 'prob')))
rownames(pscore2) <- NULL
pscore2 <- unlist(pscore2)

pscore3 <- t(data.frame(attr(pred3, 'prob')))
rownames(pscore3) <- NULL
pscore3 <- unlist(pscore3)

pscore4 <- t(data.frame(attr(pred4, 'prob')))
rownames(pscore4) <- NULL
pscore4 <- unlist(pscore4)

pscore10 <- pscore1[, 1]
pscore11 <- pscore1[, 2]
pscore20 <- pscore2[, 1]
pscore21 <- pscore2[, 2]
pscore30 <- pscore3[, 1]
pscore31 <- pscore3[, 2]
pscore40 <- pscore4[, 1]
pscore41 <- pscore4[, 2]

mat1$predicted <- pred1
mat1$pscore0 <- pscore10
mat1$pscore1 <- pscore11
mat2$ predicted <- pred2
mat2$pscore0 <- pscore20
mat2$pscore1 <- pscore21
mat3$predicted <- pred3
mat3$pscore0 <- pscore30
mat3$pscore1 <- pscore31
mat4$ predicted <- pred4
mat4$pscore0 <- pscore40
mat4$pscore1 <- pscore41

write.xlsx(mat1, "RndBN.xlsx", row.names = FALSE)
write.xlsx(mat2, "emBN.xlsx", row.names = FALSE)
write.xlsx(mat3, "svdBN.xlsx", row.names = FALSE)
write.xlsx(mat4, "kmnBN.xlsx", row.names = FALSE)


#BN Accuracy and Performance

pred1 <- read.xlsx("RndBN.xlsx", sheetName = "Sheet1", header = TRUE)
pred2 <- read.xlsx("emBN.xlsx", sheetName = "Sheet1", header = TRUE)
pred3 <- read.xlsx("svdBN.xlsx", sheetName = "Sheet1", header = TRUE)
pred4 <- read.xlsx("kmnBN.xlsx", sheetName = "Sheet1", header = TRUE)

mod1 <- prediction(pred1$pscore1, pred1$fraud.label)
mod2 <- prediction(pred2$pscore1, pred2$fraud.label)
mod3 <- prediction(pred3$pscore1, pred3$fraud.label)
mod4 <- prediction(pred4$pscore1, pred4$fraud.label)

jpeg('ROCRnd.jpeg')
plot(performance(mod1, 'tpr', 'fpr'))
dev.off()

jpeg('ROCem.jpeg')
plot(performance(mod2, 'tpr', 'fpr'))
dev.off()

jpeg('ROCsvd.jpeg')
plot(performance(mod3, 'tpr', 'fpr'))
dev.off()

jpeg('ROCkmn.jpeg')
plot(performance(mod4, 'tpr', 'fpr'))
dev.off()
```