Integrating Classification with K-means to Detect E-commerce Transaction Anomaly

XING TAN

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ARTS

Graduate Program in
Information System and Technology
York University
Toronto, Ontario

September 2015

Abstract

Effective data mining solutions have been anticipated in Electronic Commerce (E-Commerce) transaction anomaly detection model to accurately predict anomaly transaction records. However, there are many sub-optimal E-Commerce transaction anomaly detection models due to highly imbalanced data set. This thesis proposes a meta-cluster with K-means algorithm to solve the problem of highly imbalanced data. This meta-cluster with K-means algorithm will be applied as a preprocessing method. The main aim is to generate a collection of clusters from the E-commerce transaction anomaly data set, each of which contains similar instances. The Logistic Regression, Naive Bayes, RBFNetwork and NBtree classifiers will be applied to evaluate the generated clusters. Results indicate that the proposed method can be easily realized and achieve excellent performance. The most important is that the proposed method can deal with the imbalanced data sets well and minimize type-II errors.

Acknowledgements

First and foremost, I would like to express my greatest appreciation to my supervisor Dr. Zijiang Yang, for her patient guidance, enthusiastic encouragement and constructive suggestions during the planning and development of this thesis. Her extensive knowledge and professionalism had positively influenced my attitude on achieving academic excellence. She had set up a role model for us as a devoted scholar in our future career and academic endeavor.

I would like to extend my gratitude to all faculty members who had offered support and assistance to my study. Special thanks go to Prof. Younes Benslimane, who kindly offered to review my thesis proposal and this thesis. He gave me many valuable advices on how to improve my thesis.

My sincere thanks should go to my parents, who loved me unconditionally and encouraged me with their best wishes. Many thanks also go to Dr. Wenjie You, who offered insight and gave me another prospective on my research topic by working together for the recommendation system project of TruCentric Company. Special thanks go to Mr. Yan Tu, who helped me modify some grammar and typo errors of my thesis. And I would like to thank my dear friend Chenrui Xiong. It would have been a lonely lab without him.

Last but not least, I would like to give my sincerest thanks to my wife, Xi Wang. She was always there standing by my side and supporting me with love and faith in both myself and my academic pursuit, without which I would never have been able to finish my graduate study.

Table of Contents

List of Tables

List of Figures

1. Introduction

Electronic Commerce (E-Commerce) is growing rapidly in worldwide marketplace nowadays and it has tremendous potential to drive the technology growth in the future. Along with the continuous development of the E-commerce, it can be connected, anywhere, anytime via e-mobile with wireless service. Meantime, the activities made from e-commerce have generated huge amount of data. So much more transaction anomaly data are hidden in them. How to monitor and inspect these transactions become the key concerns and challenges. In this thesis, I will demonstrate how this research is dedicated to resolving these challenges. The proposed data mining-based transaction anomaly detection method can detect the anomaly action at the business and operation level. This thesis can be a roadmap for other data mining practitioners and web administrators on how to develop more effective E-commerce transaction anomaly detection models for protecting the online trading safety.

E-commerce is sales of goods and services over the Internet and extranet, electronic data interchange (EDI), or other online systems [Sugumaran, V. 2001]. In practice, E-commerce is more than just buying and selling products and services online. It is the process of buying, selling, transferring or exchanging products, information and services through computer networks [Sugumaran, V. 2001]. In other words, it uses Internet and Web to transact business. The fundamental purpose of e-commerce is to execute digitally enabled transactions [Scarle, S., Arnab, S., Dunwell, I., Petridis, P., Protopsaltis, A. and de Freitas, S. 2012]. It allows any size of business located virtually anywhere on the planet to conduct business with anyone anywhere. Therefore, the greatest contribution of E-commerce is to avoid the geophysical barriers and to impel all consumers as potential customers.

The vigorous development of E-Commerce depends on the progress of Internet technology. Online payment technology has become one of the supporting key technologies of E-Commerce. With the popularity of intelligent terminal and the improvement of the network infrastructure, mobile payment as an online payment also had a certain improvement. In 2008, the combined market for all types of mobile payments was projected to reach more than $600B globally by 2013[Wirelessintelligence.com. 2008], which would be doubled as of February, 2011. The mobile payment market for goods and services, excluding contactless Near Field Communication or NFC transactions and money transfers, is expected to exceed $300B globally by 2013 [Bonsoni.com. 2011]. Obviously, the perfection of technology and the change of consumption patterns will make the online payment is becoming increasingly popular.

Since the launch of Amazon in 1995, internet transactions have become an important part of the global marketplace. According to the Amazon annual report [Amazon annual reports, 2013],

their net sales increase from $2.5 billion to $61 billion from 2001 to 2012, of which 43 percent was outside North America.

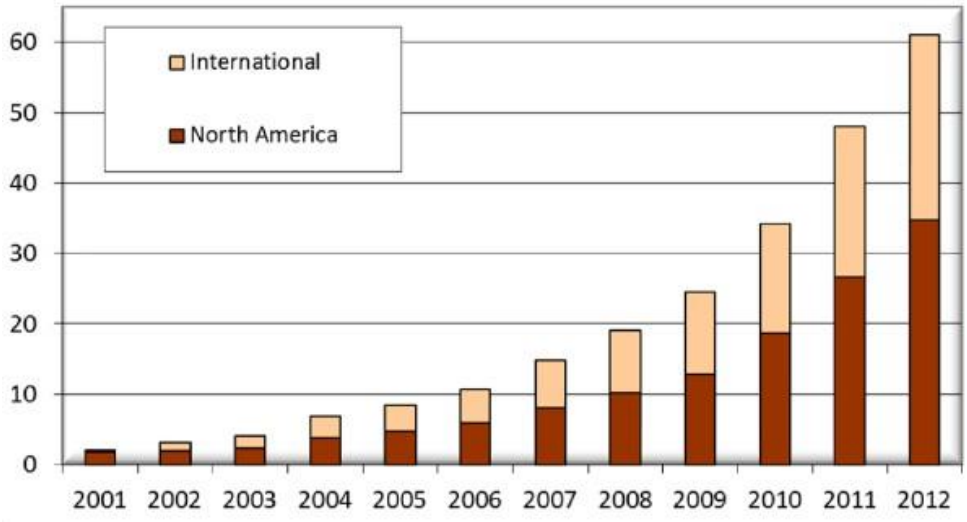


Figure 1. Amazon annual report

The Goldman Sachs investment bank predicted that the retail web sales will reach around $1 trillion by 2013 worldwide [Goldman Sachs annual reports, 2014]. The yearly increase reaches 19.4 percent since 2010.

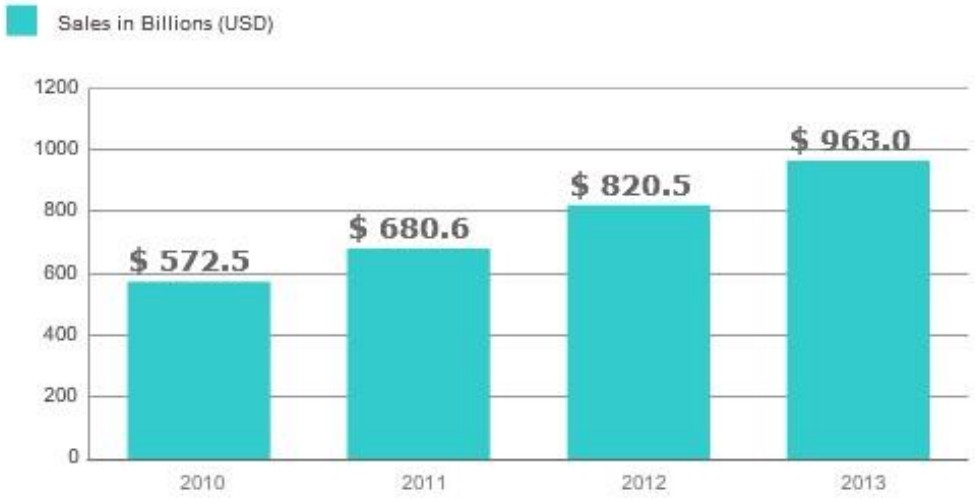


Figure 2. Goldman Sachs annual reports

From what has been described above, it is clear that the modern online business model has three characteristics, which are high efficiency, low cost and no restrictions of space and time. It becomes a trend in the development of business activities. However, the rapid development of E-commerce also gradually leads to many problems. For instance, E-commerce system collects the considerable large amounts of data, but valuable information only occupies a small proportion. Various types of attacks and fraud are increasing during the process of E-commerce transaction. Furthermore, due to the increase in credit card usage and convenient mode of online money transaction, fraudsters are also finding more opportunities to commit fraud, which affects banks and card holders to great financial losses [Sherly, k. k., & Nedunchezhian, R., 2010]. As the online money transaction is increasingly popular, more and more sensitive user information frequently transmit and store on the public Internet, the confidentiality of user account information has to face greater threats. Such as U.S. credit card information leak incidents in 2005, about 40 million user account information have been stolen. Another example is that Sony PlayStation Network had been attacked in 2011. About 77 million users privacy was leaked, some of which also bind with credit card information.

Therefore, how to effectively filter out and organize useful data from the mass database and obtain valuable information beneficial to business operations have become the trend of machine learning research field. Meanwhile, E-commerce transaction anomaly action appears frequently in causing the loss of the enterprise. Building an E-commerce anomaly detection system is the challenge to be settled urgently.

1.1 Problem Definition

Taking the E-commerce site as an example, the server logs, customer-related data and a large number of transactions are in the background database. All contained a large amount of useful data resources which are utilizable. However, these data are mixed with a large number of anomaly incursion data, disturbing the accuracy of data mining prediction. Therefore, how to quickly and effectively find all kinds of anomaly intrusions and guarantee the security of the system have become particularly important in E-commerce.

Typically, transaction anomaly action can be divided into two categories. The first category is that the system cannot be satisfied with the process of normal trading, format and other rules, such as duplicate transactions and tampered transaction. The second category is that the system satisfies the requirements of normal trading, but the transactions also have a certain fraudulent characteristics. For example, attacker steals the user transaction information for a trading, or legitimate users make an act of malicious overdraft.

In general, according to a specific implementation mechanism of payment system, which has the ability to detect and prevent the first category of transaction anomaly action. However, for the second category, the transactions which process and provide authentication information generally comply with the payment system. Thus, the payment systems are difficult to detect this type of transactions anomaly action. Obviously, the second type of execution of transaction will undoubtedly pose an enormous risk for the financial institutions and user. This thesis also focuses on this type of transaction.

According to the above mentioned theories, user's consuming behavior will be affected by their work type, income, consumption habit and living environment. It will show some particular

behavior pattern. The parameters such as type of item purchased, frequency, time and amount of purchase which are influenced by income, resource availability and life style are closely related to spending behavior of a person [Sherly, K.K., & Nedunchezhian, R., 2010]. Although there would be some fluctuations in the unexpected events or change of, but overall normal pattern will not change. For example, the low-income users most likely make the transaction in the last week of each month (due to the regular payroll date, rent or other factors) and the amount of each transaction may be relatively small. In this case, if this group of users successively has large volume transactions during a month, these transaction records should be considered as anomaly actions.

In addition, each person has their own operation habits in transaction process. Some users usually check their account balance before or after transaction. Others would like to double check the products information before purchase. Or some users will make the appropriate repayment operation immediately after using a credit card. All of the consumer behavior process can be used to detect whether the transaction flow match the system requirement or not. For example some customer is required to make the payment after the order is placed, then the system can easily find out if this transaction matches the requirements rule. Or the system also can detect some known abnormal behavior such as frequent password guessing attacks in their transaction log.

1.2 Significance

The development of Anomaly Detection model can be well used to resolve this problem. Anomaly Detection System [Lee, W. K., Stolfo, S. J., & Mok, K. W., 1999] is a proactive security protection technology. It is an important part of the information security architecture.

Since certain new mutual cooperation anomaly intrusion of attack methods continues to emerge, the improved E-commerce transaction anomaly detection models have become a new research subject in the field of anomaly detection. Generally, traditional detection technology has not been coped with increasingly complex anomaly data in the scalability and adaptability.

There is no doubt that E-commerce is a suitable application of data mining, and object-oriented data mining is a typical application of data mining. Furthermore, as the rapid growth in large-scale data set today, data mining has become a cutting-edge research topic in the field of both information technology and database area, which makes data processing technology into an advanced stage, recognized as one of the most promising information processing technology. Therefore, based on E-commerce mass transaction log data, data mining is an arresting application of numerous technologies. It mainly combines various methods and techniques such as artificial intelligence, statistics, database, machine learning, etc. [Li, H., Zhang, N., & Bao, L., 2006]. Then analytical tools will be used to extract implicit data from a large number of irregular data set which include many potentially useful information and knowledge. The anomaly detection process uses a large amount of collected information, such as transaction system logs, audit records, and network data packets to analyze or discover the process of intrusion or abnormal trisection records. Anomaly detection process is a process of data analysis. Therefore, for the anomaly detection systems, people can use data mining techniques to analyze the massive transaction data and extract the hidden security information as much as possible.

After careful study, the Integrating Classification with K-means to Detect E-commerce Transaction Anomaly model is proposed. The proposed meta-cluster with K-means model is based on the traditional clustering algorithm of data mining theory. It can effectively deal with data noise and detect experimental errors, and then generate a group of new transaction clusters

for the exact classification. Meta-cluster [Caruana, R., Elhawary, M., Nguyen, N., & Smith, C., 2006] aims at creating a new mode of interaction among users, the clustering system, and the data. Rather than finding one optimal cluster of the data, meta-cluster method [Caruana, R., Elhawary, M., Nguyen, N., & Smith, C., 2006] can find many alternate good clusters of the data and allows the user to select which of these clusters is the most useful and reasonable. After a careful study of meta-cluster method, the traditional K-Means algorithm has been improved, and then used to build an integrate meta-cluster with K-means model. This model can generate a group of clusters with a set of similarity instances. The new generated clusters can be used for the next classification stage. Four classification algorithms such as Logistic Regression, Naive Bayes classifier, Decision Tree and Artificial Neural Network will be used to predict and evaluate the accuracy of the E-commerce transaction anomaly detection mode.

1.3 List of Contribution

Among the findings in this thesis, the following two aspects are most notable contributions to E-commerce Transaction Anomaly detection. From a data mining perspective, the proposed meta-cluster framework is positioned for improving the accuracy of transaction anomaly detection models that readily resolve these challenges:

- **Meta-learning data mining application:** This thesis will propose a meta-cluster with K-Means algorithm which can make user easily evaluate the clusters and efficiently navigate to a better cluster as the main purpose. The proposed meta-cluster with K-means algorithm can divide the data set into different clusters in the preprocessing stage. The instances in each cluster have a strong correlation. Therefore, this algorithm can effectively improve the performance in the classification stage.

- **Heterogeneous data modeling:** The E-commerce transaction anomaly detection model can handle data sets with a mix of continuous and categorical variables.

- **Effective preprocessing methods against data anomalies:** This improved K-Means algorithm can be well used to reduce the noise and the effect of isolated point on clustering. More importantly, it can look for k cluster centers to receive better a division effect.

- **Data imbalance problem:** This E-commerce transaction anomaly detection model can be effectively robust against bias caused by class imbalance.

From the real world application perspective, this thesis can bring some good solutions for the existing problems of anomaly detection model and provide some new ideas to stimulate the development of E-commerce field.

- **Effective response ability:** This meta-cluster with K-means algorithm can effectively and accurately make a timely response for generating a group of new clusters with a strong correlation.

- **Excellent model generality:** This E-commerce transaction anomaly detection model can enhance prediction accuracy for the finance, banking, securities in the industrial sector, and reduce loss.

1.4 Thesis Outline

This thesis is organized as follows: Chapter 1 is an overview of the research question addressed in this thesis; Chapter 2 presents a literature review of relevant data mining clustering and classification topics, and previous research regarding the E-commerce transaction anomaly

detection methods used in my thesis; Chapter 3 introduce the relevant data analysis techniques used in this thesis. Chapter 4 gives a formal introduction of my proposed method; Chapter 5 discusses the clustering and classification results and evaluates the observed advantages and limitations of the proposed method; Chapter 6 concludes this thesis and discusses future work and potential improvement of the proposed method.

2. Literature Review

2.1 Literature Review of Anomaly Detection Model

Anti-money laundering (AML) detection technology is the predecessor of the E-commerce transaction anomaly detection techniques. The theory of anti-money laundering detection techniques was proposed in early 1970s. The implementation of information technology in the field of AML began in the 1990s. Nowadays, transactions can be easily done online. Modern technology has allowed money laundering to become an online crime, so the money laundering detection technology also needs to be established. In April 1990 [FATF Report, 1990-1991], the Financial Action Task Force (FATF) issued a report which estimated the amount of money laundered globally from 1990 to 1991 was around $800 billion to $2 trillion. The proceeds in drug industry had reached more than $300 billion, and most of which was laundered in the US

financial market. To solve the money laundering problem, an AML computer automatic monitor system is urgently needed to detect anomaly money transfers and money laundering. For example, the American Financial crime enforcement network (FINCEN) Artificial Intelligence System which was developed by Senator, Goldberg and Wooton [Senator, T.E., Goldberg, H.G., & Wooton, J., 1995]. This system used Bayesian models to determine the level of suspicious transactions, and then further analyzed the high level suspicious transaction data based on the previous results. It commendably integrated a variety of artificial intelligence technologies and software agents to identify the potential money laundering problem on the transaction reports. The tested results clearly showed that Artificial intelligence computer analysis system can greatly enhance the work efficiency and is an essential method for AML.

Two years later, Stofella [Stofella, P., 1997] developed a DBInspector model. This model was employed in the anti-money laundering activities performed by the supervision department of the Italian central bank. It focused on the employment of high performance database and 3D data visualisation technologies for the construction of a data mining environment. The DBInspector software environment was an integrated and open set of tools for the analysis and inspection of large databases. The users were allowed to interact with different data to process and visualize data flows in this environment. The DBInspector system was implemented in the high performance computing and networking environments, such as government, financial, and industrial organizations which maintained and managed large databases. Specifically, database servers based on parallel technology had allowed the possibility of real time inspection and

analysis of relational data stores in a large scale, which has not been possible in the past because of weak performance of available technologies.

In 2001, a decision support system based on data mining techniques in e-banking was proposed by Ionita, I. and Ionita, L. [Ionita, I., & Ionita, L., 2001]. They used data mining technologies in banking domain because it was suitable due to the nature and sensitivity of bank data and real time complex decision process. The main concern in this design was to make good decisions in order to minimize the risk level and anomaly transactions associated to the bank. Next year, Syeda, Zhang and Pan [Syeda, M., Zhang, Y.Q., & Pan, Y., 2002] used parallel granular neural network (PGNNs) to improve the speed of data mining and knowledge discovery process for credit card fraud and anomaly detection. Based on the implemented system, PGNNs algorithm could reasonably improve the 10-fold processing speed. However, this method had a limitation in the system that it needed more processors to solve the load imbalance problem, otherwise a high anomaly detection error would occur.

The Wolfsberg Group was formed by a group of international banks to share ideas on how to fight global money laundering using artificial intelligence (AI) System. In 2002, this group pointed out that International money laundering had been about the Mafia, drug smuggling, and arms deals involving sums in excess of $500 billion per year. Unlike many types of financial fraud, money laundering could range from a single transaction to the culmination of months of complex transactional activities. Additionally, the bank would not share past cases because of the sensitivity of information. To eliminate these increasingly complex transactional activities,

Kingdon and Feldman [Kingdon, J., & Feldman, K.S., 2002] designed a bank transaction data monitoring and analysis system which could automatically detect payment fraud and money laundering in the financial sector through the system. Two years later, Kingdon, J. [Kingdon, J., 2004] designed another artificial intelligence system to automatically identify a set of customer behaviour patterns. This system can efficiently identify customers' abnormal trading behaviours and make decisions dynamically, adaptively, and timely. People can use a machine with intelligence to judge context. This identification process could operate on the scale, and resolve problems with transparency and justification. The author believed this new generation of adaptive operational analytics could provide new possibilities from risk management to service provision.

With the development of the E-commerce, the traditional financial fraud and money laundering tend to be more complex and hard to detect. The advent of the era of big data is the future development trend of E-commerce field. As a result the E-commerce transaction anomaly detection technology becomes more and more urgent. The topic of how to exploit and apply an effective and real-time anomaly detection model has been widely researched by many specialists and scholars in recent years. Phua, Alahakoon and Lee [Phua, C., Alahakoon, D., & Lee, V., 2004] presented a multiple classifiers systems which was an innovative anomaly transaction detection method. They applied Back Propagation (BP) neural networks, together with C4.5 algorithm and Naive Bayesian as the base classifiers. Then they used meta-learning method to determine which classifier should be selected based on the skewed data distributions. The

originality of meta-classifier lies in how to choose the best base classifiers from a single meta-classifier (stacking). Then these predictions of base classifiers (bagging) were combined to improve cost savings (stacking-bagging) on automobile insurance claims. Based on the traditional classifier method, a new anomaly detection method (meta-learning approach) was compared with the C4.5 trained using undersampling, oversampling, and SMOTEing without partitioning (sampling approach) in this paper. Results showed that the fixed decision threshold and cost matrix were presented. They found several marginally higher cost savings using the partitioning and multiple algorithms approach. Furthermore, the combination of classifiers had produced the best cost savings from all three algorithms. In conclusion, although the anomaly and fraud detection method was not directly used as the target application, their approach was considerably practical.

In 2009, Kundu, Panigrahi, Sural and Majumdar [Kundu, A., Panigrahi, S., Sural, S., & Majumdar, A. K.., 2009] considered that E-commerce transaction anomalies are interspersed with genuine transactions. The simple pattern matching method which people frequently use cannot guarantee the accuracy of the detection results. Therefore, a combined anomaly detection model should be developed as misuse detection techniques. According to this paper, they used a BLAST-SSAHA Hybridization model for credit card transaction anomaly detection which was a heuristics that improved the performance of the sequence alignment algorithm. It combined two sequence alignment algorithms: the Basic Local Alignment Search Tool (BLAST) which was the most popular heuristic approach, and the Sequence Search and Alignment by Hashing Algorithm

(SSAHA) which was one of the fastest algorithms for sequence alignment. The BLAST-SSAHA Hybridization model included a two-stage sequence alignment. The first stage was profile analyzer (PA) which used past transaction record sequences to determine the similarity of an incoming sequence of transactions. The second stage was deviation analyzer (DA), during which stage the abnormal transactions were tracked on a deviation analyzer for possible alignment with past anomaly behaviors. The test result showed that the processing speed was fast enough to satisfy online detection of credit card transaction anomaly while maintaining high accuracy. The proposed BLAST-SSAHA hybridization approach can be effectively used to counter fraud in other domains such as telecommunication and banking fraud detection.

In the same year, Wang and Ju [Wang, N., & Ju, C.H., 2009] built a model based on similar coefficient sum to predict whether a credit card transaction is anomaly or not. This method focused on the outlier detection based on the similar coefficient sum. By computing the similar coefficient sum of every two objects, the anomaly record would be found. During anomaly detection process, two types of mistakes occurred because the anomaly data was far less than the normal data in this data set. The first type of mistake is that the abnormal transactions were mistakenly considered as normal transactions, which called the first class error or False Negative error. The second type of mistake is that the normal transactions were considered as abnormal transactions, which called second class error or false positive error. To address the two errors, the distance threshold parameter λ would be used as outliers to calculate the error rate and accuracy. The experiment results showed that when $\lambda=12$, the first class error rate reached the lowest and

the total accuracy was the highest. When $\lambda=9$, the second class error rate reached the lowest. The results showed this method was feasible and valid, and the performance was better than anomaly detection using clustering when the anomaly data was far less than normal data. However, there was a limitation for this method which was that the processing speed was relatively slow, as a result, the algorithm cannot be used in real transaction anomaly detection system, because it cannot forecast the probability of anomaly transaction quickly enough.

In 2010, Sherly and Nedunchezhian [Sherly, k. k, & Nedunchezhian, R., 2010] built a model called Bootstrapped Optimistic Algorithm for Tree Construction (BOAT) to detect credit card fraud. The main objective of this model was to build an efficient fraud detection system, which was used to capture the changes in customers' behavior by combining decision tree classification and K-means clustering techniques. The fraud detection process had two stages: the first stage was to compare the incoming transaction with the genuine transaction in the database, and then computed the profile score. If the deviation was obviously different with the normal behavior, it would then pass to the second stage. The second stage was to confirm the deviation score, and used the fraud history database to reduce the false alarm rate of suspected anomalies which were checked with the fraud history database. The BOAT model had some advantages. First of all, the model could support incremental update of transactional database. Secondly, the BOAT model could handle maximum fraud coverage in higher speed and lower cost. In conclusion, this proposed model was sufficient to use in real life E-commerce transaction data. More importantly, accuracy could be ensured in detecting fraud transactions. However, there was also a limitation

in this model: this method only focused on users' level anomaly and misuse detection. If users wanted to ensure highly secured transactions in the future, the system level fraud detection also needed to be extended by profiling the system behavior.

In 2011, Lu and Ju [Lu, Q. & Ju, C., 2011] built a credit card transaction anomaly detection model based on the class weighted support vector machine (SVM). This model focused on a large-scale, high dimensions and highly imbalanced data set. First of all, to solve the high dimensions problem, this method used the principal component analysis (PCA) to screen out the main factors from a great deal of indicative attributes. To reduce the training dimension of SVM efficiently. Then, this model adopted an improved SVM-Imbalance Class Weighted SVM (ICW-SVM) to validate the accuracy of model because the credit card transaction data were highly imbalanced. ICW-SVM algorithm set different weights on normal and fraudulent transactions to adjust the position of boundary, so it could avoid the problem of imbalanced data classification. The results showed that ICW-SVM with PCA algorithm (Accuracy=0.9128/Lift coefficient=7.3587) had higher precision and effectiveness than BP (Accuracy=0.8735/Lift coefficient=5.8756), Decision Tree (Accuracy=0.8468/Lift coefficient=3.2872), and Bayesian (Accuracy=0.8986/Lift coefficient=6.3177). Therefore, this model was practical and adaptive, and it was more suitable to solve credit card fraud and anomaly detection problem in real transaction data set.

In the same year, Minegishi and Niimi [Minegishi, T., & Niimi, A., 2011] developed a decision tree learning algorithm named Very Fast Decision Tree learner (VFDT). Since the data

were generated intermittently in different intervals, this algorithm used the real data as the data stream and used sensor-network and stream mining technologies to provide solutions for imbalanced data streams. In this paper, the researchers selected credit card transaction data as data stream to detect fraud and anomaly transactions. Because the extremely different rates of classes existed in the card transaction data, a new statistical criterion used in the node-construction algorithm named VFDT was proposed and implemented. During this process, the entropy class was weighted by Hoeffding bounds, and the Hoeffding bounds were compared to the information gain which was used to split the nodes of the conventional algorithm. The result testified that this method was a well fit for the imbalanced distribution data streams, such as E-commerce transaction data and credit card transaction data. However, due to the recall of the weighted class and the accuracy of the VFDT varied in this experiment, the results did not conform to the same weight. As a result, the best means to weight in these experiments could not be found. So how to decide the weight without pre-experiments became a very important challenge in the future work.

In 2013, Quan, Li, Jia and Han [Quan, Y., Jia, Y., Li, S.D., & Han, W.H., 2013] proposed a Co-occurrence Matrix algorithm based on users' behaviors. It accurately modeled users' behaviors using co-occurrence matrix. The co-occurrence matrix space was established to obtain profiles of the normal users' behaviors through the principal component analysis. In the detection phase, the method acquired the trading patterns of users, and then converted the patterns to the revised co-occurrence matrix. Then it classified users' behaviors as normal or malicious by

measuring the distance between the patterns and profile employed in the second matrix norm. This experiment collected four buyers' transaction records using ECMALL E-commerce system, and each buyer had 5000 log records. The experiment results showed that Buyer1's accuracy=0.945, recall=0.932, F-measure=0.938; Buyer2's accuracy=0.938, recall=0.928, F-measure=0.933; Buyer3's accuracy=0.920, recall=0.947, F-measure=0.933; Buyer4's accuracy=0.942, recall=0.923, F-measure=0.93. The experiment results showed that the proposed method had a good performance on forecasting accuracy, and the operating speed and storage space were acceptable. However, the shortcoming of this model was that this method could not deal with the transaction anomaly detection on cross-trading platform if more than one E-commerce trading platform was associated with the users.

In the same year, Sen and Dash [Sen, K. S., & Dash, S., 2013] used a meta-learning algorithm to detect credit card fraud and anomaly transactions. This model combined five supervised machine learning algorithms: Adaboost, Classification and Regression Tree (CART), Logitboost, Bagging, and Garding to classify credit card transaction data. The data mining results were compared on the basis of misclassification and correct classification rates to analyze the performance of different meta-learning algorithms. The results showed that the Correct Classification Rate of Bagging was 0.877, which was better than the other 4 algorithms (Adaboost was 0.847, CART was 0.834, Logitboost was 0.855, and Garding was 0.536). Therefore, Bagging was the best algorithm to detect fraud because it was easier to interpret compared to the other four algorithms. More importantly, Bagging algorithm had a smaller

misclassification rate. Therefore, these meta-learning algorithms could be applied on credit card fraud and anomaly detection.

2.2 Literature Review of Anomaly Detection Data Set

In the 2009 UC San Diego data mining contest, one of the main difficulties of the tasks was that the data are highly imbalanced, only about 2% of data were labeled as anomaly. Yang and King [Yang, H. & King I., 2009] used an ensemble learning approach to deal with the imbalanced E-commerce transaction anomaly data. The objective of this approach was to get benchmark results of these classifiers without much modification, and to help select a classifier for future tuning. First of all, the categorical variables were preprocessed and all variables were normalized. Secondly, several popular classifiers including Support Vector Machines, Neural Networks, AdaBoosts, and Logistic Regression were applied. Based on these algorithm results, the area under the ROC curve (AUC) was observed as a good indicator to improve the lift score. The AUC results of four algorithms respectively showed that SVM was 88.5%, RBF was 74.0%, AdaBoost was 87.3%, and LR was 85.1%. Then the ensemble method which combined the above classifiers was proposed to optimize the AUC score, the ensemble method was 89.0%. This developed method achieved significantly better results. However, there was a bottleneck in the whole experiment which was to find a way to deal with the high imbalanced data. This experiment adopted down-sampling on the negative samples, to alleviate the imbalance problem. However, as a heuristic way they did not find good parameters to improve the results.

Compared to the ensemble learning approach, neural networks algorithm was also widely used to detect abnormal data patterns in the area of online transaction. In 2010, Khosravani [Khosravani, R., 2010] presented a linear approximation to a Neural Network model for the same imbalanced data set of E-commerce anomaly detection. He considered neural networks as a non-linear model lack of transparency and interpretability. If a black-box approach was used for transaction anomaly detection modeling, the result would be undesirable. Based on this hypothesis, he proposed a novel technique, which could convert a non-linear model to an interpretable linear model at the tail of the score range. The score range was an actual scores matter. Based on approximation of the non-linear model at the high score range (e.g. top decile), the researchers used a linear model to evaluate the scores of neural network with the range of interest. In this experiment, the neural network model was approximated by linear regression and logistic regression models at the tail of the score distribution for detecting anomalies in the e-commerce transactional data. The linear model could be used to identify the high scores, and highlight the marginal contribution of each input variable to the output of the model. In this case, the accuracy of linear approximation approach was pretty high even in the high score range. It was obvious that the model comprehensibility became very important within this range of score. Therefore, how to identify the marginal effect is important to the users.

In order to improve operation speed, the other researchers like Catalin Patulea, Robert Peace and James Green [Patulea, C., Peace, R., & Green, J., 2010] also presented an implementation of genetic algorithm (GA) training of feedforward artificial neural networks (ANNs) classier in the

Compute Unified Device Architecture (CUDA) programming platform. There were four steps for the GA training of ANN algorithm. The first step was to compute the ANN output values for each training instance. Because ANN output had a high ratio of mathematical operations to memory accesses, it could be well suited to CUDA architecture. The second step was to find the threshold which defined the outputs fall in the top 20%. Because each top 20% calculation is performed independently for each candidate ANN, there could be enough candidates to occupy the entire device by the parallel performance of this step scales. The third step was to compute the number of top 20% instances which were truly positive. Then, the ratio of this positive rate to the overall positive rate was the lift of each candidate. The fourth step was to apply genetic operators: mating, mutation and selection. The selected candidate ANNs become the new population, to be used in the next generation. Compared to a traditional sequential implementation of GA training of ANNs, the graphical processing units (GPUs) method used in this paper was considered as parallel processors. In the result, this method promoted significant performance improvements at operation speed, and the GPUs method had 10-fold speedup. In the future study, the accuracy of this method could be improved by the incorporating hill-climbing methods, such as parallel backpropagation.

In the same year, Lee, Ham and Jiang [Lee, M., Ham, S., & Jiang, Q., 2010] proposed a new sampling method called Oversampling via Randomly Imputed Features (ORIF) special for the high imbalanced E-commerce transaction anomaly data set. Some of artificial instances could be generated for minority classes using this method. During the process of simple random sampling,

the existing feature values which corresponding to minority classes were imputed for a new minority class observation. Then they focused on the very high imbalanced data set which did not have many positives classes, applied the F-measure method to calculate the minimum misclassification error rate when all the observations were classified as negatives. The goal was to find some good classification algorithms, such as Linear Discriminant Analysis, Logistic Regression, Decision Tree, K-Nearest Neighbors, Naive Bayes, Support Vector Machine, Random Forest, Weighted Random Forest, SDC-Linear and SDC-Gaussian were used to validate the result of F-measure and Recall. Finally, the experimental results showed clearly that it would be able to improve classification performance by searching appropriate parameters in each classification method after using ORIF method. K-nearest neighbors (F-measure: 0.3403 / Recall: 0.3078), Random Forest (F-measure: 0.2885 / Recall: 0.2792) and SDC (F-measure: 0.3249 / Recall: 0.2745) had better performance than others. This study showed the main advantage of ORIF was that the distance metric was not required in the feature space which is hard to define when it is a mixture of numerical and categorical variables. However, ORIF also broke the interaction of features by randomly choosing the values from the whole positive observations, as a result may not work well if there were interactions of features which were significant.

In 2011, Yang, Cao, and Yan [Yang, Z., Cao, S., & Yan, B., 2011] proposed linear discriminant analysis and data mining approaches to identify the E-commerce anomaly. They considered that the imbalanced data made the data mining approaches dominated by the data of the majority class. To solve this problem, a Linear Discriminant Analysis (LDA) was used to

deal with the imbalanced data set for the E-commerce anomaly detection problem. It performed well in time complexity. In this experiment, BayesNet, Bagging and J48 were applied to conduct the evaluation. Because the imbalanced data made these approaches dominated by the data of the majority class, serious type-II errors would occur in the result. In order to minimize type-II errors, LDA was used to deal with the imbalanced data set. After the test, the AUC results of three algorithms were 82.1% for BayesNet, 82.7% for Bagging, and 79.4% for J48. They all yielded good performance. LDA could increase the AUC and sensitivity performance tremendously, but the accuracy rate and the specificity were lowered a little.

3. Data Analysis Techniques

E-commerce transaction anomaly detection can usually be based on the following two assumptions: The first assumption is that there is a clear distinction between existing normal transactions and anomaly transactions. The second assumption is that the abnormal transaction action only occupies a small proportion in all transactions. In this thesis, I will focus on dealing with the transaction records under the second assumption. This chapter will discuss the following three types of E-commerce transaction anomaly detection technology: statistical classification methods, data mining classification methods, and clustering methods.

Statistical classification methods are the collections of methods which take a statistical approach to classify data instances. The traditional statistical classifier is to construct an underlying probabilistic model which provides a probability measure of class membership

instead of mere classification [Michie, D., Spiegelhalter, D.J., & Taylor, C.C., 1994]. The traditional statistical classifiers usually use distribution of the data points such as normal distribution and Poisson distribution to model E-commerce transaction anomaly detection, and then test the abnormal inconsistency. The statistical approaches frequently use two modern statistical techniques such as k-Nearest Neighbor (k-NN) and Naive Bayes (NB), and a classical statistical technique like Logistic Regression (LR). However, there are some limitations in these statistical approaches. For example, the data distribution in reality often does not match any of the known distribution. In addition, statistical approaches are not suitable for multidimensional data model and the results are usually unsatisfactory.

Data mining classification methods are gaining significant momentum nowadays in most industry sectors, and widely applied in anomaly detection field. Data mining is the process of finding knowledge from a large amount of data. It encompasses an interdisciplinary collection of methods and techniques from numerous scientific disciplines to achieve the purpose, such as machine learning, mathematics and artificial intelligence [Ngai, E., Xiu, L., & Chau, D., 2009]. The term actually represents a collection of processes including data preparation, data mining and result evaluation, and in the most cases it is referred as the essential process during which the knowledge is extracted from the embedding data [Han, J., & Kamber, M., 2001]. Consequently, the use of computer-based classification algorithms has the advantage on processing power over traditional statistical methods.

Data mining clustering methods are the process of making a group of abstract objects into classes of similar objects. This process indicates that clustering analysis methods firstly partition the set of abnormal transaction data into groups based on data similarity and then assign the labels to the groups. There are three types of theories mainly used in clustering analysis methods:

- The first one is deviation-based method. It uses a group of instance characteristics to identify the abnormal data from transaction data set. The described deviation from the given object is defined as an abnormal behaver. This approach is based on sequence abnormalities technology. It simulates a mechanism familiar to human beings. After seeing a series of similar data, an element disturbs the series is considered as an exception [Arning, A. Agrawal, R. & Raghavan, P., 1996].

- The second one is distance-based method, which can separate the outlier as abnormal data from the whole data set. According to Knorr algorithm [Knorr, E. M., & Ng, R.T., 1998], an outlier is a statistical term for any data value which seems to be out of place with respect to the rest of the data. This approach can detect abnormality of the global data. But distance-based method is not suitable for the data sets with various densities. Therefore, it is not suitable for the localized anomalies detection.

- The third one is density-based method. It uses local outlier factor (LOF) concept to find anomalous data points by measuring the local deviation of a given data point with respect to its neighbours [Breunig, M.M., Kriegel, H.P., Ng, R.T., & Sander, J., 2000]. Density-based

approaches have been successfully proven to detect different types of abnormal data, even in the extremely complex presence of noise.

As discussed above, the following techniques were used in this thesis:

3.1 Clustering Methods

Clustering algorithms involve various areas such as data mining, statistics, machine learning, database, etc. They are the process of dividing data set into several clusters. The data within the same cluster have a high similarity, and vice versa. Clustering algorithms are well known as data mining algorithm which can be used to detect users' anomalous behaviours, as well as unusual behaviours in E-commerce transaction status. In the transaction anomaly detection model, many different types of abnormal data are mixed in the data set. Therefore, how to recalculate the cluster coefficient with strong dependency during the different instances is the key of solving this problem.

3.1.1. K-means

The most commonly used clustering algorithm is K-means. K-Means [MacQueen, J.B., 1967] is a clustering analysis method commonly used in the anomaly detection model, which is a dynamic clustering algorithm proposed by J. B. MacQueen in 1967. K-means is an iterative refinement algorithm which attempts to minimize a squared error criterion [Duda, R.O., & Hart, P.E., 2001]. Each cluster is initialized by setting its mean to a random point in the data set.

The input of the clustering algorithms is a data set. Each data record is usually presented by an attribute vector (x_1, x_2,…, x_n), where x_i is a continuous or discrete variable representing data as attribute values. The E-commerce trading system consists of various activity data records, which can be deemed as a set of attributes E and E = {E_1, E_2,…, E_n}, where n is the number of data objects in set E. Each object in set E is described by m attributes such as (F_1, F_2,…, F_m). Therefore, any event e_i in set E can be expressed as a m-dimensional feature vector (f_{i1}, f_{i2},…, f_{im}), where f_{ij} is a numerical value of F_j. The outputs of the clustering algorithm are several clusters. Each cluster contains at least one data object, and the data objects have similarity in the same cluster.

In order to use clustering algorithms, the difference which is usually expressed by distance should be calculated among the data. The distance calculation methods include the Euclidean distance, Manhattan distance, and Minkowski distance. One of the most commonly used methods is the Euclidean distance, which is calculated as below:

d(i, j)=$\sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+\cdots+(x_{ip}-x_{jp})^2}$, where I and j represent the two data of the data set respectively and p represent the number of attributes .

In the transaction anomaly detection model, the transaction log records show that the number of normal events is greater than the anomaly events. Several different types of relationships and correlations exist among the variables. Therefore, there are following problems and limitations with using K-means algorithm:

● Unable to determine the number of clusters k

When the K-Means algorithm is used to solve the problem, people usually do not know how many categories should be used to divide the data set and which category is the most appropriate. The k value as one of the input parameters has a great impact on the final clustering results, but there are no generally accepted solutions to determine the value k.

● Unable to select the initial cluster centers accurately

For the K-Means algorithm, the choice of the initial cluster centers has a very important impact on the clustering results. If the initialization falls into a nearby local minimum point, it will cause the algorithm to converge to the local minima points.

● Unable to address bias in clustering results caused by isolated points

When the data objects do not match the general model, the isolated points would be defined by the metrics of rules. For the anomaly detection, isolated points indicate anomaly data. K-Means algorithm is highly sensitive to isolated points, and the small amount of data has a greater impact on the clustering results. Isolated points will have a large extent of deviation from the average value and affect the accuracy of clustering algorithms.

3.2 Statistical Classification Approaches

3.2.1 Logistic Regression

Logistic regression is part of a category of statistical models called generalized linear models. This broad class of models includes ordinary regression and analysis of variance (ANOVA), as well as multivariate statistics such as ANOVA and log-linear regression. The logistic function transforms the linear combination into an interval [0, 1] [Ye, N., 2003]. Thus, in order to use logistic regression, the dependent variable is transformed into a continuous value which is a function of the probability of the event happening [Witten, I.H., Frank, E., & Hall, M.A., 2011]. According to this algorithm, it is used to predict the probability of normal or abnormal transactions. Four assumptions of Logistic regression analysis have been included:

- Dependent variable Y_i is a Dichotomous variable
- Data must be selected from a random sample
- The relationship between independent variable and dependent variable is nonlinear
- The independent variables are not multi-collinearity

This thesis is based on E-commerce transaction anomaly detection model to explain the principle of Logistic Regression analysis. I assume that dependent variable Y=0 stands for the transaction normal action and Y=1 stands for transaction anomaly action. P stands for the

probability of anomaly transaction occurring when Y=1, and P_i=1-P stands for the probability of anomaly transaction not occurring when Y=0.

$$\text{Logit}(P_1) = \ln\left[\frac{P_1}{1-P_1}\right] = \alpha + \beta_1 X_1 + \beta_1 X_2 + \ldots \beta_1 X_k$$

$$P_1\,(Y_i=1) = \frac{\exp(\alpha+\beta_1 x_1+\beta_2 x_2+\ldots.\beta_k x_k)}{1+\exp(\alpha+\beta_1 x_1+\beta_2 x_2+\ldots.\beta_k x_k)}$$

$$P\,(Y_i=0) = 1-P_i = \frac{1}{1+\exp(\alpha+\beta_1 x_1+\beta_2 x_2+\ldots.\beta_k x_k)}$$

x_1, $x_2 \ldots x_k$ are independent variables, β_1, $\beta_2 \ldots$ β_k are regression coefficients, and α is intercept. Logistic regression model frequently chooses 0.5 as a break point. If p>0.5, the E-commerce transactions are determined as abnormal transactions; if p<0.5, the E-commerce transactions are determined as normal transactions.

Therefore, Logistic regression can deal with binary dependent variables, and variables do not need to meet the normal distribution and homoscedasticity.

3.2.2 Bayesian Methods

Bayesian methods mainly consist of Naive Bayes classifier and Bayesian belief networks. Naive Bayes classifier is a very powerful classification technique because of its computationally simple process [Hand, D.J., & Yu, K., 2001]. Naive Bayes classifier is a simple model which describes a particular class of Bayesian network. All of the features are based on the assumption of class-conditional independence. Therefore, the Naive Bayes algorithm can predict the

probability of a given case within a certain class. More importantly, it is easier to be used because of its simplicity which requires fewer data to get a good result in many cases.

Bayesian Network is proposed to solve the particular deficiency of the Naïve Bayesian classifier by taking into account the joint conditional dependencies among the attributes via a directed acyclic graph (DAG) which illustrates the hierarchical relationship among all the attributes, where a directional link represents a parenthood of the low-ordered attributes to the high-ordered attributes. In the initial paper, the algorithm was coined [Pearl, J., 1985], it was specified that the graph implemented an arbitrary order bounded by factors like logical convention, and experiential knowledge. Each of the attributes is conditionally independent of other attributes. So it will be appropriate to use a Bayesian network when these relationships are presented, or have enough data to derive them. When dealing with the complexity and uncertainty of E-commerce transactions, the Naive Bayes classifier as a probabilistic model has an overall judgment advantage which can determine the overall characteristics of abnormal transactions.

For the E-commerce transaction data set, it normally can be described by an n-dimensional vector space where $X = \{x_1, x_2, x_3 ... x_n\}$, and the n attributes include $A_1, A_2, A_3 ... A_n$. The transaction data will be divided into two categories, which are the normal transaction C_1 and anomaly transaction C_2. If the transaction data set is X, the probability of anomaly transaction can be identified as $P= P(C_2 | X)$.

According to Bayes' theorem, which is $P(C_2 \mid X) = \frac{[P(X \mid C_2)\, P(C_2)]}{P(X),}$, prior class probabilities $P(C_2) = \frac{S_2}{s}$ where S_2 is the number of samples in the training set, and s is the total number of $P(X_k)$ training samples. There is no dependencies among these attributes, $P(X) = \prod_{k=1}^{n} p(X_k)$, and $P(X \mid C_2) = \prod_{k=1}^{n} p(X_k|C_2)$. By selecting a sample data, a proper threshold K is selected in the unusual trading activities, and then use value K as the baseline for the training data. This way use can improve the accuracy of the anomaly detection.

There are three ways to detect the E-commerce transaction anomaly data. The first way is to use the outliers of transaction data to analyze. The second way is to use the catastrophe point of transaction sequence which is based on wavelet analysis to detect. The third way is to use the abnormal trading paths on link mining to identify. As mentioned above, we can get the abnormal transaction data set, and then use Naive Bayes classifier in anomaly detection model. The probability of transaction anomaly activities is

$$P(C_2|X) = \frac{P(X_k|C_2)P(C_2)}{P(X)} = \frac{\prod_{k=1}^{n} P(X_k \mid C_2)P(C_2)}{\prod_{k=1}^{n} p(X_k)}.$$

For the probability result, the value P and K will be compared. If $P \geq K$, the transaction records will be considered as anomaly. If $P < K$, the transaction records will be considered as normal.

Naive Bayes classifier has the following advantages. First of all, it uses the probabilistic models which are capable of providing overall judgement for uncertainty transactions. Secondly,

it takes full advantage of past experience through self-adaptive supervised learning. Thirdly, it can achieve high speed and high accuracy in large database applications.

3.3 Data Mining Classification Methods

3.3.1 Decision Tree

Decision Tree algorithm is a sequential partitions of a set of data which maximize the differences of a dependent variable (response or output variable). They offer a concise way of defining groups which are consistent in their attributes but which vary in terms of the dependent variable [Alfonso, P., Rafael, J., & Elena, G., 2011]. Decision Tree algorithm is an inductive arithmetic based on instances and it can deduce classification rules from a sort of cases without order and regulation.

To avoid overfitting the model, the tree can be pruned by eliminating the branches with few or scarcely significant entries [Alfonso, P., Rafael, J., & Elena, G., 2011]. As a result, if we start from the complete model, after the tree pruning this will gain in generalization capacity (assessed with testing data), at the expense of reducing the degree of purity of its leaves [Larose, D.T., 2005].

Decision Tree arithmetic is a favorable method because it can expeditiously estimate E-commerce transaction anomaly status from the transaction data set. Those transactions should be labeled with anomaly or normal action that further analysis is based on. In this thesis, NBtree algorithm is selected from the decision tree model.

NBtree is an open source Java implementation of the Decision Tree algorithm as the Weka data mining tool. NBtree was proposed by Manuel J. Fonseca, Joaquim A. Jorge [Fonseca, M.J., Jorge, J.A., 2004] as the successor of the formerly proposed R*-Tree and X-Tree algorithm. NBtree is a simple and compact algorithm to index high–dimensional data points of variable dimension. As discussed above, it is a discriminant model of decision tree where internal nodes partition the data into subsets and each leaf node contains a generative model for estimating conditional probability using variables not in the path to that leaf [Grigoris, K., 2002]. The assumptions of NB-Tree as below:

Let X = [X_1, X_2, …, X_n] be the vector of input variables, and Y be the output binary variable (response event). To compute probability of response P{Y=1/0|X_1, X_2, …, X_n }, one needs to make assumptions for independence amongst input variables. NB-Tree learns these assumptions from data by recursively building a decision tree [Kovalerchuk, B. & Vityaev, E, 2001].

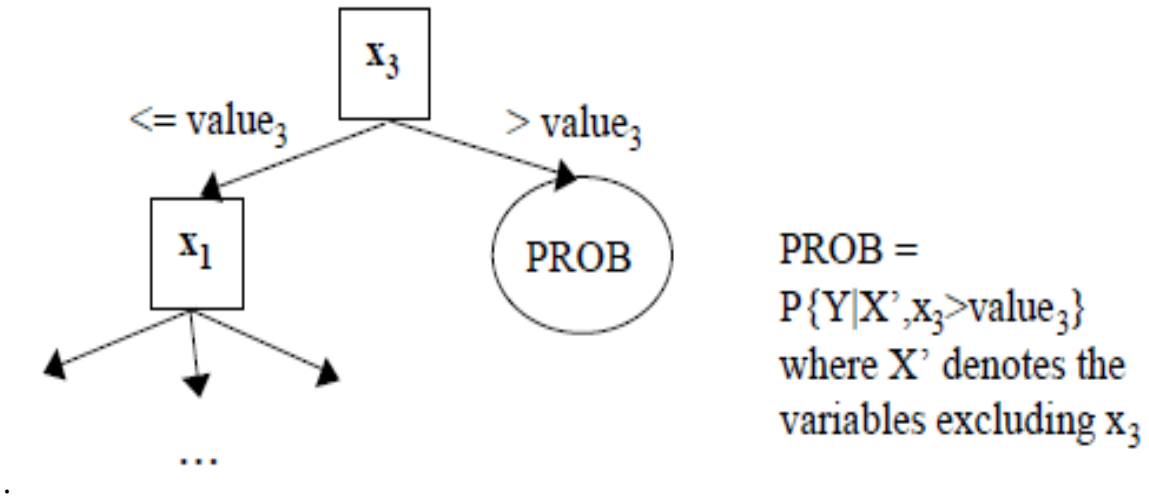


Figure 3. Decision tree [(Kovalerchuk, B. & Vityaev, E, 2001).]

NB-Tree working principle is as below:

From Bayes formula P (Y/X) = P(Y) * P(X/Y) / P(X)

$$P(X) = \sum P(X/Y_i) * P(Y_i)$$

$$P(X/Y) = \prod P(X_i / Y)$$

In the NB-Tree model, Y is a binary variable which is stranded by 0 (normal transaction) or 1 (abnormal transaction). If transaction status P>0.5, the transaction will be identified as occurring anomaly action. If P<0.5, the transaction will be considered as normal.

3.3.2 Artificial Neural Network

Artificial Neural Network (ANN) is an information operation system, which can imitate brain structures and functions. It is suitable to process the models which have many conditional inaccuracies and fuzzy factors. Neural network has so many characteristics such as distributed storage, fault tolerance, massively parallel processing, self-study, self-organization and self-adaptability etc.

The ANN algorithm consists of three basic types of nodes or layers, which are input layer, hidden layer and output layer (see Figure 4). Each node of input layer is corresponding to each of input variable; each node of output layer is corresponding to the target variable, hidden layer exists in the input layer and output layer. The complexity of the neural network is determined by the number of hidden layers and nodes in each layer. The input nodes are in charge of receiving

the initial values of data from each case in order to transmit them to the network. The output nodes receive input and calculate the output value (Kovalerchuk, B. & Vityaev, E, 2001). The neurons are information processing units of neural network, and they are the basis of neural network operation. Let X = (X_1, X_2, …, X_m) as neurons input; W = (W_{1i}, W_{2i}, …, W_{mi}) as the weight; θ as threshold value; and U and F respectively stand for the basis function and the activation function of neuron, which are the arithmetic unit of neuron.

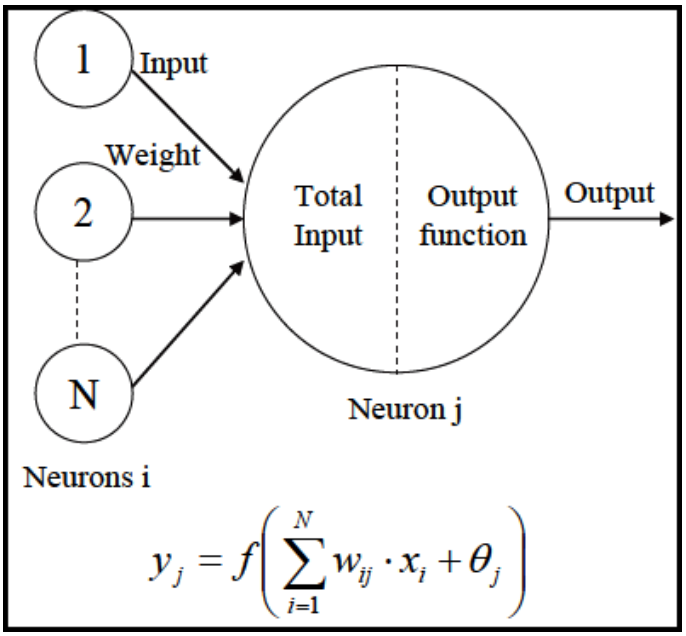


Figure 4. Neural network [(Kovalerchuk, B. & Vityaev, E, 2001).]

In this thesis, the Radical Basis Function (RBF) network algorithm will be applied in the classification stage. Next I will introduce the structure and operation process of the RBF network.

Radial function is a simple class of functions proposed by Powell in 1985. In principle, Radial function can be used in any linear or nonlinear model, and any single-layer or multi-layer

network. In 1998, Broomhead and Lowe [Broomhead, D. S., & Lowe, D., 1988] considered the RBF network was associated with the traditional radial functions in a single-layer network.

The RBF network uses the linear function as the basis function. If there is more than one hidden layers, a nonlinear function will be used in RBF network. RBF network learning process has two stages:

The first stage is the forward propagation process. The information is processed by each layer, and transmits from the input layer to hidden layer. Then RBF network calculates the actual output value y_i = F ($\sum_{m=1}^{n} X_m W_{mi} + \theta_i$).

The second stage is the reverse process. If the output layer failed to achieve the desired output value, the model will recursively compute the difference value (error) according to actual output and the expected output. Then it adjusts the regulation weight. This method uses RBF training network to repeat the process of forward propagation and error reverse propagation according to the training samples. When each training sample meets the algorithm requirements, the RBF network can be considered complete.

Therefore, RBF network algorithm has a high fault tolerance, learning capacity and error correcting capability. It can deal with continuous noise data very well and predict the results precisely.

4. Proposed method

Utilizing the traditional clustering methods to evaluate the E-commerce transaction anomaly has a limitation because of the large amount of variables and complex conditions, consequently making the result less accurate. To address this limitation, the meta-learning method is introduced which has the ability to solve this problem by repeatedly learning from the learning result. According to [Jurek, A., Bi, Y., Wu, S., & Nugent, C., 2012], the authors presented a new technique to combine multiple classifiers, which is less complex than all existing meta-models and has demonstrated the ability to improve classification accuracy in many application domains.

The concept of the proposed algorithm is to apply the improved K-means algorithm in the E-commerce transaction anomaly data set to generate a group of clusters. Each of clusters has a set of similar instances. Subsequently, classifiers will be applied in the generated clusters.

4.1 Cluster Modeling

Based on the investigation on traditional K-means algorithm, the proposed data processing methods will resolve pain points per identified at the end of chapter 3.1: the isolated points, the selection of k value and the selection of cluster center. Later on, the Meta-cluster with improved K-means algorithm will be applied in anomaly detection system to solve the defects of traditional E-commerce anomaly detection. Finally, through the optimized clustering, I will seriatim classify each cluster.

4.1.1 Isolated Point Handling

In the process of the anomaly detection, since the number of normal events is greater than that of the anomaly events, the distribution of normal events should be more intensive, and isolated point should be considered a record of intrusion event. Since K-Means algorithm is vulnerable to a disturbance by isolated points, the proposed algorithm has been improved to reduce the noise and the effect of isolated points on clustering. For each point *i*, we should calculate the distance D_i between two points, and calculate the average distance H. If D_i >H, then we can conclude that this point is an isolated point, where *n* is the dimensionality of the sample data.

$$D_i = \sum_{j=1}^{n} \sqrt{\sum_{h=1}^{d} (x_{ih} - x_{jh})^2} \quad (1)$$

$$H = \sum_{i=1}^{n} \frac{D_i}{n} \quad (2)$$

4.1.2 K Value Determination

K-Means algorithm divides the classification by Euclidean distance. Assuming the number of clusters is k, the initial value is k_1 and $k_1<k$, which means that at least two clusters can be considered as one cluster. If the initial value is k_2 and $k_2>k$, it means that a cluster will be divided into a number of clusters.

4.1.3 Initial Cluster Center Selection

My thesis will propose a method which can look for k cluster centers to receive better division effect. The isolated points are removed from the initial sample set, and then the number of cluster centers is calculated. The basic idea of this algorithm is as follows: firstly, a distance threshold value R is set. Secondly, the density of each data object relative to the distance threshold is calculated according to the definition of point density. Thirdly, data objects are sorted according to the size of density. Last but not the least, the densest data objects are identified as the initial cluster centers. The initial cluster center selection algorithm is as follows: Figure 5 presents a pseudo code representation of this approach. The program of initial cluster center selection algorithm is shown in Appendix B.1

Algorithm 1 Cluster Center Algorithm

Input: *N records of Dataset D, density radius R*

Output: *Initial Cluster center set S*

(1) Initialize the initial cluster centers set S as empty.

(2) For (read each record p_i in dataset D)

//Calculate the point density of record.

(3) Order the dataset D in descending order based on the point density size.

(4) While (dataset D still has unread records)

{

//Read the current record;

If (this record is read twice

//Put the record into the set S;

Else{

Calculate the distance between current record and each initial cluster center in the set S;

//Store the minimum distance in the dist-min;

If (dist-min < minnimum distance between records in set S)

If (the number of record in set S<k)

//Added the current record into set S;

Else if(the number of record in set S =k)

// Remove the record for which the density difference is the smallest from maximum distance point to the midpoint;

}

}

Figure 5. Cluster center selection algorithm

4.1.4 The Meta-Cluster with K-means Algorithm

Using integrated meta-cluster with K-means algorithm yields tangible positive influence over the three identified pain points. When people get the results of clustering algorithm, they can produce a number of clusters. Each cluster contains part of the transaction records. According to the previous assumptions mentioned in Chapter 2, there are substantial differences

in normal and abnormal transaction records. Because the number of normal transaction records is also far greater than the number of abnormal transaction records, I can set a threshold N. If the number of clustering transaction records (i.e., data objects) is greater than the value of threshold N, the cluster will be marked as normal. Otherwise, it is marked as an anomaly. According to these studies, the process of meta-cluster with K-means algorithm is as follows (the program is shown in Appendix B.1):

Algorithm 2 Meta-cluster with K-means Algorithm

Input: *The initial value of K that K = M, and input sample set*

Output: *The value of k*

Process:

(1**)** Scan the dataset A, and calculate the distance D_i and average distance H;

(2) for (each data point i)

If D_i> H, consider that point is an isolated point;

(3) Remove the isolated points from the dataset A' to obtain a new dataset, and record the number of samples in dataset A' as M, then output isolated points;

(4) Input the initial value of K that K = M, and input sample set A';

(5) Run K-Means algorithm to get K clusters;

(6) Merge, split, and cluster;

(7) Output the value of k;

(8) for (read each data pi from sample set A')

Calculate the point density of A';

(9) Order the dataset A' in descending order of the point density size;

(10) Scan each record in dataset A' to calculate the distance, and then obtain k initial cluster centers;

(11) Cluster;

Figure 6. The Meta-cluster with K-means algorithm

After using meta-cluster with K-means Algorithm, correlated transaction records should be clustered, and each cluster contains part of the transaction records. Next step, classification algorithms will be applied to separately evaluate each of the new generated clusters. In this thesis, four classifiers – logistic regression, Naive Bayes, NBtree, and RBF network – will be used to evaluate each of the generated clusters. These four classification methods have been proven to work best with the meta-cluster K-means, whose results indicate notable improvements over traditional methods as presented in chapter 5.6.

4.2 Data Preprocessing

Before clustering and classifying, a number of pre-processing decisions have to be made. According to the previous study of data sample analysis, two potential problems need to be addressed. The first one is multicollinearity problem. The second one is discretization problem. In order to address these potential problems, I have applied a number of heuristics aiming at optimizing the clustering process.

4.2.1 Multicollinearity Problem Handing

In this anomaly transaction data set, two groups of variables are highly correlated. The first group is “amount” vs. “total” and the second group is “hour1” vs. “hour2”.

First of all, the group of “total” and “amount” can be easily processed. The two variables are 99.95% identical of the time (45 different observations out of 94682 observations). Meanwhile, no abnormal transactions fall into these 45 observations. Therefore, feature “total” is removed completely. Next, to process the group of “hour1” and “hour2” is an important step. The two variables are 98.03% identical (1861 different observations), and 52 abnormal transactions are

hidden in the 1861 observation records. A new variable named “hourdiff” is created in light of this finding, which stands for the difference of “hour1” and “hour2”. It is proven by feature selection method that variable “hourdiff” is more relevant than variable “hour2” which is then removed from the original data set.

4.2.2 Discretization process

In this data mining task, many different forms of data can be used. There are two types of data in this transaction anomaly detection data set, which are continuous variable and discrete variable. Unfortunately, not all classifiers can process continuous data. Therefore, I choose a specific discretization technique as one of the preprocessing methods in this task. Discretization is the process that converts numeric variables into nominal variables. Discretization process can be divided into unsupervised discretization and supervised discretization. A supervised discretization method can be further classified by the way its algorithm proceeds: bottom-up (each value represents an interval and they are merged progressively to constitute the appropriate number of intervals) or top-down (the whole data set represents an interval and it is progressively divided to constitute the appropriate number of intervals). However, there are no significant performance differences between the two latest approaches [Zighed, D., Rakotomalala, R., & Feschet, F., 1997]. Unsupervised discretization can be grasped as a problem of sorting and separating intermingled probability laws [Potzelberger, K., & Felsenstein, K., 1993]. The existence of an optimum analysis was studied by Teicher (1963) and Yakowitz and Spragins (1963). However, this method needs a strong statistical hypothesis. In this thesis, an unsupervised discretization method has been used in preprocessing stage. The multivariate unsupervised discretization can be performed by clustering techniques using all attributes globally. It is also possible to consider each cluster obtained as a class and improve the

discretization quality by using unsupervised discretization methods [Chmielewski, M.R., & Grzymala-Busse, J.W. 1994]. Doing this method can bring many benefits in addition to satisfying data type requirement of a certain classifier. Specifically, it can be used to reduce the number of continuous attribute values. Discrete values can replace actual data values. Therefore it can reduce and simplify the original data so that the mining results are simpler and easier to use, and the training time may also improve significantly due to value reduction by discretization. The program of unsupervised discretization algorithm is shown in Appendix B.3. The process of unsupervised discretization consists of four steps as follows:

Algorithm 3 Unsupervised Discretization

Input: The value of *variables which need to be discretized*
Output: *The intervals value of continuous variables*

(1) Sort the values of the continuous variables to be discretized;

(2) Determine a cut-point of adjacent intervals for merging;

(3) Split or merge intervals values of continuous variables using some criterion;

(4) Stop at some point.

Figure 7. The unsupervised discretization algorithm

5. Results and Discussion

In this chapter, I will use a series of in-depth analyses of the classification task to validate the proposed method. By using WEKA GUI, several popular data mining classification algorithms have been performed to evaluate this data set. In order to make decisions about how to select models and variables, I respectively used 50% random splits of each cluster data for quick validations to see whether a result is better than others. According to several tests and validations, the logistic regression, Naïve Bayes, NB-Tree and RBF network are selected from available algorithms as my base classifier models. Then the 10-fold cross validation method will be applied to more accurately evaluate the E-commerce transaction anomaly detection model. Therefore, in the following sections of this chapter, the result will be discussed respectively on those best achievements, and we will see how meta-cluster with K-means algorithm has exhibited its performance advantages against other methodologies.

5.1 Data Sample Overview

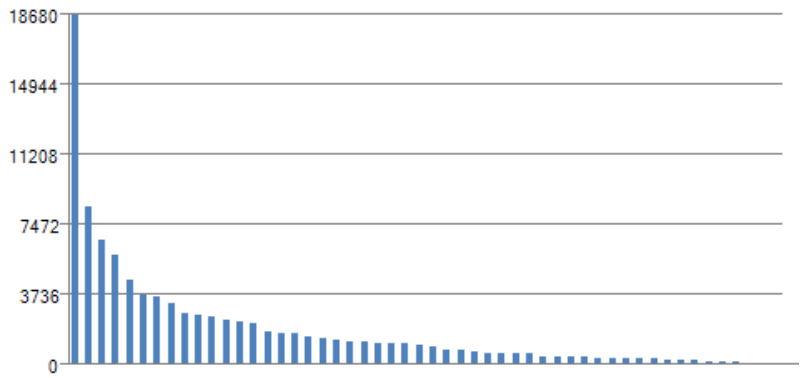
The data used in this thesis were obtained from 2009 UC San Diego Data Mining Contest task which was provided by Fair Isaac Corporation. The data set class labels are provided and named 0 (not anomalous) and 1 (anomalous), which include 94682 transactions, of which 2094 transactions are anomalies. The classes are highly imbalanced with only 2.2% of the observations being anomalous, which means that there are roughly fifty times as many negative examples as positive examples. Table 1 can clearly represent this class label.

Values	0	1
Records	92588	2094
Percentage of Records	97.8%	2.2%

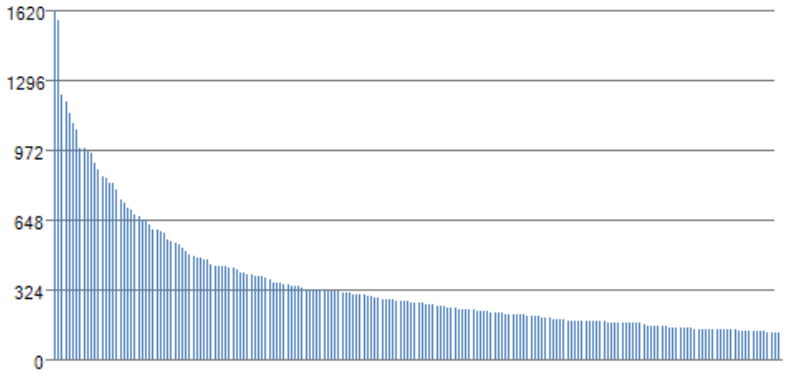
Table 1. Distribution of class label

This data set consists of nineteen variables. Three variables corresponding to “state”, “zip” and “email domain” are categorical variables, and the other sixteen variables are numeric variables. For the variable “domain”, 9810 category domains can be categorized into three groups: shopping websites, news websites, and email sites. AOL, YAHOO/MSN, and HOTMAIL/GMAIL are the top three most frequently used by users. Other domains, e.g. ‘MIL’ etc., seldom appear in the transactions. As seen in Figure 8-A, the variable “state” includes 53 states in this data set, in which most transactions states are recorded as CA, FL, TX, and NY, while other states such as AP (Armed Forces Pacific) and AE (Armed Forces Africa), etc. only appear in a few transactions. The variable “zip” as shown in Figure 8-B, consists of three digits, of which two represent region. The controversial opinion is to identify the variable “zip” as a categorical variable or a numeric variable. When I used “zip” as categorical variable, I found the

results were not satisfactory since there are a number of noises in the variable. The following histograms demonstrate the “state” and “zip” variables.



(A) Variable” state”

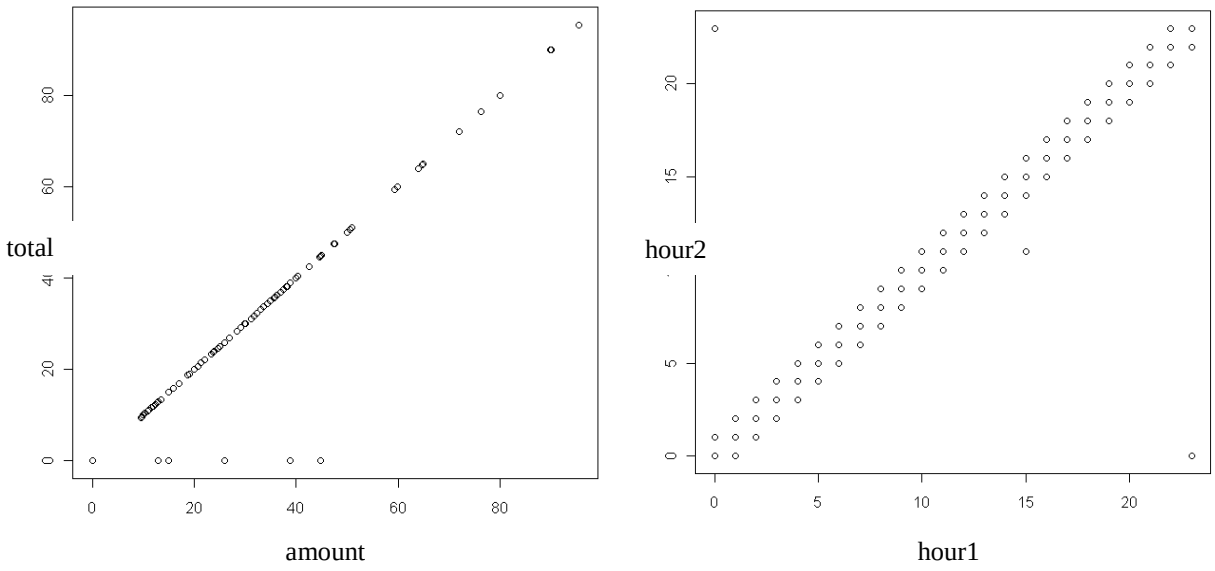


(B) Variable” zip”

Figure 8. Histogram for Variable "state" and "zip"

For the other sixteen numeric variables, the following two groups of variables have strong correlations with each other, as shown in the Figure 9. The first group is “amount” vs “total”, which shows only 45 observations are different from the 94682 transaction records. The two variables are 99.95% identical of the time. Meanwhile, there are no abnormal transactions in the 45 observations records (see Figure 9-A). The second group is “hour1” vs “hour2”, of which the two variables are 98.03% identical of the time, while the other 1861 observations are different

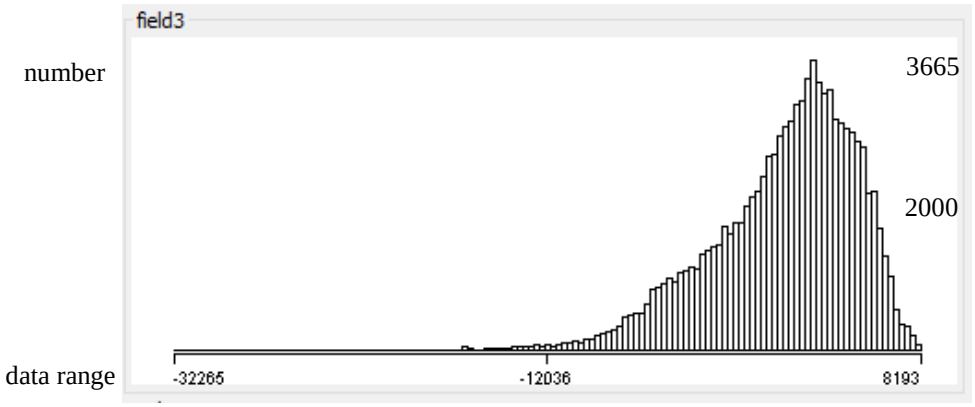
from the whole 94682 transaction records. According to validation, 52 abnormal transactions are labeled as 1, which are hidden in the 1861 observation records (see Figure 9-B).



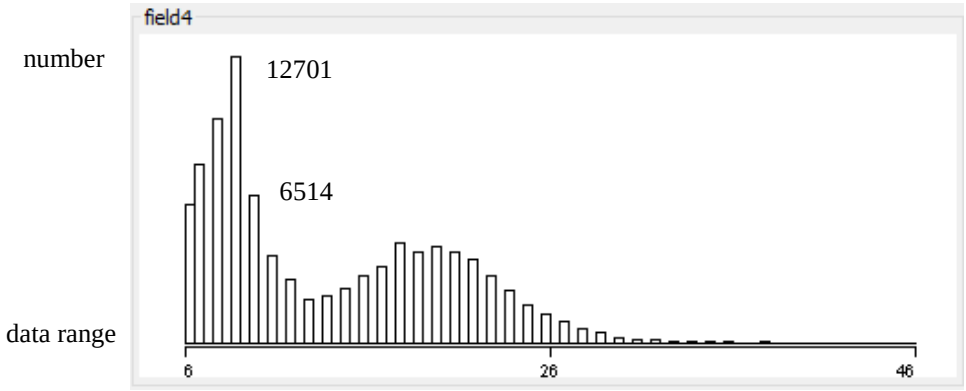
(a) "Amount" vs. "Total" *(b) "Hour1" vs." Hour2"*

Figure 9. Related variables

The rest twelve variables can be divided into three groups. The first group is feature "field" which includes "field1", "field2", "field3", "field4", and "field5". The five features take an integer value interval. The Figure 10-A clearly shows that the value range of feature "field3" is from -32265 to 8193. Another feature "field4" (shown in Figure10-B) represents bimodal distribution which has a value range between 6 and 46. The rest three features "field1, field2, and field5" are binary distribution (see Figure 26 in Appendix A.1). The second group is feature "indicator" which includes two binary features: "indicator1" and "indicator2" (see Figure 27 in Appendix A.1). The third group is feature "flag" which contains four binary features "flag1, flag2, flag3 and flag4", and another continuous feature "flag5" has positive integer values (see Figure 28 in Appendix A.1).



(A) Variable "field3"



(B) Variable "field4"

Figure 10. Distribution of "field3" and "field4"

5.2 Evaluation Criteria

There are difficulties encountered in the distribution of the data as well as the evaluation criterion. It is obvious that the measure of accuracy alone does not justify success of the imbalanced classification because a very high classification error rate on the minority class will not affect overall accuracy. In this regard, the overall accuracy of the classification will only be used as a reference, so more measures will be taken into consideration in the analysis, such as the Receiver Operating Characteristics (ROC), Sensitivity and Specificity. ROC is a very common measure to classify performance in machine learning field. The Area under ROC curve (AUC) is

one of the most used interpretations of ROC, which indicates the probability of a classifier ranks a randomly chosen positive instance is higher than a randomly chosen negative one. Fawcett [Fawcett, T., 2006] gives a detailed algorithm for generating the ROC curve and calculation of the AUC measure, which requires the target classifier possess a ranking mechanism to calibrate the likelihood of its classified instances as positive. In Weka, the measure of AUC is automatically calculated in the classification result's summary statistics. It is also claimed that AUC is very effective in evaluating cases of cost sensitive learning and imbalanced classification. Using AUC as the primary evaluation measure enables us to benchmark the result against the official leaderboard released by the 2009 UC San Diego Data Mining Contest, where AUC is selected as the only performance measure. Sensitivity and specificity, on the other hand, represent the probability of a positive/negative instance can be correctly classified. The Use of the two measures pertains to the statistical implication of the classification reveals that sensitivity is actually a derivative of the Type-I error rate and specificity is a Type-II error rate:

	Classified (0) **E-commerce nonanomaly**	**Classified (1)** **E-commerce anomaly**
True E-commerce nonanomaly	True Positives (TP)	False Positives (FP) Type II error
True anomaly	False Negatives (FN) Type I error	True Negatives (TN)

Table 2.Confusion Matrix

Sensitivity = TP / (TP +FN) = 1-Type-II Error %

Specificity = TN / (TN + FP) = 1-Type-I Error %

Since I am more concerned with the classifier's capability of recognizing the negatives instance which represents the true anomaly of the E-commerce transactions, sensitivity is

particularly more useful than the overall accuracy of the classifier in this imbalanced setting on the premise that specificity is not overly compromised. This quality can be easily observed from the AUC value.

5.3 Data Preprocessing

As mentioned in Chapter 4, I have already introduced the proposed method of meta-cluster with K-means algorithm, data preprocessing and transformation. I will provide a detailed explanation and analysis of the data preprocessing results with use of these methodologies. The implementation of proposed method is mainly based on the popular statistical software WEKA data mining package, which is an open-source software developed by the researchers of University of Waikato. WEKA software provides a series of data mining algorithms for classification and clustering, as well as a variety of data mining utilities for data preprocessing and feature selection. WEKA is open-source software. It offers users to use the developed WEKA modules and external Java applications. During the course of my thesis study, I have developed some custom Java programs which invoke WEKA data mining libraries for data analysis. The following preprocessing process such as meta-cluster with K-means algorithm, extraction of the clustering results, sample data discretization, nominal attribute value conversion and Information Gain feature ranking have been developed based on WEKA modules. The programs are either compiled into the native WEKA code package or are used as stand-alone applications for ad-hoc purposes. All the programs developed in this thesis can be found in Appendix B.

5.3.1 Data Preprocessing and Transformation Results

We already know that the multicollinearity problem exists in variables “hour1” and “hour2”. They are 98.03% identical and 52 abnormal transactions are hidden in the 1861 different observation records. To solve this problem, a new variable which named “hourdiff” will be used. The variable “hourdiff” is the difference between “hour1” and “hour2”. From Figure 11, the new variable “hourdiff” is a discrete variable which has values from -23 to 23. Variable “hourdiff” is considered as more relevant than “hour2” in this model, and variable “hour2” will be removed from the original data set.

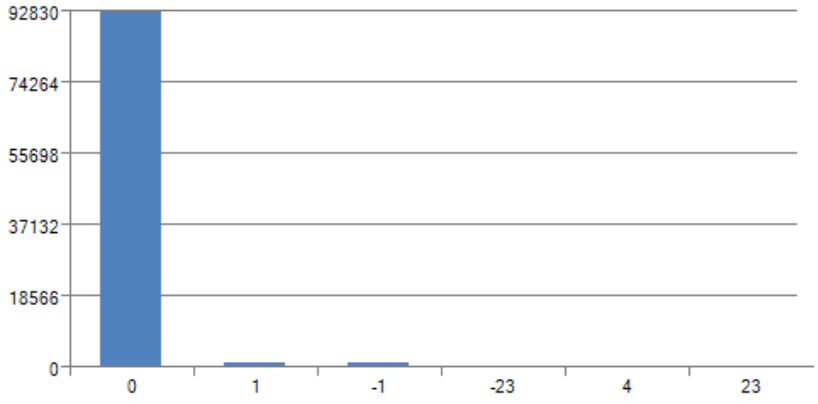


Figure 11. Histogram for variable "hourdiff"

The string variables need to be converted to numeric variables. For the variable “domain”, I decided to use number 1, 2, and 3 respectively to replace AOL.COM, YAHOO.COM/MSN.COM, and HOTMAIL.COM/GMAIL.COM. Other uncommon domains will be replaced by number 4. The distribution of processed variable “domain” can be shown in Figure 12

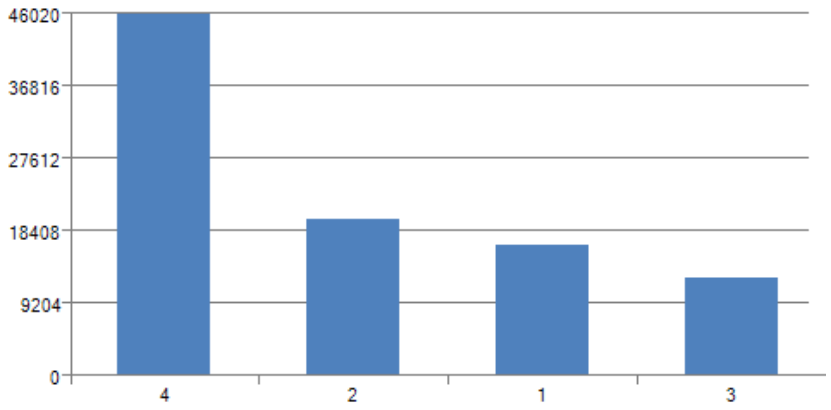


Figure 12. Histogram for processed variable "domain"

Another string variable “state” includes 53 states in this data set. I firstly calculated the number of each “state” category based on the number of the transactions. Then I set the variable “state” as a new category variable using digit order. At last, the original string variable “state” will be removed. The distribution of processed variable “domain” is shown in Figure 13.

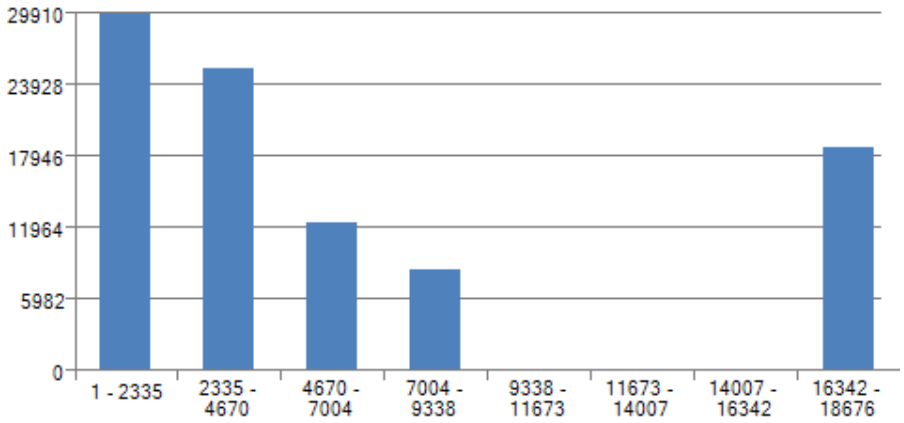


Figure 13. Histogram for processed variable "state"

After the data transformation and preprocessing, a group of new variables will be generated. The 18-dimensional variables which can be divided into two groups: discrete variables and continuous variables. Table 3 shows the two groups of variables.

Discrete Variable	Continuous Variable
hourdiff	amount
state	hour1
domain1	zip1
field1	field3
field2	
field4	
field5	
indicator1	
indicator2	
Flag1	
flag2	
flag3	
flag4	
flag5	

Table 3. The categories of variables

5.3.2 Results Using Meta-Cluster with K-means Algorithm

Before the implementation of the meta-cluster with K-means algorithm, the factor analysis should be considered. As the principle component feature selection was used for the purpose of reducing dimensions which successfully reduced the feature space from the original eighteen attributes to eight attributes. The principle component analysis performed on the original data set extracted eight principal components at the cut-off Eigenvalue of 1, with a cumulative variance of 77.31%. The result of principle component feature selection is proven to be good. The fewer principle components in the feature space accounts for a larger degree of variability. The breakdown of the total variance of the feature selection methods can be found in Table 4. According to the principle component analysis, eight features such as “amount”, “hour1”, “zip1”, “field1”, “field3”, “flag2”, “flag3”, and “flag5” are used in the meta-cluster with K-means algorithm.

Component	Initial Eigenvalues		
	Total	% of Variance	Cumulative %
1	1.736	9.643	19.643
2	1.572	8.731	28.375
3	1.495	8.305	36.679
4	1.239	6.881	43.561
5	1.112	6.179	59.740
6	1.079	5.994	65.733
7	1.050	5.833	71.566
8	1.033	5.740	77.307
9	.959	5.329	80.636
10	.940	5.223	81.859
11	.912	5.064	82.923
12	.900	5.000	85.923
13	.880	4.886	86.809
14	.829	4.603	87.412
15	.783	4.353	91.765
16	.537	2.986	94.751
17	.483	2.683	97.434
18	.462	2.566	100.000

Table 4. Principal Component Analysis

The use of meta-cluster with K-means algorithm is considered as a semi-supervised learning method. According to the result of principal component analysis, features “hourdiff”, “state”, “domain1”, “field2”, “field4”, “field5”, “flag1”, “flag4”, “indicatort1”, “indicator2”, and “class” are removed from the original data set. Finally, the meta-cluster with K-means algorithm generates twelve clusters. The result of “within cluster sum of squared errors” is 3512.32 using the seed number 100. It means that the effect of meta-cluster with K-means algorithm is fine. Table 5 shows the twelve clusters result.

Class / Record No		0	1
Cluster 11 (9037)	Records	8992	45
	Percentage of Records	**99.50%**	0.50%
Cluster 12 (9413)	Records	9372	41
	Percentage of Records	**99.60%**	0.40%
Cluster 6 (7632)	Records	7578	54
	Percentage of Records	**99.30%**	0.70%
Cluster 7 (7843)	Records	7790	53
	Percentage of Records	**99.30%**	0.70%
Cluster 1 (6477)	Records	6403	74
	Percentage of Records	**98.90%**	1.10%
Cluster 2 (5567)	Records	5477	90
	Percentage of Records	**98.40%**	1.60%

(A) Percentage of non-anomaly records > 97.8%

Class / Record No		0	1
Cluster 8 (3558)	Records	3088	464
	Percentage of Records	**86.90%**	13.10%
Cluster 5 (11226)	Records	10829	397
	Percentage of Records	**96.50%**	3.50%
Cluster 3 (10415)	Records	10106	309
	Percentage of Records	**97.03%**	2.97%
Cluster 9 (8726)	Records	8493	233
	Percentage of Records	**97.30%**	2.70%
Cluster 10 (5476)	Records	5349	127
	Percentage of Records	**97.70%**	2.30%
Cluster 4 (9318)	Records	9111	207
	Percentage of Records	**97.70%**	2.20%

(B) Percentage of non-anomaly records < 97.8%

Table 5. The results of meta-cluster with K-means algorithm

As seen from Table 5, the twelve clusters are divided into two groups. Group A includes cluster 1, cluster 2, cluster 6, cluster 7, cluster 11 and cluster 12. The percentages of the non-anomaly records of the six clusters are all higher than 97.8%. Based on the calculation, just 357 anomaly records are hidden in these six clusters, which account for 16.6% of the total 2094 anomaly records. While Group B includes cluster 3, cluster 4, cluster 5, cluster 8, cluster 9 and cluster 10. For these six clusters, the percentages of the non-anomaly records are less than 97.8% relative to the original data set, which is lower than Group A. Especially for cluster 8, the percentage of anomaly data is 13.1% which lower the negative effects of the overall percentage. The foregoing data shows that the meta-cluster with K-means algorithm has a good performance in the highly imbalanced data set.

After using meta-cluster with K-means algorithm, each generated cluster can be deemed as a training data set which is subsequently used to compose the final classifier. The results of the clustering results are then ready for the subsequent 10-fold validation.

5.3.3 Discretization and Data Modality Conversion

The Discretization and Data Modality Conversion are a very important step in data preprocessing, which has a direct impact on the result of the following feature selection and classification stages. In my research, four classifiers including Logistic Regression, Naive Bayes, RBFNetwork and NBtree have been used to evaluate each of the generated clusters. But RBFNetwork and NBtree algorithm cannot get a good performance when dealing with continuous variables. Therefore, discretization of continuous variables can be used to resolve this problem. Discretization can reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals [Zighed, D., Rakotomalala, R., & Feschet, F.,

1997]. Interval labels can then be used to replace actual data values. Discretization of continuous variables is also a requirement for the machine learning because those four algorithms only can work for nominal variable space.

In my thesis, this task is to respectively discretize the continuous variables in the twelve generated clusters and measure the effects of discretization on the performances of four machine learning algorithms. There are four continuous variables that I need to control in the discretization process. The generated clusters include the same variables for each data set. Each cluster defines each transaction record with eighteen variables, four of which are continuous variables (amount, hour1, zip1, and field3). There are two approaches to discretize each of the four continuous variables. The first one is to choose several threshold values, and then divide the instances into the same number of sets. For example, “amount1” between (0, 19.08] can be considered as LOW, (19.08, 47.7] can be considered as MEDIUM, (47.7, inf) can be considered as high. The second approach I could set a reasonable interval width values, and then label the values for each of these intervals. For instance, a label for each “amount1” interval width was defined 8 or 10 or so on. After many experiments, the feature “amount1” sets optimal interval width to 10. Figure 14 demonstrates a clear illustration for the discretization preprocessing of variable "amount1" as an example, with the same techniques applied to other continuous variables.

No.	Label	Count
1	'(-inf-9.54]'	1086
2	'(9.54-19.08]'	44563
3	'(19.08-28.62]'	6421
4	'(28.62-38.16]'	4291
5	'(38.16-47.7]'	31359
6	'(47.7-57.24]'	6126
7	'(57.24-66.78]'	782
8	'(66.78-76.32]'	3
9	'(76.32-85.86]'	6
10	'(85.86-inf)'	45

Figure 14. Discretization of variable "amount1"

Following the discretization process, data modality conversion is to use nominal attribute value conversion to convert the numeric variable to the nominal variable. This step is simple: when Weka imports data from an .arff file, it makes all of the cells with a number to be Weka type numeric. However, if the variables represent a small numbers of possible choices (e.g. 0 and 1), I just convert the variables from numeric to nominal (See Program in Appendix B.4). The numeric variables are discretized in the previous step to enable processing in this data modality conversion task.

5.4 Feature Selection

Feature selection is also a major consideration in preprocessing stage. Because Feature selection can reduce the dimensionality of variable space and remove the redundant or noisy data. Feature selection has two advantages: The first advantage is that feature selection can help enhance accuracy in many machine learning algorithms especially by means of variable ranking. Furthermore, the probability of over-fitting also increases as the dimension of feature space increases; feature selection is a powerful means to avoid over-fitting [Ng, A. Y., 2004]. The

second advantage is that feature selection can improve the efficiency of model training performance. In some situations, the data size may be very large so that model training and ranking are time-consuming. To solve this problem, feature selection should be implemented before model training and the complexity of the learning algorithms can be reduced.

There are three types of feature selection method in data preprocessing stage. The first method is filter, which can compute a score for each variable and then selects features according to the scores [Mladenic, D., & Grobelnik, M., 1999]. Yang and Pedersen [Yang, Y., & Pedersen, J. O., 1997] conducted comparative studies on filter methods, and they found that information gain (IG) and chi-square (CHI) are among the most effective feature selection methods for classification. The second method is wrapper. The learning system as a black box can be utilized to score the subsets of features. The last one called embedded method which will perform feature selection within the process of training [Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C.J., 1984]. In my thesis, I will use Information Gain Variable Ranking for feature selection preprocess.

Entropy is a commonly used measure in the information Gain theory, which characterizes the purity of an arbitrary collection of examples [Novakovic, J., 2009]. It is based on the Information Gain variable ranking methods. The entropy of Y is: $H(Y) = -\sum_{y \in Y} p(y) log_2(p(y))$, where p(y) is the marginal probability density function for the random variable Y.

Given the entropy as a criterion of impurity in a training set, I can define a measure reflecting additional information about Y provided by X that represents the amount by which the entropy of Y decreases [Novakovic, J., 2009]. The Information Gain variable ranking method calculates a variable's entropy score by:

Information Gain = H(Y)-H(Y|X) = H(X)-H(X|Y)

I implement the Information Gain Variable Ranking for the next four classification algorithms. Take cluster 8 as an example, the result of Information Gain Variable Ranking can be shown in Table 6, and the other results are included in Appendix A.2

Cluster 8		
Variable	**Ranking**	**Score**
field3	1	0.483572575
field1	2	0.109295816
zip1	3	0.08423172
field4	4	0.028974695
flag5	5	0.020937558
amount	6	0.015361589
flag3	7	0.012724533
state	8	0.011144217
hour1	9	0.004305125
field5	10	0.004037672
hourdiff	11	0.003270264
domain1	12	0.00250687
indicator1	13	0.002027886
flag4	14	0.00147493
flag1	15	0.000551059
field2	16	0.000409471
flag2	17	4.89383E-06
indicator2	18	4.83E-09

(The score of variable "hour2" is 0.00215796)

Table 6. The results of Information Gain Variable Ranking

According to the results of ranking score, I selected variables "amount", "hour1", "state", "zip1", "field3", "field5", "flag3", and "flag5" as the optimal variables for model validation.

5.5 Classification Modeling

I hereby propose a classification framework that is accurate, computationally efficient and can solve the problem of data imbalance. Previous introduced preprocessing methods have been completed, and the Information Gain Variable Ranking method has removed the irrelevant and/or redundant features based on feature-class and inter-feature correlation. Each preprocessed data set is then used to validate the E-commerce transaction anomaly detection model. The classification performance will be evaluated on the four scales: accuracy, AUC rate, sensitivity and specificity. When I evaluate the sensitivity and specificity rate, the results are based on their resulting confusion matrix of the proposed algorithm. I am going to prove that these classifiers are overall superior to some generally recognized classification methods adopted in the domain. The data mining analysis is conducted in the WEKA data mining software package, which contains a variety of popular machine learning algorithms for the data mining tasks.

5.5.1 Classification Algorithms

The Logistic Regression is also called a Logit model. It is used to model dichotomous outcome variables. In the Logit model the log odds of the outcome is modeled as a linear combination of the predictor variables [Ye, N., 2003]. The Naïve Bayes classifier is a quite straight-forward implementation of the algorithms. It uses the probabilistic models, which are capable of providing overall judgement for uncertainty transactions. More importantly, it can achieve high speed and high accuracy in large database. RBF network is an artificial neural network which uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters [Broomhead, D. S., & Lowe, D., 1988]. NBtree is a simple and compact algorithm to index high–dimensional

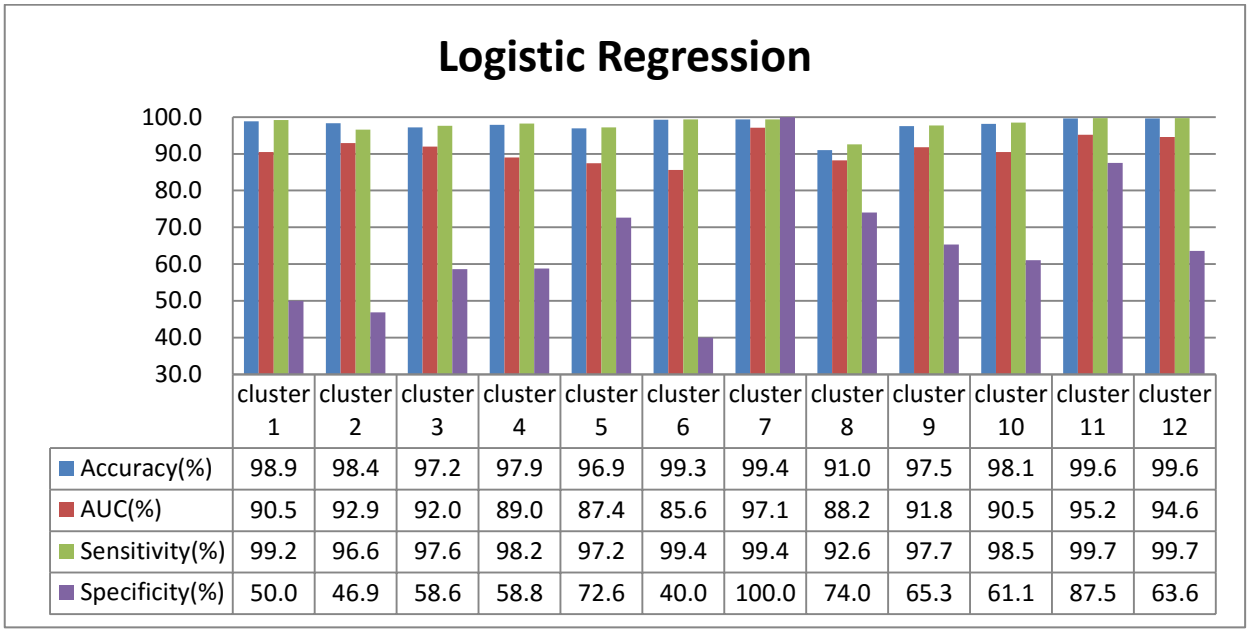
data points of variable dimensions. It is a discriminant model of decision tree where internal nodes partition the data into subsets and each leaf node contains a generative model for estimating conditional probability using variables not in the path to that leaf [Grigoris, K., 2002].

5.5.2 10-Fold Cross Validation Results

In this section, 10-Fold Cross Validation is used to evaluate the performance of the anomaly detection model. 10-fold cross validation is to partition instances into ten subsets, and then uses selected algorithms to train and test each subset. There are three steps when using 10-fold cross validation. Firstly, Weka software breaks the data into ten subsets of size n/10. Then, the selected algorithm is trained by nine subsets and tested on one. Lastly, the algorithm repeats ten times and takes a mean accuracy as the final result. As stated in previous chapters, AUC rate is the official evaluation metrics for the data set. The accuracy, sensitivity and specificity will be as reference metrics to help model selection in case additional evaluation criteria are required.

The result of the logistic regression algorithm's 10-fold cross validation is shown in Figure 15 where the results are clustered by each generated cluster, and the confusion matrices are shown in Appendix A.3 (A). I can see that the Logistic Regression algorithm has the best score in cluster 7 and cluster 12, which respectively have the highest among the twelve clusters with the AUC of 97.1% and 94.6%. The cluster 6 and cluster 5's results are worst among all the cases with the AUC of 85.6% and 87.4%. For the value of accuracy and sensitivity, each cluster's magnitude of accuracy and sensitivity is very close and relatively proportional to each other. Cluster 8 has a pretty good 91% accuracy, while other eleven clusters' are all over 97%. The performance of sensitivity is also excellent. The sensitivity for all of the twelve clusters is more than 92%. It indicates the logistic regression algorithm produces excellent prediction for the non-

anomaly cases. However, the result shows that there is a significant difference in the specificity measure. If I set 70% as a dividing point, four clusters have a high specificity rate. The cluster 7's specificity is even 100%. It means the entire anomaly records have been classified correctly. The other 8 cluster's specificity rates are among [40%, 65.3%]. The results of these eight clusters are a bit low, which indicate a high rate of type-II errors.

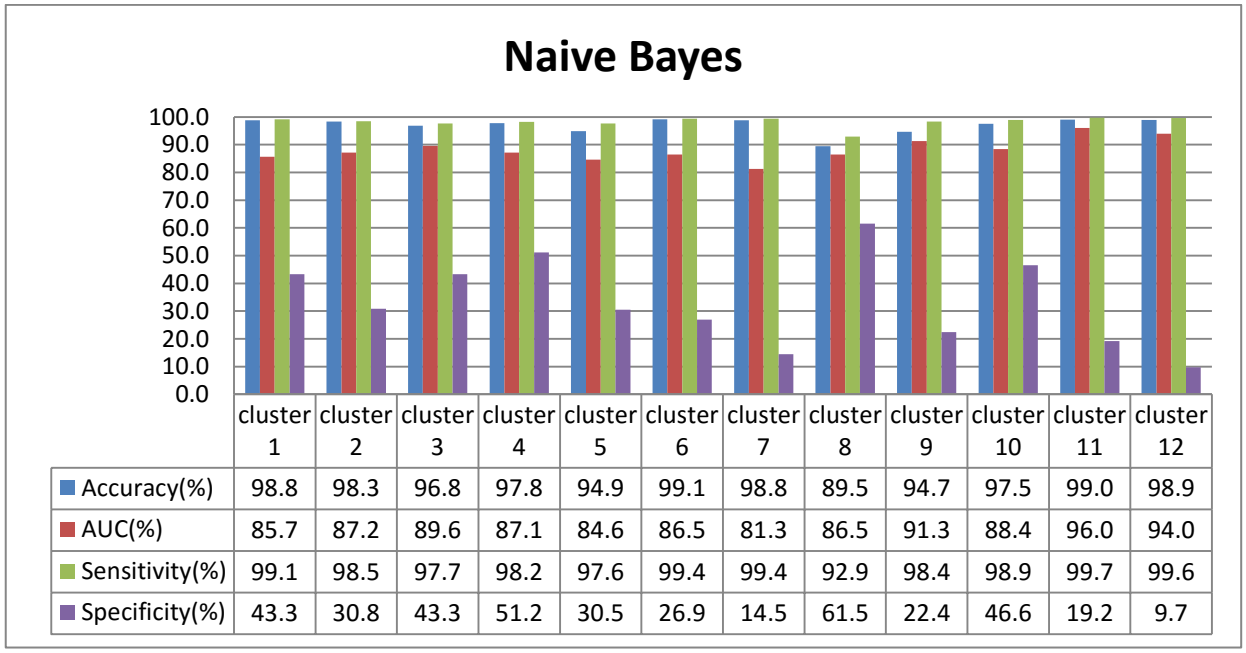


	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	cluster 11	cluster 12
Accuracy(%)	98.9	98.4	97.2	97.9	96.9	99.3	99.4	91.0	97.5	98.1	99.6	99.6
AUC(%)	90.5	92.9	92.0	89.0	87.4	85.6	97.1	88.2	91.8	90.5	95.2	94.6
Sensitivity(%)	99.2	96.6	97.6	98.2	97.2	99.4	99.4	92.6	97.7	98.5	99.7	99.7
Specificity(%)	50.0	46.9	58.6	58.8	72.6	40.0	100.0	74.0	65.3	61.1	87.5	63.6

Figure 15. Result of 10-fold cross validation of Logistic Regression

Compared to the value of sensitivity and specificity measure, it is clear that the sharp contrast of the values has been affected by the data imbalance problem. While the sensitivity is sufficiently high, the logistic regression algorithm is dominated by the data of the majority class, resulting very low specificity which means high Type-II error rate. However, according to the special performance of cluster 7 and cluster 11, the value of specificity is very high (e.g. cluster 7 has the 100% specificity; cluster 11 has the 87.5% specificity). The reason is the percentage of non-anomaly data in cluster 11 and cluster 7 respectively reach up to 99.5% and 99.3%, which

basically means no anomaly data in the two data sets. Meanwhile, it can prove that the meta-cluster with K-means algorithm can be well used in the high imbalanced data set.

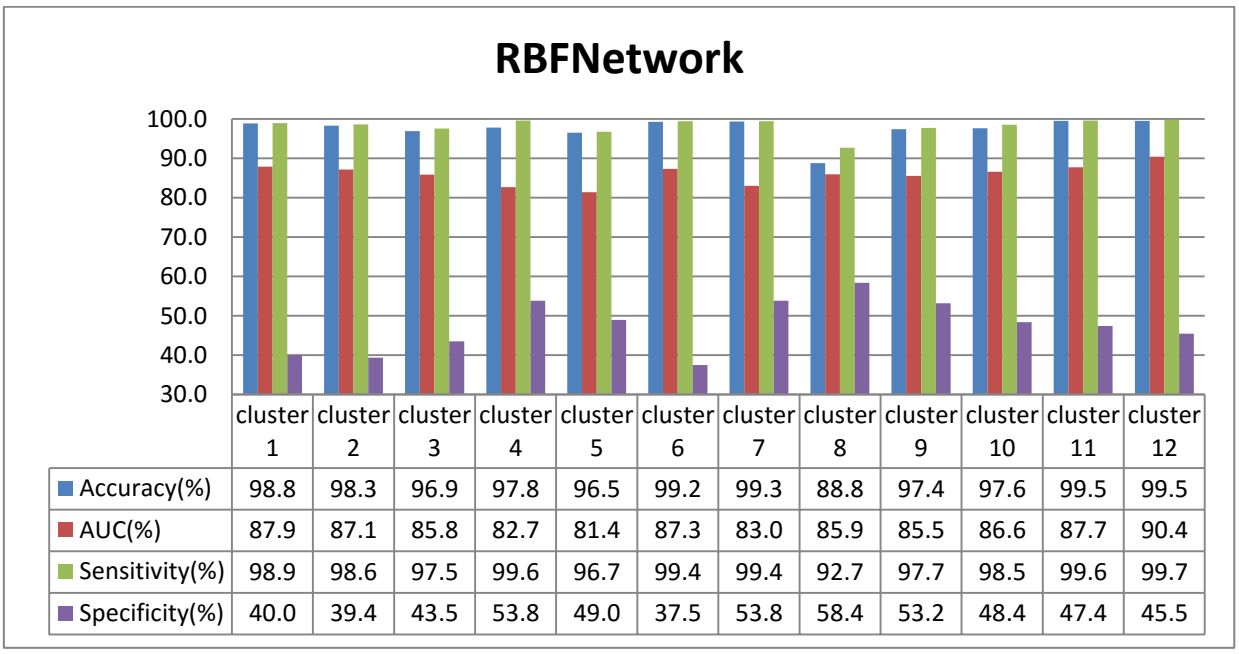


	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	cluster 11	cluster 12
Accuracy(%)	98.8	98.3	96.8	97.8	94.9	99.1	98.8	89.5	94.7	97.5	99.0	98.9
AUC(%)	85.7	87.2	89.6	87.1	84.6	86.5	81.3	86.5	91.3	88.4	96.0	94.0
Sensitivity(%)	99.1	98.5	97.7	98.2	97.6	99.4	99.4	92.9	98.4	98.9	99.7	99.6
Specificity(%)	43.3	30.8	43.3	51.2	30.5	26.9	14.5	61.5	22.4	46.6	19.2	9.7

Figure 16. Result of 10-fold cross validation of Naive Bayes

The result of the Naive Bayes algorithm's 10-fold cross validation is shown in Figure 16 in which the results are also clustered by each generated cluster, and the confusion matrices are shown in Appendix A.3 (A). From Figure 16, Naive Bayes classifier has the best AUC score in cluster 11 and cluster 12, which respectively present the AUC rate is 96% and 94%. For the other ten clusters, the AUC rates are all located in the interval range of (80%, 90%). The cluster 7 has the worst value with the AUC of 81.3%. The AUC results of Naive Bayes are generally lower than the Logistic Regression. To evaluate the accuracy and sensitivity, each cluster's magnitude of accuracy and sensitivity is very close and has an excellent performance. All of the accuracy rates are over 95% except that the cluster 8 falls into 89.5%. The sensitivity rates of 12 clusters are all more than 97.6% but cluster 8 is 92.9%. The result of specificity is significantly different

with other 3 measures. The specificity rates of cluster 4 and cluster 8 are higher than 50%, the other ten clusters' specificity rates are in the range of [9.7%, 46.6%]. The result indicates a high rate of type-II errors in Naive Bayes algorithm. Compared to the sensitivity and specificity, it is also clear that the sharp contrast of the values has been affected by the data imbalance problem. The Naive Bayes classifier will be dominated by the data of the majority class if the sensitivity is very high.

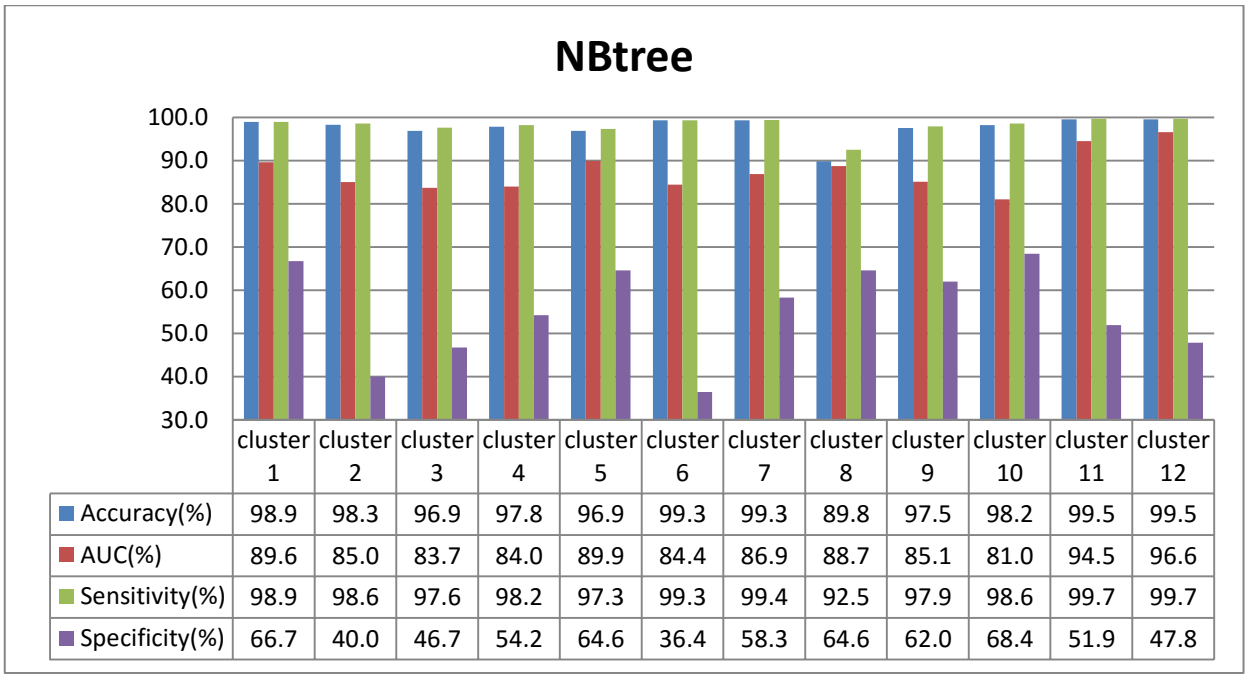


	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	cluster 11	cluster 12
Accuracy(%)	98.8	98.3	96.9	97.8	96.5	99.2	99.3	88.8	97.4	97.6	99.5	99.5
AUC(%)	87.9	87.1	85.8	82.7	81.4	87.3	83.0	85.9	85.5	86.6	87.7	90.4
Sensitivity(%)	98.9	98.6	97.5	99.6	96.7	99.4	99.4	92.7	97.7	98.5	99.6	99.7
Specificity(%)	40.0	39.4	43.5	53.8	49.0	37.5	53.8	58.4	53.2	48.4	47.4	45.5

Figure 17. Result of 10-fold cross validation of RBFNetwork

The third classifier which I applied is RBFNetwork. The result of the RBFNetwork algorithm's 10-fold cross validation is shown in Figure 17, and the confusion matrices are shown in Appendix A.3 (A). From Figure 13, RBFNetwork classifier has a similar performance to the Naive Bayes. All of the AUC rates are gathered located in the interval range of (80%, 90%). RBFNetwork algorithm has the best 90.4% AUC score in cluster 12. The cluster 5 has the worst value with the AUC of 81.4%. The AUC results of RBFNetwork are generally lower than the

Logistic Regression and Naive Bayes. For the results of accuracy and sensitivity each cluster's performance of accuracy and sensitivity is consistently excellent as the previous two algorithms. The accuracy rates of all of the clusters are more than 96.5% except cluster 8 (88.8%). The sensitivity rates of twelve clusters are all above 96.7% except cluster 8 (92.7%). The performance of specificity has a significant difference compared with other 3 measures. There are only four clusters such as cluster 4, cluster 7, cluster 8 and cluster 9, of which the specificity rates are beyond 50%. The others cluster's specificity rates are in the range of [37.5%, 49.0%]. The results of specificity rate indicate that type-II errors also exist in the RBFNetwork algorithm. Compared to the value of sensitivity, if the sensitivity rate is enough high, the logistic regression algorithm will be dominated by the data of the majority class. It indicates that the RBFNetwork algorithm has been affected by the data imbalance problem.



	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	cluster 11	cluster 12
Accuracy(%)	98.9	98.3	96.9	97.8	96.9	99.3	99.3	89.8	97.5	98.2	99.5	99.5
AUC(%)	89.6	85.0	83.7	84.0	89.9	84.4	86.9	88.7	85.1	81.0	94.5	96.6
Sensitivity(%)	98.9	98.6	97.6	98.2	97.3	99.3	99.4	92.5	97.9	98.6	99.7	99.7
Specificity(%)	66.7	40.0	46.7	54.2	64.6	36.4	58.3	64.6	62.0	68.4	51.9	47.8

Figure 18. Result of 10-fold cross validation of NBtree

The last algorithm is NBtree. Figure 18 can clearly present the result of the NBtree algorithm's 10-fold cross validation, and the confusion matrices are shown in Appendix A.3 (A). The Figure 18 has shown the performance of NBtree is similar to the Naive Bayes and RBFNetwork. The NBtree algorithm has the best AUC score in cluster 11 (94.5%) and cluster 12 (96.6%).For others clusters, all of the AUC rates are located in the interval range of (80%, 90%). The cluster 10 has the worst value with the AUC of 81.0%. The results of accuracy and sensitivity are also similar to previous three classifiers. All of the performance for the accuracy and sensitivity are consistent excellent except cluster 8. Cluster 8 has the lowest accuracy and sensitivity with the value of 89.8% and 92.5%. The performance of specificity stands in sharp contrast to other three indicators. The result of cluster 1, cluster 5, cluster 8 cluster 9 and cluster 10 are higher than 62%, but lower than 69%. The others cluster's specificity rates are in the range of [36.4%, 58.3%]. The results of specificity rate make clear that the RBFNetwork algorithm also has type-II errors.

5.5.3 Optimization Analysis of Twelve Clusters Results

As we have known, AUC rate is the official and important evaluation metrics for this imbalanced data set. According to observe the AUC% result of 10-fold cross validation and make a comparison to the four algorithms, the AUC results of cluster 11 and cluster 12 have obvious advantages compared with other clusters. AUC rates are all located in the interval range of (90%, 97%) except the RBFNetwork in cluser11 (AUC rate is 87.7%). Others cluster's AUC values are mostly concentrated in the range of (80%, 90%), but individual algorithm's AUC% value is more than 90%. As cluster 7 an example, the interval range of Naive Bayes, RBFNetwork and NBtree algorithms is in the range of [81.3%, 86.9%], but Logistic Regression

is as high as 97.1%. Throughout the performance of the four algorithms, logistic regression algorithm has the best AUC score in the twelve clusters.

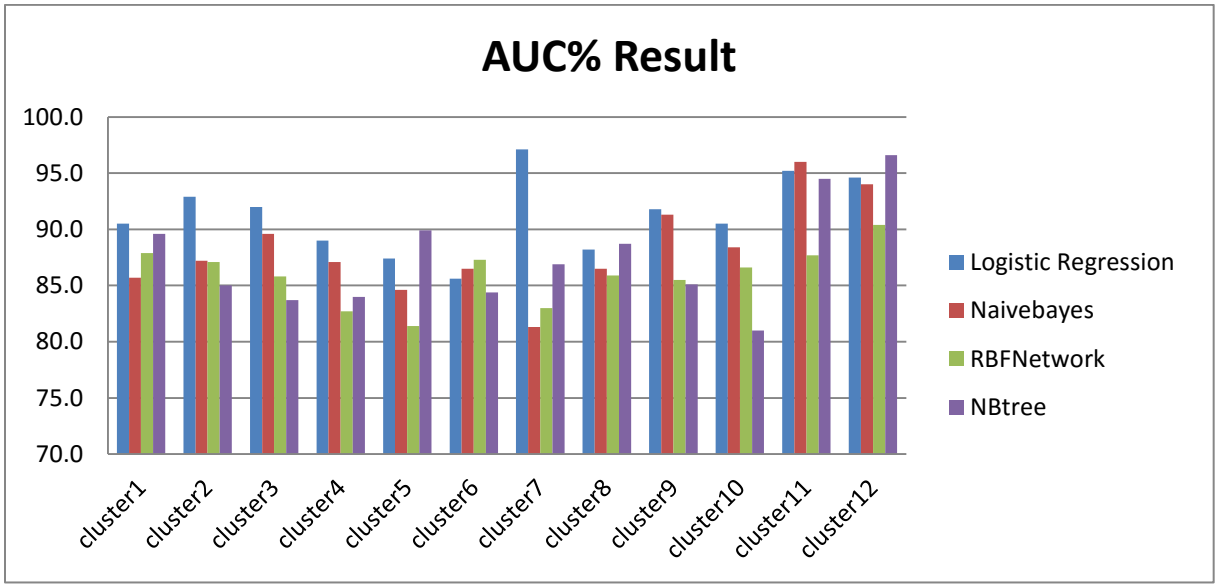
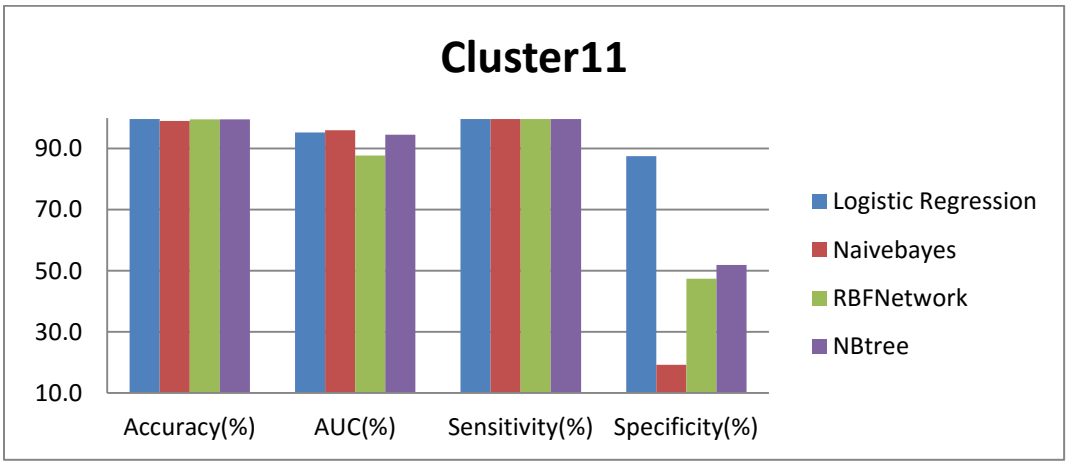


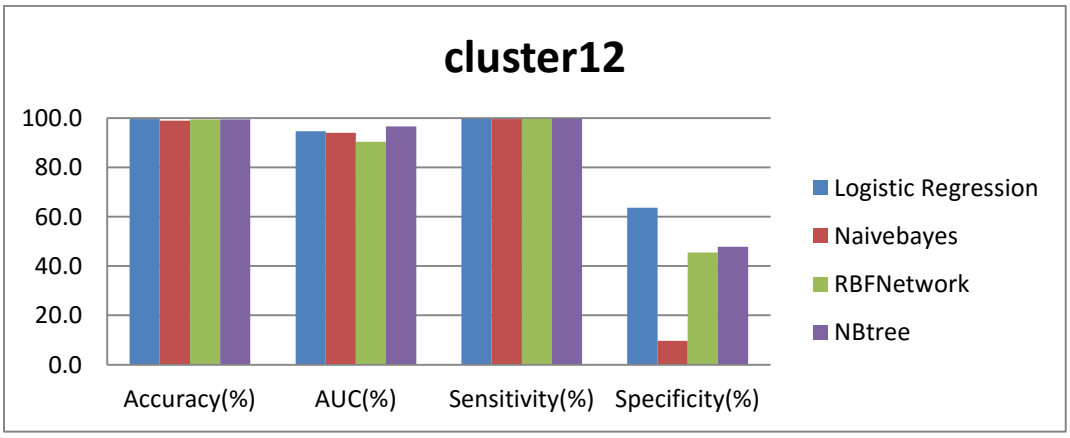
Figure 19. AUC% result of 10-fold cross validation

Next, I will provide a detailed analysis and explanation for the optimal cluster 11 and cluster 12. As seen in Figure 20, the results of cluster 11 and cluster 12 are very similar. Firstly, the results of accuracy and sensitivity are amazing excellent. For the two clusters, the accuracy of logistic Regression, RBFNetwork and NBtree are all over 95.5%. Naive Bayes's accuracy is a bit low, but also keeps on 99%. The Naive Bayes classifier has only managed to correctly predict 93 out of the total 9028 instances for cluster 11, and 99 out of total 9413 instances for cluster 12. The performance of Logistic Regression and NBtree managed to counter the data imbalance, showing much higher specificity and better AUC. As cluster 11 as an example, the Logistic Regression algorithm presents 95.2% AUC and 87.5% specificity. The performance of NBtree is excellent too, which takes AUC of 94.2% and specificity of 51.9%. Compared to Logistic Regression and NBtree, Naive Bayes and RBFNetwork exhibit slightly weaker performance in

the terms of AUC and specificity. While Naive Bayes is the best in the AUC (96.0%), but it gets the worst value with specificity of 19.2%. Such high Type-II error rate by Naive Bayes classifier is unacceptable, since most valuable information which embeds in the positive instances is overlooked and reflected by their misclassification. To sum it up, the Naive Bayes and RBFNetwork exhibit low specificity rate compared to the Logistic Regression and NBtree algorithm. But Naive Bayes classifier is more easily affected by this high imbalanced data set, and fall into the type-II errors problems.



(A) Cluster 11



(B) Cluster 12

Figure 20. Result of 10-fold cross validation of cluster 12 & cluster 11

5.6 Comparison to Other Models

In order to do parallel evaluation of meta-cluster with K-means algorithm, many researchers conducted in solving this high imbalanced classification problem have been reviewed. In Chapter 2, I have already introduced the preprocessing methodologies and classification algorithm used to solve this data set. As per statistics, I can see that Neural Networks, Support Vector Machines (SVM), Logistic Regression, and Decision Tree are classifiers most applied. Preprocessing techniques like Linear Discriminant Analysis (LDA), discretization and feature ranking are virtually indispensable in this data set, but the outcomes largely differ due to difference in method implementation. However, people seldom used cluster method for preprocessing. In practice, I consider that the success achieved by meta-cluster with K-means algorithm owes to sufficient data preprocessing. For this reason I will primarily discuss the optimal cluster 12 which is generated by meta-cluster with K-means algorithm and compared to those from other researchers, and then briefly about the classification algorithms used.

5.6.1 Evaluation for Preprocessing Effectiveness

To evaluate the performance of meta-cluster with K-means algorithm is the most important evaluation criterion for the model. Firstly, the original data set should be preprocessed by the same steps such as discretization, data modality conversion and information gain feature ranking. Next, I used the same classifiers which are Logistic Regression, Naive Bayes, RBFNetwork, and NBtree to validate the proposed approach. The result of the 10-fold cross validation is shown in Figure 21. I can see that traditional method also keep high accuracy and sensitivity, and the rates for the two indicators are all above 97%. The reason is these algorithms have been affected by the data imbalance problem. So I just focus on comparing the results of AUC and specificity.

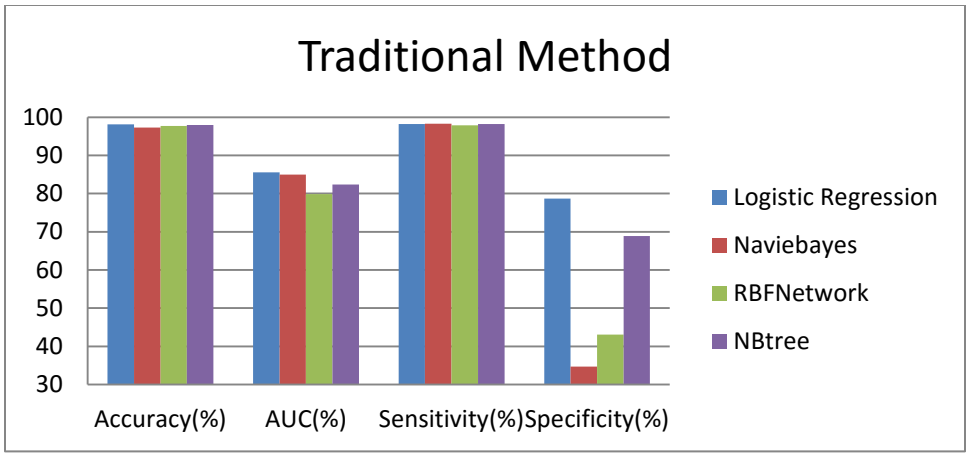


Figure 21. Result of 10-fold cross validation of traditional method

The first evaluation criterion is AUC rate. From figure 22 I find that the AUC rates of generated twelve clusters are obviously higher than traditional method. For the logistic regression algorithm, it is clear that a sharp contrast exist between clusters 1 to 12 and traditional method. AUC rates of clusters 1 to 12 are all gathered in the interval range of [85.6%, 97.1%], and their all performances are better than traditional method's AUC rate of 85.6%. The performance of RBFNetwork is as same as Logistic Regression classifier, all of cluster 1-12's AUC rates are better than traditional method which AUC rate is 79.9%. Naive Bayes and NBtree have similar performance. For the Naive Bayes, cluster 5 with AUC of 84.6% and cluster 7 with AUC of 81.3% are less than traditional method's AUC rate of 85.0%, other AUC results of 10 clusters are all over 85%. For the NBtree classifier, all of the AUC rates are more than traditional method's 96.5% except the cluster 10 (81.0%). According to the comparison of AUC performance, the generated clusters which are preprocessed by meta-cluster with K-means algorithm exhibit an excellent performance.

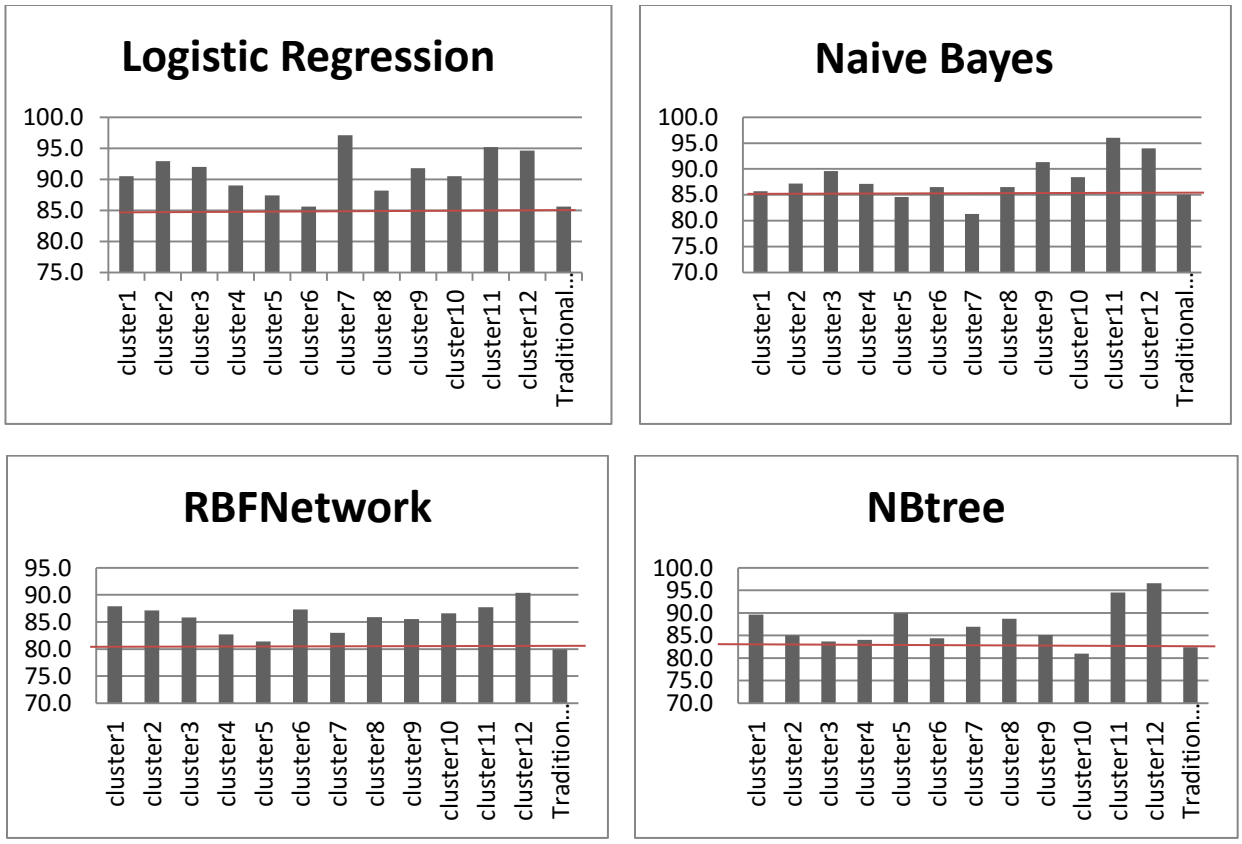


Figure 22. AUC comparison with traditional method

The second evaluation criterion is specificity rate. From Figure 22 the specificity rates of the generated twelve clusters have different performance for the four algorithms. Firstly, for Logistic Regression and NBtree classifiers, the specificity rates of clusters 1 to12 are generally less than the traditional methods. For the Logistic Regression classifier, only two clusters (such as cluster 7 with specificity of 100% and cluster 11 with 87.5%) have higher specificity rates than traditional method (78.7%). Other ten clusters have the specificity rate from 40.0% to 74.0%. The performance of NBtree classifier is obvious. All of the clusters get a less specificity rate than the traditional method (68.9%). These results indicate that the generated clusters are more easily affected by the data imbalance problem, and higher Type-II error rates exist in these four algorithms. Secondly, to evaluate the RBFNetwork algorithm, three clusters (such as cluster 1

with specificity of 40.0%, cluster 2 with 39.4%, and cluster 6 with 37.5%) have less specificity rate than traditional method (43.1%). Other nine clusters have the specificity rates in the range of [43.5%, 58.4%]. These results indicate that RBFNetwork has better performance on the generated clusters, and it can be used to reduce the Type-II error rate. Lastly, the specificity results of Naive Bayes are diverse. Half of the clusters have better specificity results than traditional method (37.4%) and the other half are less than 37.4%.

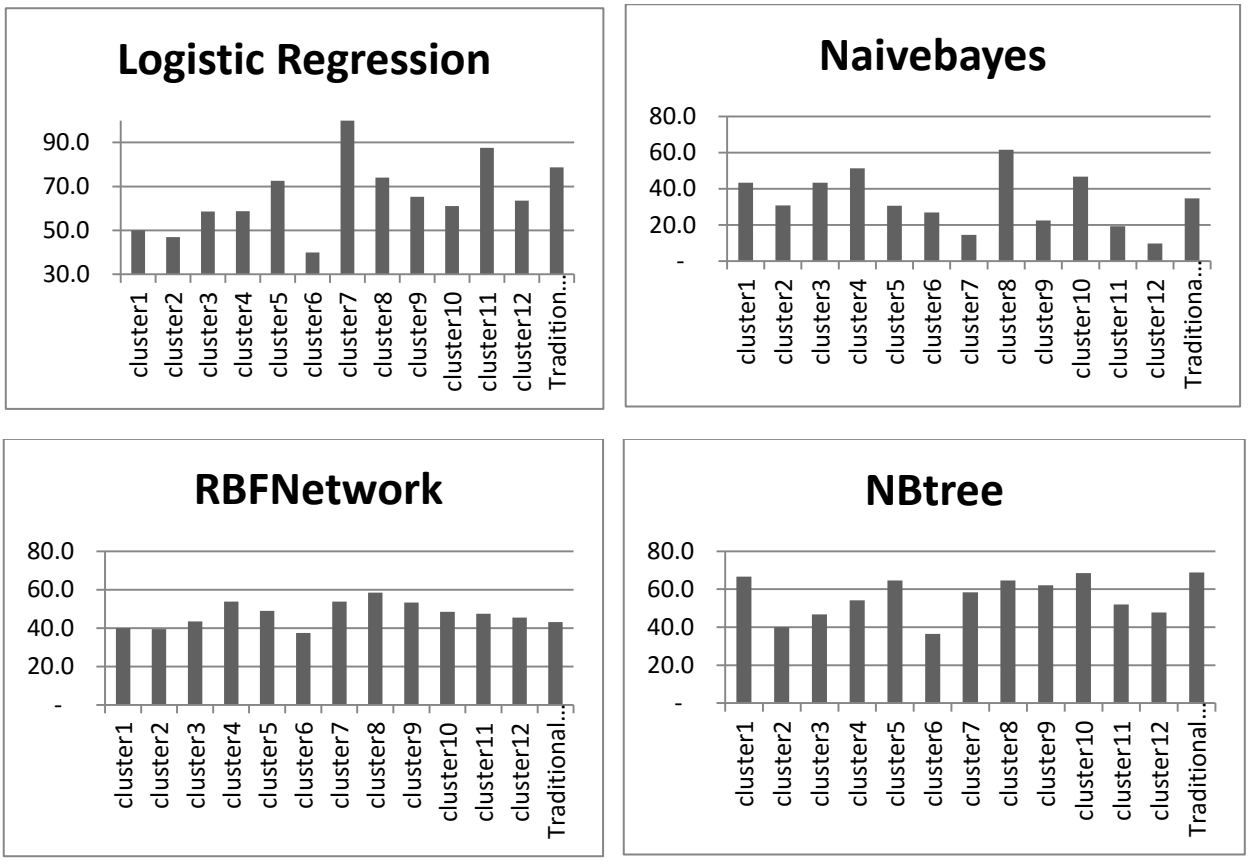


Figure 23. Specificity comparison with traditional method

In order to bring a more accurate and direct comparison, I will calculate the average accuracy, AUC, specificity and sensitivity of the proposed algorithm. The Classification Confusion Matrix of all twelve clusters results is as shown in Appendix A.3 (B). I use the following formula to calculate average results of proposed algorithm, which is shown in Table 7.

$$\text{AUC}=\frac{\sum_{ins_i \in positiveclass} rank_{ins_i} - \frac{M*(M+1)}{2}}{M*N} = \frac{Sum\ of\ true\ ranks - \frac{M*(M+1)}{2}}{M*N}$$

M is the number of positive samples, and N is the number of negative samples.

Accuracy= (TP+TN) / (TP+FP+FN+TN)

	Accuracy(%)	**AUC(%)**	**Sensitivity(%)**	**Specificity(%)**
Logistic Regression	98	91.5	98.4	66.4
Naviebayes	97.2	89.4	98.5	37.8
RBFNetwork	97.8	86.1	98.4	52.1
NBtree	98	88.7	98.4	59.6

Table 7. The average result of proposed algorithm

Next I will compare the average performance of proposed method with the traditional method. From Figure 24, we can easily see that Logistic regression (AUC=91.5%), Naive Bayes (AUC=89.4%), RBFNetwork (AUC=86.1%) and NBtree (AUC=88.7%) have more excellent performance than the traditional method which takes the AUC rate of Logistic regression (AUC=85.6%), Naive Bayes (AUC=85.0%), RBFNetwork (AUC=79.9%) and NBtree (AUC=82.4%). For the specificity indicator, the proposed meta-cluster with K-means algorithm can improve the specificity rate of Naive Bayes from 34.7% to 47.8% and RBFNetwork from 43.1% to 52.1%. For the other two indicators of accuracy and sensitivity, the proposed method produces a similar performance.

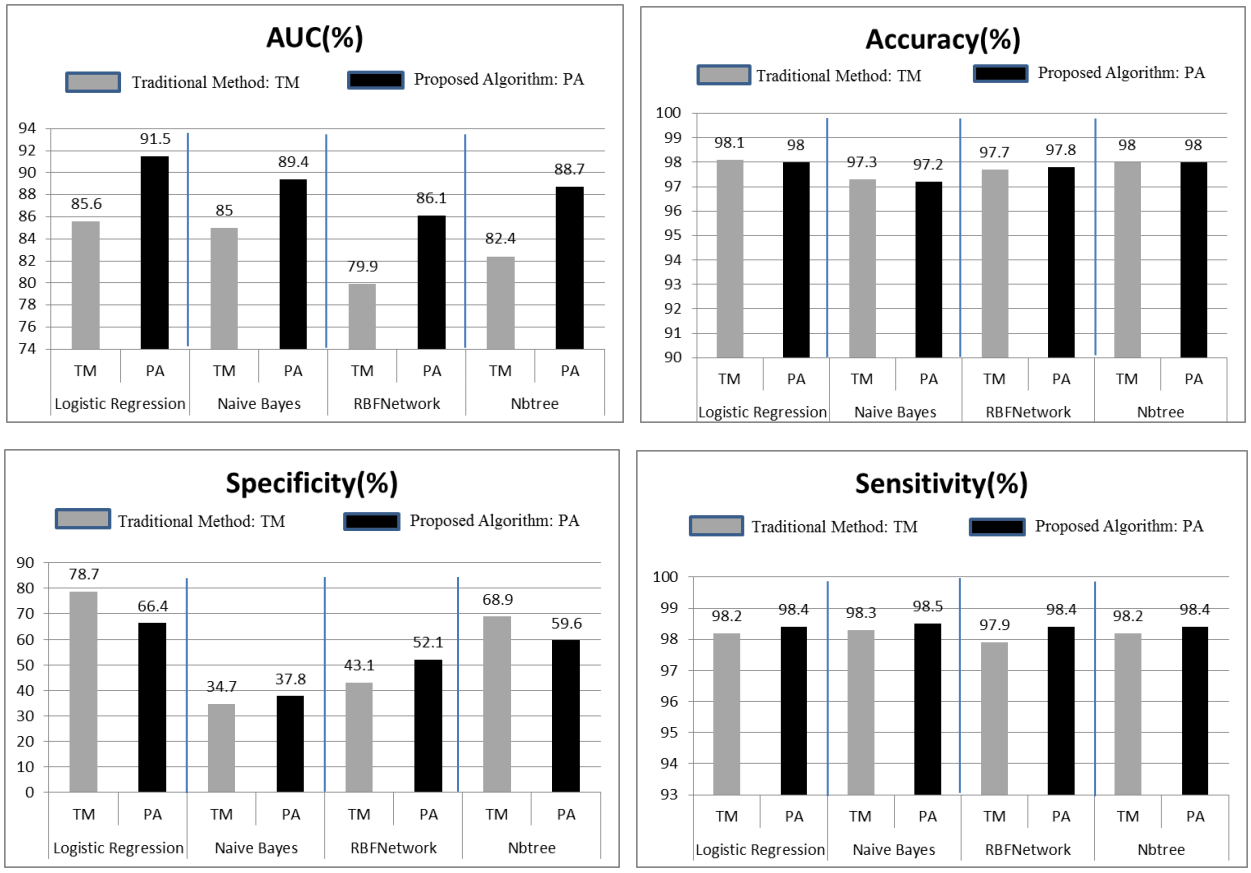


Figure 24. Overall comparison of proposed method and traditional method

5.6.2 Evaluation for Classification Performance

In Chapter 2, literature review has listed the classification results of some remarkable participants of 2009 UC San Diego data mining contest who attempted the data set used in this research. The first evaluation indicator is AUC rate. I selected [Yang, Z., Cao, S., & Yan, B., 2011] and [Yang, H. & King I., 2009]'s result as the comparison model. From Figure 24, the lowest value of AUC in the average result of twelve clusters is RBFNetwork (86.1%), and the range is [86.1%, 91.5%]. Cao, Yang, & Yan's LDA method presents the best result with the AUC rate of 87.1%. Three algorithms such as Logistic Regression, Naive Bayes and NBtree perform have better AUC results than Cao, Yang, & Yan's LDA method. Yang and King's ensemble method gets the best AUC rate of 89.0%, which is lower than my Logistic Regression

(91.5%) and Naive Bayes (89.4%). Throughout the overall AUC performance, it can be concluded that the meta-cluster with k-means algorithm performs best in terms of AUC performance.

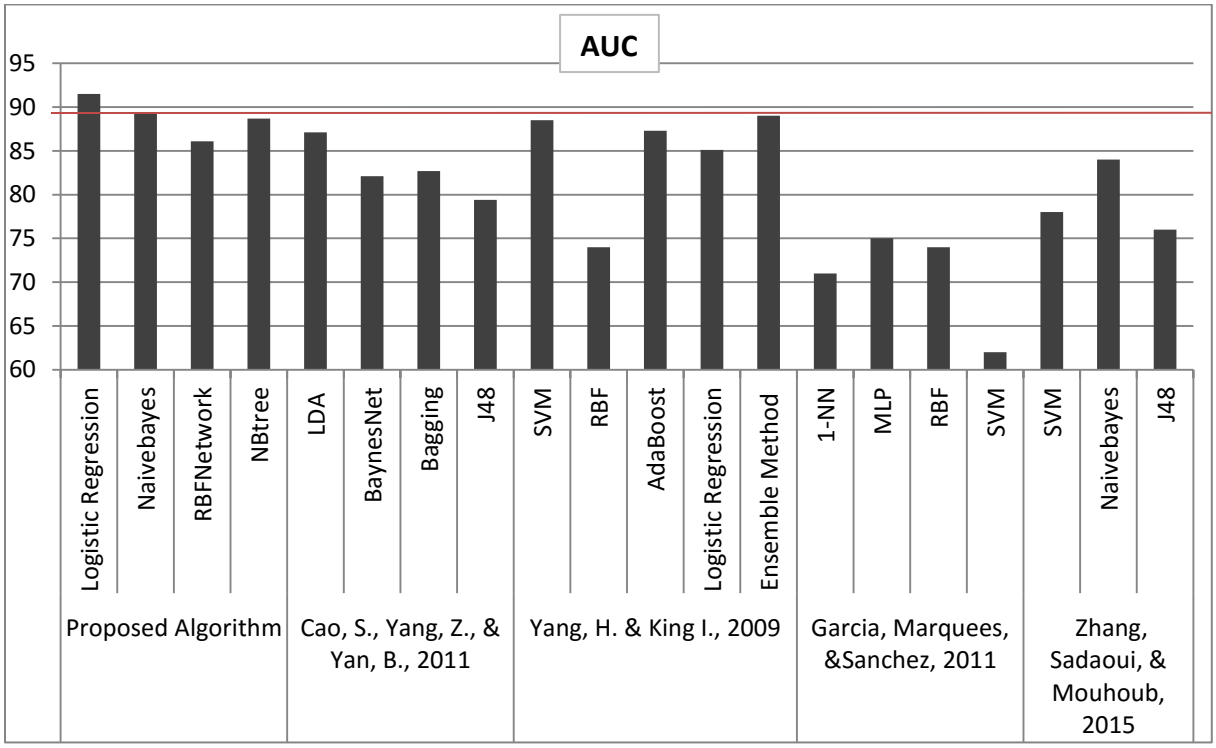


Figure 25. AUC comparison of proposed method and other researchers' model

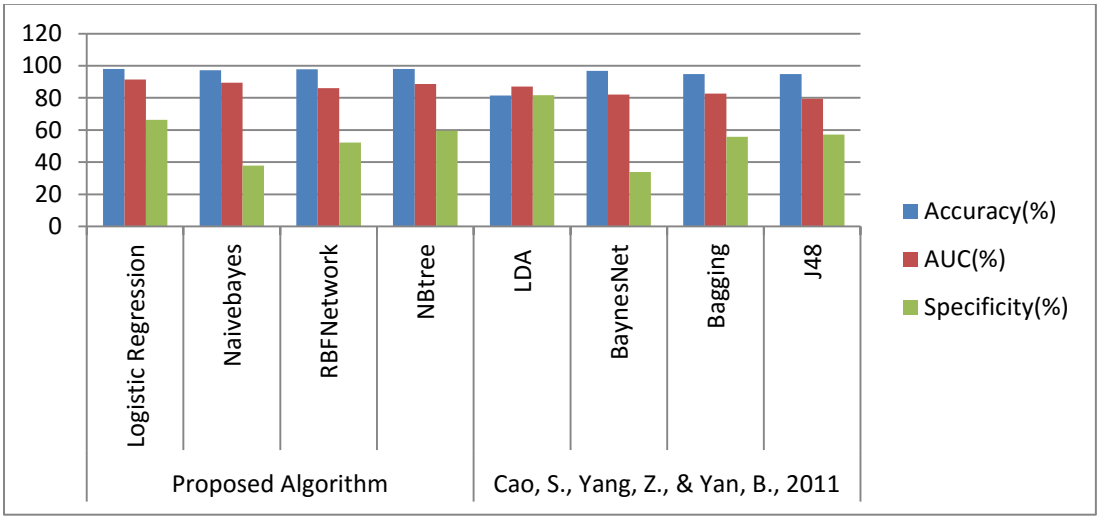


Figure 26. Overall comparison with [Yang, Z., Cao, S., & Yan, B., 2011]'s model

Next I will evaluate the overall performance of accuracy, specificity and AUC. The proposed meta-cluster with K-means algorithm and Cao, Yang, & Yan's LDA method are carefully compared. From Figure 25 we can observe that Logistic Regression, Naive Bayes, RBFNetwork and NBtree scored excellent accuracy within the range of [97.2%, 98.0%] and AUC within the range of [86.1%, 91.5%]. Cao, Yang, and Yan presented the model which showed the accuracy in the range of [81.5%, 96.9%] and AUC in the range of [79.4%, 87.1%]. It can be seen that our method is much higher than [Yang, Z., Cao, S., & Yan, B., 2011] with an average difference of 5%. However, Naive Bayes method exhibits low specificity compared to other seven algorithms. It may be due to the fact that the specificity is possibly sacrificed since greater weight has been applied to the negative class in the data set during classification. And the LDA method gets the best specificity of 81.6%, but it sacrifices the total accuracy rate of 81.5%. For my method, the result presents a good specificity within the range of [52.1%, 66.4%] except Naive Bayes classifier, while the accuracy and AUC perform very well. Naive Bayes classifier is more easily affected by the highly imbalanced data set. However it does not affect the overall performance of my proposed method. Therefore, this meta-cluster with K-means algorithm can minimize type-II errors and keep good AUC and accuracy performance.

6. Conclusion

6.1 Summary

In this thesis, the new integrated method based on meta-learning presents an effective E-Commerce transaction anomaly detection framework. It can effectively resolve real-world classification problems with data imbalance. A literature review of the related anti-money laundering detection, E-Commerce transaction anomaly detection model in the industry and the most salient ongoing challenges in data mining classification projects is presented. Then the different classification methods applied in this thesis are introduced. After that, the proposed meta-cluster with K-means algorithm is discussed. The improved K-means algorithm can be used to generate a group of clusters which has a set of similar instances from the E-commerce transaction anomaly data set. In the classification phase, the effectiveness of the proposed method is validated using a real-world E-Commerce transaction anomaly detection classification data set. The proposed method produces a better prediction than the traditional methods by a

considerable margin. The discussion on how the proposed method is different from others and the major factor that enables the proposed model to excel in the classification task is provided. The comparison to the researches in the literature clearly shows that the proposed meta-cluster with K-means algorithm can get an excellent performance in terms of AUC and accuracy. At the same time it can minimize type-II errors.

The proposed Meta-cluster with K-means algorithm has broad applicability. By comparing the previous results of the experimental data, it can be concluded that the proposed method has good performance with E-commerce highly imbalanced data set in anomaly detection, identification, and determination. Here I would like to mention that, according to the characteristics of meta-cluster with K-means algorithm, its functions can be extended to more applications for the data highly imbalanced problem, such as disease monitoring and prevention, risk prediction of financial products, natural disaster anomaly detection, and etc. And I believe my thesis would bring a lot of inspiration and new solutions to the experts in the industrial sector.

6.2 Future Work

This thesis shows that meta-cluster with K-means method is not only a very effective meta-learning based classification method, but also adaptable to semi-supervised problems. However, this semi-supervised method is a complex process. The performance of clustering results will directly affect the accuracy of anomaly detection model. When researchers face the high imbalanced data set, the core work is to effectively and efficiently choose correct seed number for the meta-cluster with K-means algorithm. This is a cumbersome process. How to quickly and accurately select the seed number will become the most important topic and research direction in the future work.

References

1. Amazon annual reports, 2013, http://phx.corporate-ir.net/phoenix.zhtml?c=97664&p=irol-reportsannual.

2. Caruana, R., Elhawary, M., Nguyen, N., & Smith, C., "Meta Clustering", 2006, Cornell University, Ithaca, New York.

3. Danielsson, P. 1980, "Euclidean Distance Mapping", Computer Graphics And Image Processing, volume 14, PP. 227-248.

4. Goldman Sachs annual reports,2014, http://www.goldmansachs.com/

5. Financial Action Task Force on Money Laundering Report, 1990-1991.

6. Ionita, I., & Ionita, L., 2001, "A decision support based on data mining in e-banking", DOI: 10.1109/RoEduNet.2011.5993710.

7. Jurek, A., Bi, Y., Wu, S., & Nugent, C., 2012, "Classification by Cluster Analysis: A New Meta-Learning Based Approach", School of Computing and Mathematics University of Ulster, MCS 2011, LNCS 6713, pp. 259–268.

8. Lee, W. K., Stolfo, S. J., & Mok, K. W., 1999, "A data mining framework for building intrusion detection models", Proceeding of the IEEE Symposium on Security and Privacy. California: IEEE Computer Socidy Press, pp. 120-132

9. Li, H., Zhang, N., & Bao, L., 2006, “Using an improved clustering method to detect anomaly activities”, Wuhan University Journal of Natural Sciences, volume 11(6), pp. 1814-1818

10. MacQueen, J.B., 1967, "Some methods for classification and analysis of multivariate observations", Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, University of California Press, Berkeley, pp. 281–297.

11. Scarle, S., Arnab, S., Dunwell, I., Petridis, P., Protopsaltis, A. and de Freitas, S. 2012, “E-commerce transactions in a virtual environment: virtual transactions”, Electronic Commerce Research, volume 12 (3), pp. 379-407.

12. Sugumaran, V. 2001, “Intelligent Support Systems: Knowledge Management”, Oakland University, pp. 100.

13. Witten, I.H., Frank, E., & Hall, M.A., 2011, “Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann (Third Edition)”, ISBN 978-0-12-374856-0.

14. Yang, H. & King I., 2009, “Ensemble Learning for Imbalanced E-commerce Transaction Anomaly Classification”, ICONIP, LNCS 5863, pp. 866–874.

15. Yang, Z., Cao, S. & Yan, B., 2011, “Using Linear Discriminant Analysis and Data Mining Approaches to Identify E-commerce Anomaly”, IEEE, ICNC, pp. 2406-2410.

16. Khosravani, R., 2010, “A Linear Approximation to a Neural Network Model for E-Commerce Anomaly Detection”, Annual International Conference on Computer Science Education: Innovation & Technology, ISBN: 978-981-08-7466-7.

17. Ho, L. & Papavassiliou, S., 2000, “Network and Service Anomaly Detection in Multi-Service Transaction-based Electronic Commerce Wide Area Networks”, IEEE Computer Society, ISCC, pp. 291-296.

18. Wirelessintelligence.com. 2008. Deployment Tracker Mobile Money Live. (Accessed 06 February 12).

19. Bonsoni.com. 2011. Research shows mobile phone payment double by 2013

20. Sherly, k. k, & Nedunchezhian, R., 2010, BOAT adaptive Credit Card Fraud Detection System [C] // IEEE international Conference on Computational Intelligence and Computing Research.

21. Lee, M., Ham, S., & Jiang, Q., 2010, “E-commerce Transaction Anomaly Classification”, Statistics Department, Stanford University.

22. Quan, Y., Jia, Y., Li, S.D., & Han, W.H., 2013, “Anomaly Detection on E-commerce Transactions Log based on Co-occurrence Matrix”, National University of Defense Technolog.

23. Patulea, C., Peace, R., & Green, J., 2010, “CUDA-accelerated genetic feedforward-ANN training for data mining”, School of Systems and Computer Engineering, Carleton University, High Performance Computing Symposium (HPCS2010), Journal of Physics: Conference Series 256 (2010) 012014, doi:10.1088/1742-6596/256/1/012014.

24. Phua, C., Alahakoon, D. & Lee, V., “Minority Report in Fraud Detection: Classification of Skewed Data,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 50-59, 2004.

25. Lu, Q. & Ju, C., 2011, “Research on Credit Card Fraud Detection Model Based on Class Weighted Support Vector Machine”, Journal of Convergence Information Technology, Volume 6, doi:10.4156/jcit.vol6.issue1.8.

26. Sen, K. S., & Dash, S., 2013, “Meta Learning Algorithms for Credit Card Fraud Detection”, International Journal of Engineering Research and Development, e-ISSN: 2278-067X, p-ISSN: 2278-800X, Volume 6, Issue 6.

27. Minegishi, T., & Niimi, A., 2011, “Detection of Fraud Use of Credit Card by Extended VFDT”, Internet Security (WorldCIS), E-ISBN: 978-0-9564263-7-6.

28. Kundu, A., Panigrahi, S., Sural, S., & Majumdar, A. K.., 2009, “BLAST-SSAHA Hybridization for Credit Card Fraud Detection”, Dependable and Secure Computing, ISSN: 1545-5971

29. Wang, N., & JU, C.H., 2009, “Research on Credit Card Fraud Detection Model Based on Similar Coefficient Sum”, 2009 First International Workshop on Database Technology and Applications, ISBN: 978-0-7695-3604-0.

30. Senator, T.E., Goldberg, H.G., & Wooton, J., 1995, “The financial crimes enforcement network AI system(FAIS)--Identifying potential money laundering from reports of large cash transactions[J]”, AI Magazine, 16(4):21-39.

31. Duyne, P.C., & Miranda, H.D., 1999, “The emperor's clothes of disclosure:Hot money and suspect disclosures[J], Crime,Law & Social Change, 31(3):245-271.

32. Knorr, E. M., & Ng, R.T., 1998, “Algorithms for mining Distance-based outliers in large Datasets” [C] // Proceeding of 24th International Conference on Very Large Data Bases, ISBN:1-55860-566-5.

33. Stofella, P., 1997, “The DBInspector Project” [J] // Proceedings of the IEEE international workshop on research issues in data engineering, AI Magazine, (5):73-75.

34. Kingdon, J., & Feldman, K.S., 2002, “Data monitoring and analysis system for bank transactions.constructs aggregate profiles of received data and investigates t0 identify its characteristic patterns of behavior[C]”, SEARCHSPACE LTD (SEAR-Non-standard).

35. Kingdon, J., 2004, “AI fights money laundering”, IEEE Intelligent Systems, 19(3):87-89.

36. Breunig, M.M., Kriegel, H.P., Ng, R.T., & Sander, J., 2000, "LOF: Identifying Density-based Local Outliers", Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. doi:10.1145/335191.335388. ISBN 1-58113-217-4.

37. Duda, R.O., & Hart, P.E., 2001, “Pattern classification and scene analysis”, New York: Wiley, ISBN: 978-0-4714-2977-7.

38. Ngai, E., Xiu, L., & Chau, D., 2009, “Application of data mining techniques in customer relationship management: A literature review and classification”, Expert Systems with Applications, 2592–2602.

39. Han, J., & Kamber, M., 2001, “Data Mining: Concepts and Techniques”, San Francisco: Morgan Kaufmann.

40. Michie, D., Spiegelhalter, D.J., & Taylor, C.C., 1994, “Machine Learning, Neural and Statistical Classification”, Ellis Horwood Upper Saddle River, NJ, USA, ISBN: 0-13-106360-X

41. Arning, A. Agrawal, R. & Raghavan, P., 1996, “A Linear Method for Deviation Detection in Large Databases”, KDD-96 Proceedings.

42. Ye, N., 2003, “The handbook of Data Mining. New Jersey: Lawrence Erlbaum Associates”.

43. Pearl, J., 1985, “BAYESIAN NETWORKS: A MODEL OF SELF-ACTIVATED MEMORY FOR EVIDENTIAL REASONING”, Report No. CSD-850017.

44. Alfonso, P., Rafael, J., & Elena, G., 2011, “Data Mining: Machine Learning and Statistical Techniques”, University of the Balearic Islands, ISBN 978-953-307-154-1.

45. Larose, D.T., 2005, “Discovering Knowledge in Data: An Introduction to Data Mining”, Hoboken, N.J: Wiley-Interscience.

46. Hand, D.J., & Yu, K., 2001, “Idiot’s Bayes – not so stupid after all? International Statistical Review”, 385-398, DOI: 10.1111/j.1751-5823.2001.tb00465.x.

47. Fonseca, M.J., Jorge, J.A., 2004, “NB-Tree: An Indexing Structure for Content-Based Retrieval in Large Databases," INESC-ID/IST/Technical University of Lisbon R. Alves Redol, 9, 1000-029 Lisbon.

48. Grigoris, K., 2002, “Modeling Private Firm Default: PFirm”, Business Analytic Solutions.

49. Broomhead , D. S., & Lowe, D., 1988, “Multivariate functional interpolation and adaptive networks”, Complex systems, 2:321- 355.

50. Zighed, D., Rakotomalala, R., & Feschet. F., 1997, “Optimal Multiple Intervals Discretization of Continuous Attributes for Supervised Learning”, Proceedings of the 3rd International Conference on Knowledge Discovery in Databases (KDD-1997), 295-298.

51. Potzelberger, K., & Felsenstein, K., 1993, “On the Fisher Information of Discretized Data”, The Journal of Statistical Computation and Simulation, 46(3 & 4), 125-144.

52. Chmielewski, M.R., & Grzymala-Busse, J.W. 1994, “Global Discretization of Continuous Attributes as Preprocessing for Machine Learning”, Proceedings of the 3rd International Workshop on Rough Sets and Soft Computing, 294-301.

53. Fawcett, T., 2006, “An introduction to ROC analysis”, *Pattern Recognition Letters*, 861–874.

54. Ng, A. Y., 2004, “Feature selection”, L1 vs. L2 regularization, and rotational invariance. ICML.

55. Yang, Y., & Pedersen, J. O., 1997, “A comparative study on feature selection in text categorization”, ICML.

56. Mladenic, D., & Grobelnik, M., 1999, “Feature selection for unbalanced class distribution and Na ïve Bayes”, ICML.

57. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C.J., 1984, “Classification and regression trees”, Wadsworth and Brooks.

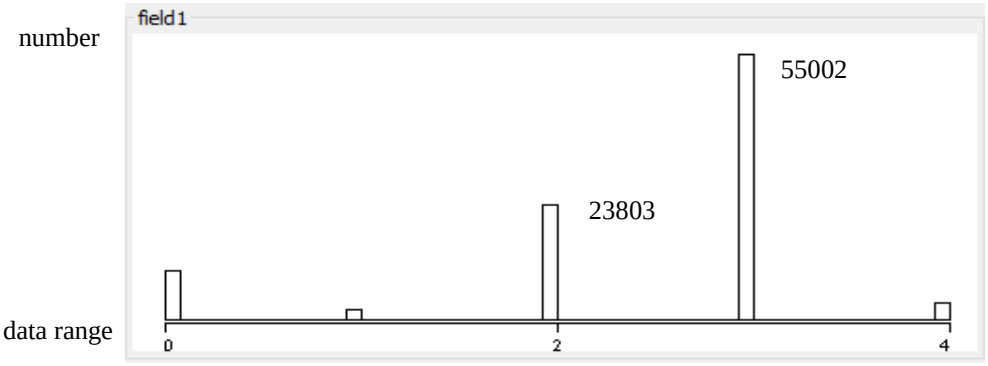
58. Novakovic, J., 2009, “Using Information Gain Attribute Evaluation to Classify Sonar Targets”, 17th Telecommunications forum TELFOR.

59. Garcia, Marquees, & Sanchez, 2011, “Improving Risk Predictions by Preprocessing Imbalanced Credit Data”, Dep. Computer Languages and Systems, Institute of New Imaging Technologies, Spain.

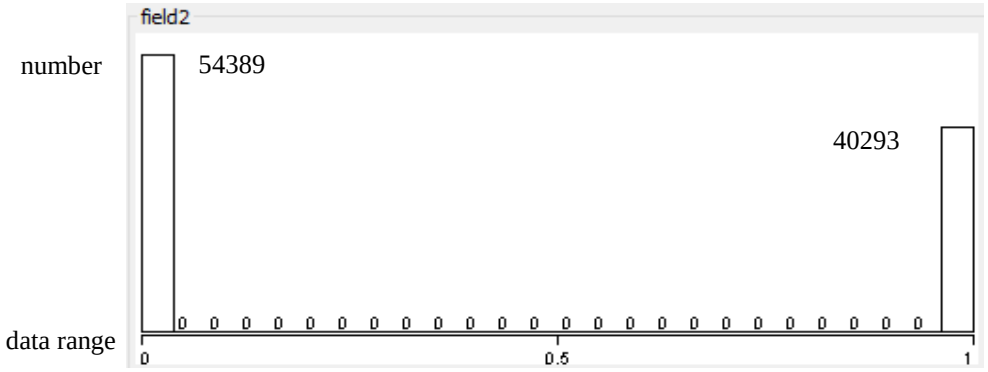
60. Zhang, Sadaoui, & Mouhoub, 2015, “An Empirical Analysis of Imbalanced Data Classification”, Department of Computer Science, University of Regina, SK, Canada, doi:10.5539/cis.v8n1p151.

Appendix A. Additional Figures and Tables

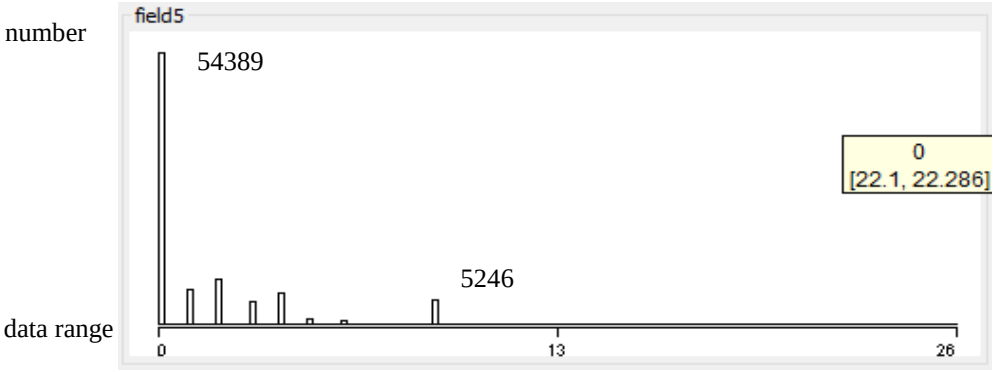
A.1 Characterization of Modeling Variables



(A) Variable "field1"



(B) Variable "field2"



(C) Variable "field5"

Figure 27. Histogram for variable"field1", "field2", and ""field5"

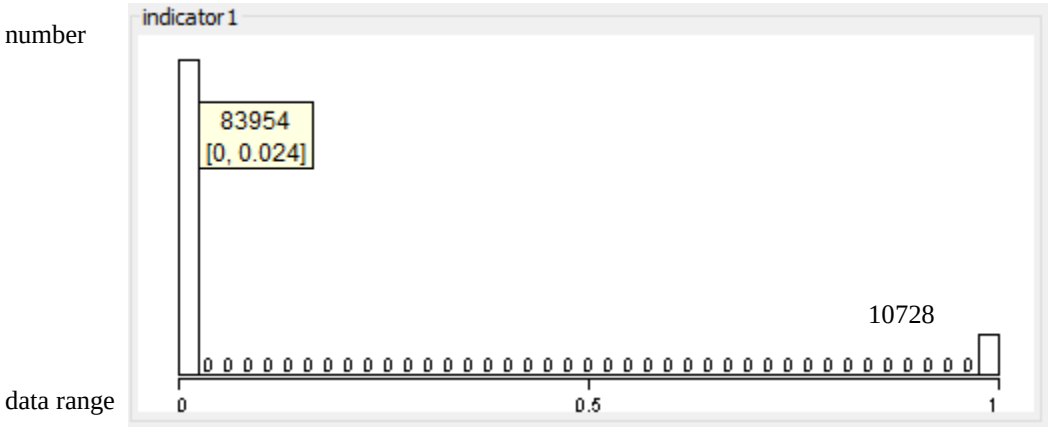
Values	0	1	2	3	4
Records	10147	2159	23803	55002	3571
Percentage	10.70%	2.30%	25.10%	58.10%	3.80%
Anomaly rate	1.40%	1.30%	1.40%	1.30%	24.40%

(A) Variable "field1

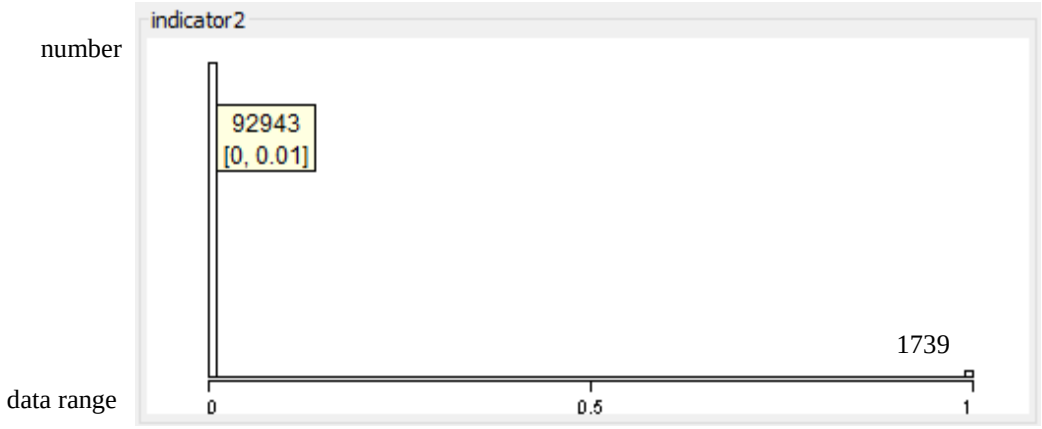
Values	0	1
Records	54389	40293
Percentage	57.4%	42.6%
Anomaly rate	2.3%	2.1%

(B) Variable "field2"

Table 8. Distribution of "field1", "field2", and ""field5"



(A) Variable "indicator1"



(B) Variable "indicator2"

Figure 28. Histogram for variable"indicator1"and "indicator2"

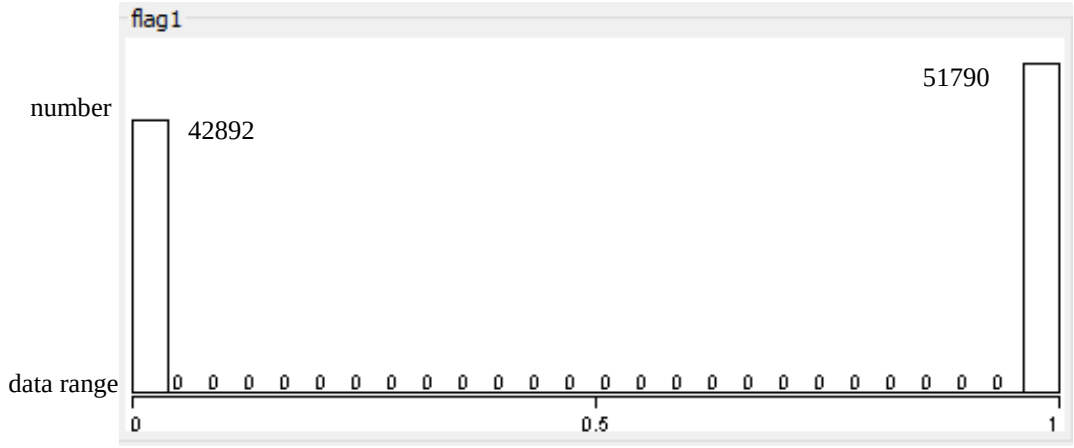
Values	0	1
Records	83954	10728
Percentage	88.7%	11.3%
Anomaly rate	2.1%	2.9%

(A) Variable "indicator1"

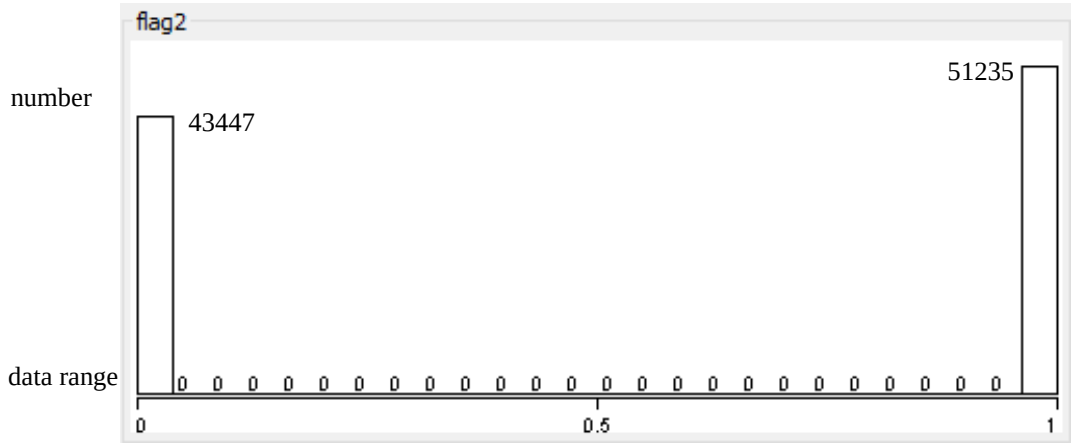
Values	0	1
Records	92943	1739
Percentage	98.2%	1.8%
Anomaly rate	2.2%	2.5%

(B) Variable "indicator2"

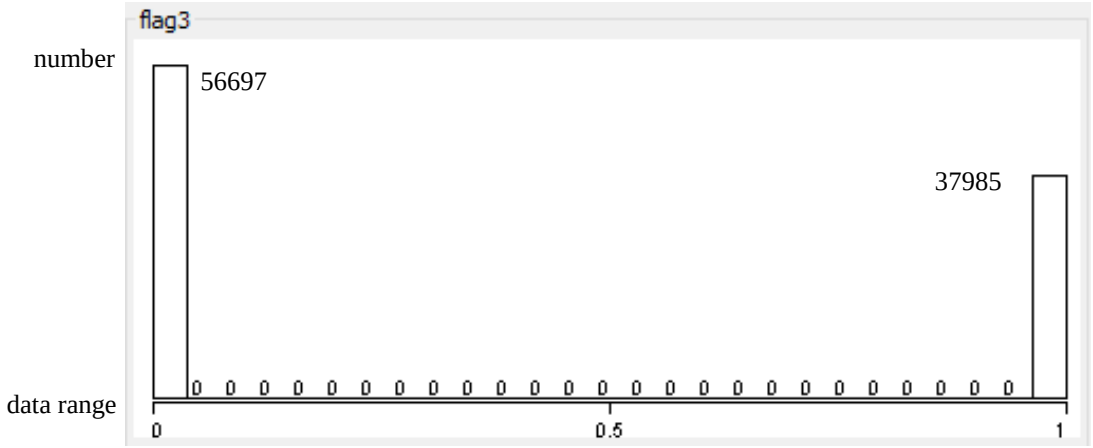
Table 9. Distribution of "indicator1"and "indicator2"



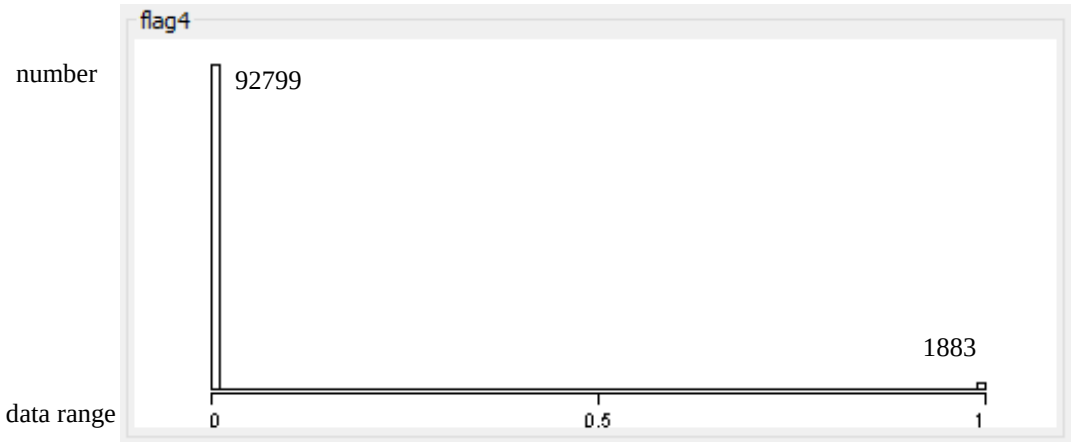
(A) Variable "flag1"



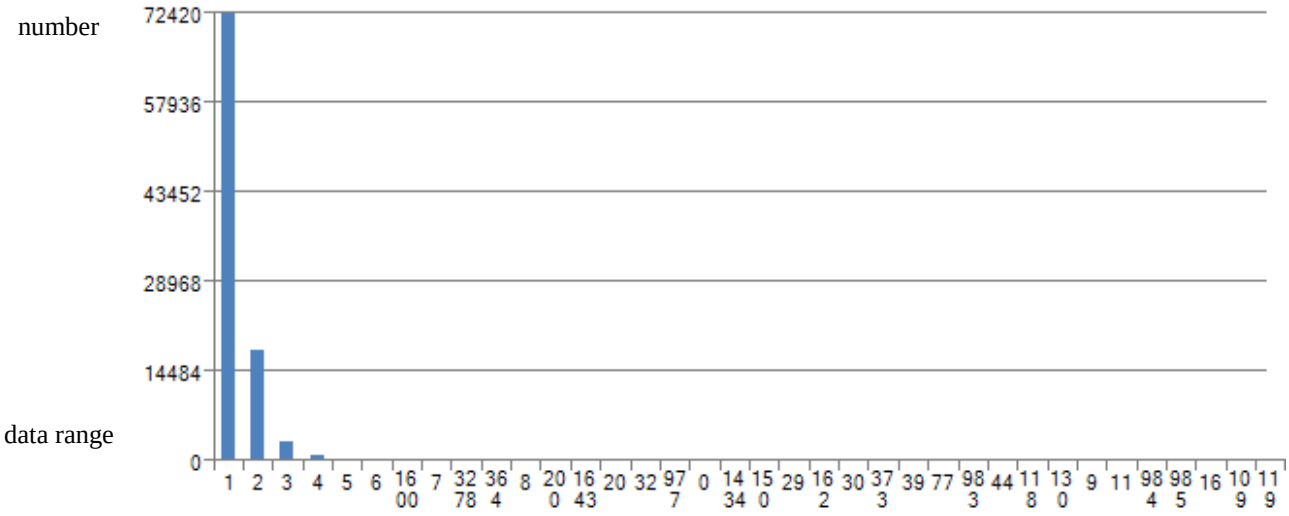
(B) Variable "flag2"



(C) Variable "flag3"



(D) Variable "flag4"



(E) Variable "flag5"

Figure 29. Histogram for variable"flag1", "flag2", "flag3", "flag4" and "flag5"

Values	0	1
Records	42892	51790
Percentage	45.3%	54.7%
Anomaly rate	2.2%	2.4%

(A) Variable "flag1"

Values	0	1
Records	43447	51235
Percentage	45.9%	54.1%
Anomaly rate	2.1%	2.3%

(B) Variable "flag2"

Values	0	1
Records	56697	37985
Percentage	59.9%	40.1%
Anomaly rate	1.3%	3.6%

(C) Variable "flag3"

Values	0	1
Records	92799	1883
Percentage	98.0%	2.0%
Anomaly rate	2.1%	6.3%

(D) Variable "flag4"

Table 10. Distribution of "flag1", "flag2", "flag3" and "flag4"

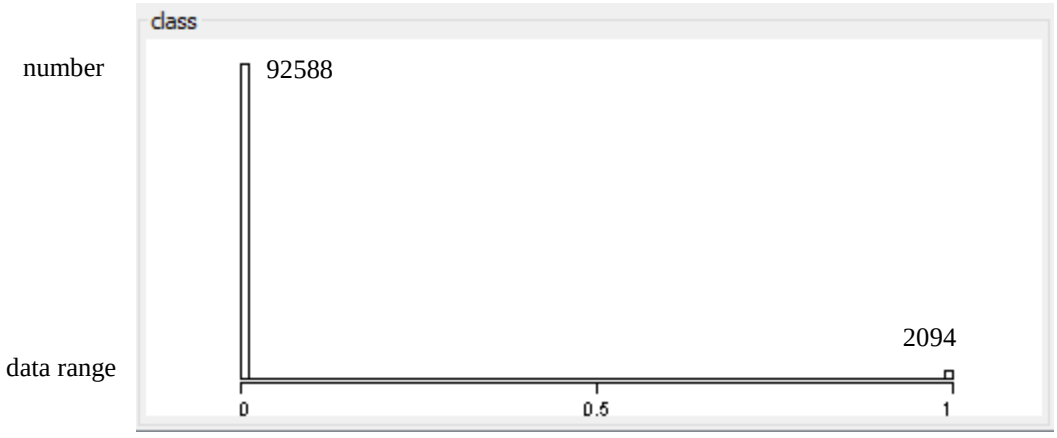


Figure 30. Histogram for variable "class"

A.2 Results of Information Gain Variable Ranking

Cluster 1		
Attribute	**Ranking**	**Score**
hour1	1	0.00786748
flag3	2	0.00665177
field3	3	0.00336876
zip1	4	0.0020328
flag5	5	0.00182446
amount	6	0.0017116
field5	7	0.00143599
state	8	0.00113284
field4	9	0.00060005
domain1	10	0.00053644
flag2	11	0.0005198
flag1	12	0.00035976
field2	13	0.00008832
field1	14	0.00008758
indicator2	15	0.00004285
flag4	16	0.000007
indicator1	17	0.00000524
hourdiff	18	0.00000256

(*The score of variable "hour2" is 0.00000147)*

Cluster 2		
Attribute	**Ranking**	**Score**
hour1	1	0.019513
zip1	2	0.004749
field3	3	0.004427
flag3	4	0.004237
field4	5	0.003599
domain1	6	0.002593
flag5	7	0.002156
state	8	0.001695
amount	9	0.00165
field5	10	0.001017
field2	11	0.000695
indicator1	12	0.000458
field1	13	0.000408
indicator2	14	8.81E-05
flag4	15	3.16E-05
hourdiff	16	1.27E-05
flag2	17	3.69E-06
flag1	18	6.32E-08

(The score of variable "hour2" is 2.51E-05)

Cluster 1 selected attributes: amount, hour1, state, zip1, field3, field5, flag3, and flag5.

Cluster 2 selected attributes: amount, hour1, state, zip1, domain1, field3, field4, field5, flag3, and flag5.

Cluster 3		
Attribute	**Ranking**	**Score**
field1	1	0.021378084
hour1	2	0.009641386
flag3	3	0.005514448
field4	4	0.00497875
field3	5	0.003073697
zip1	6	0.001918778
flag4	7	0.001713996
state	8	0.001125916
field5	9	0.001023847
amount	10	0.000866779
flag5	11	0.000589557
domain1	12	0.000553222
indicator2	13	0.000163764
indicator1	14	0.000155348
flag1	15	0.000049209
flag2	16	0.000026612
hourdiff	17	0.000004172
field2	18	0.000000954

(The score of variable "hour2" is 0.00000236)

Cluster 4		
Attribute	**Ranking**	**Score**
field1	1	0.0292361
flag3	2	0.0080652
field3	3	0.006832
hour1	4	0.0057462
field4	5	0.0034829
zip1	6	0.003462
domain1	7	0.0018271
flag4	8	0.0016777
amount	9	0.0016
flag1	10	0.0012346
hourdiff	11	0.0011425
field5	12	0.0003715
state	13	0.0002986
indicator2	14	0.0002255
field2	15	0.0001777
indicator1	16	1.551E-05
flag5	17	6.96E-06
flag2	18	6.85E-06

(The score of variable "hour2" is 0.0008425)

Cluster 3 selected attributes: amount, hour1, state, zip1, field1, field3, field4, field5, flag3, and flag4

Cluster 4 selected attributes: amount, hour1, hourdiff, zip1, field1, domain1, flag1, field3, field4, flag3, and flag4

Cluster 5		
Attribute	**Ranking**	**Score**
field1	1	0.04450474
amount	2	0.01792496
flag5	3	0.01169368
hour1	4	0.01007308
zip1	5	0.00920212
flag3	6	0.00680713
field4	7	0.00528456
field5	8	0.00408101
state	9	0.00345605
field3	10	0.00329551
flag4	11	0.00271759
flag1	12	0.00197408
domain1	13	0.00122266
field2	14	0.00080992
flag2	15	0.00050976
hourdiff	16	0.00043335
indicator1	17	0.00021753
indicator2	18	0.00000237

(The score of variable "hour2" is 0.00024896)

Cluster 6		
Attribute	**Ranking**	**Score**
field3	1	0.055959688
zip1	2	0.015345187
field4	3	0.006321636
field1	4	0.006139976
flag5	5	0.00471437
flag3	6	0.004465764
state	7	0.004266033
amount	8	0.00272139
domain1	9	0.002216346
field5	10	0.001810403
hourdiff	11	0.001660277
flag1	12	0.001233087
hour1	13	0.001011045
flag4	14	0.000950446
indicator2	15	0.000189666
field2	16	0.000164309
indicator1	17	6.60787E-05
flag2	18	1.03E-08

(The score of variable "hour2" is 0.000965214)

Cluster 5 selected attributes: amount, hour1, state, zip1, field1, flag1, domain1, field3, field4, field5, flag3, flag4, and flag5

Cluster 6 selected attributes: amount, hour1, hourdiff, state, zip1, field1, domain1, flag1, field3, field4, field5, flag3, and flag5

Cluster 7		
Attribute	**Ranking**	**Score**
field3	1	0.0531879
zip1	2	0.0201362
field1	3	0.0076736
amount	4	0.0049654
state	5	0.0043669
field4	6	0.0043569
field5	7	0.0029001
flag3	8	0.0025636
hour1	9	0.001916
flag5	10	0.001235
domain1	11	0.0006516
flag4	12	0.0001432
indicator2	13	0.0001287
hourdiff	14	0.000102
field2	15	0.0000717
flag2	16	0.0000413
flag1	17	0.0000319
indicator1	18	0.0000298

(The score of variable "hour2" is 0.000087)

Cluster 8		
Attribute	**Ranking**	**Score**
field3	1	0.483572575
field1	2	0.109295816
zip1	3	0.08423172
field4	4	0.028974695
flag5	5	0.020937558
amount	6	0.015361589
flag3	7	0.012724533
state	8	0.011144217
hour1	9	0.004305125
field5	10	0.004037672
hourdiff	11	0.003270264
domain1	12	0.00250687
indicator1	13	0.002027886
flag4	14	0.00147493
flag1	15	0.000551059
field2	16	0.000409471
flag2	17	4.89383E-06
indicator2	18	4.83E-09

(The score of variable "hour2" is 0.002895146)

Cluster 7 selected attributes: amount, hour1, state, zip1, field1, field3, field4, field5, flag3, and flag5

Cluster 8 elected attributes: amount, state, zip1, hour1, hourdiff, field1, field3, field4, field5, flag3, and flag5

Cluster 9		
Attribute	**Ranking**	**Score**
field3	1	0.1523592
field1	2	0.05101158
zip1	3	0.03268566
amount	4	0.02528034
state	5	0.01122358
flag5	6	0.00992411
field4	7	0.00680149
flag3	8	0.00472209
hour1	9	0.00449891
hourdiff	10	0.00393454
field5	11	0.0028194
domain1	12	0.00261825
indicator1	13	0.00069039
flag1	14	0.00060433
flag2	15	0.00058905
field2	16	0.00043456
flag4	17	0.0002762
indicator2	18	0.00000544

(The score of variable "hour2" is 0.00040357)

Cluster 10		
Attribute	**Ranking**	**Score**
field1	1	0.04397811
amount	2	0.03278108
zip1	3	0.00728985
hour1	4	0.00714151
field4	5	0.00707418
state	6	0.00425334
flag5	7	0.0036666
field5	8	0.00262661
flag3	9	0.00152598
domain1	10	0.00109696
field2	11	0.0007334
field3	12	0.00065795
flag2	13	0.00057534
hourdiff	14	0.00032773
indicator1	15	0.00030733
flag4	16	0.00007799
flag1	17	0.00001305
indicator2	18	0.00000971

(The score of variable "hour2" is 0.00018764)

Cluster 9 Selected attributes: amount, hour1, hourdiff, state, zip1, field1, domain1, field3, field4, field5, flag3, and flag5

Cluster10 Selected attributes: amount, hour1, state, zip1, field1, domain1, field4, field5, flag3, and flag5

Cluster 11		
Attribute	**Ranking**	**Score**
field3	1	0.0432233
zip1	2	0.0138844
field1	3	0.010418
field4	4	0.0056067
amount	5	0.0040491
state	6	0.0038459
flag1	7	0.0022798
flag5	8	0.002136
hour1	9	0.0020135
flag3	10	0.001721
field5	11	0.0011736
domain1	12	0.0005403
flag4	13	0.0002992
indicator2	14	0.0001408
field2	15	0.0001259
hourdiff	16	0.0001197
indicator1	17	0.0000499
flag2	18	0.0000166

(The score of variable "hour2" is 0.00000236)

Cluster 12		
Attribute	**Ranking**	**Score**
field3	1	0.0355842
zip1	2	0.0159622
field1	3	0.0094094
amount	4	0.0049944
field4	5	0.0042571
state	6	0.0042301
flag5	7	0.0022257
hour1	8	0.0011694
flag4	9	0.0009048
field5	10	0.0008625
flag3	11	0.0007885
domain1	12	0.0005429
hourdiff	13	0.0004084
indicator2	14	0.0002871
flag1	15	0.0002066
indicator1	16	0.0001216
field2	17	0.0000562
flag2	18	0.0000427

(The score of variable "hour2" is 0.0008425)

Cluster 11 Selected attributes: amount, hour1, state, zip1, field1, flag1, field3, field4, field5, flag3, and flag5

Cluster 12 Selected attributes: amount, hour1, hourdiff, state, zip1, domain1, field1, field3, field4, field5, flag3, flag4, and flag5

traditional method		
Attribute	**Ranking**	**Score**
field1	1	0.02419278
hour1	2	0.01705351
amount	3	0.00655684
flag3	4	0.00441007
field3	5	0.0032206
field4	6	0.00309467
flag5	7	0.00187188
zip1	8	0.0011524
flag4	9	0.00077323
hourdiff	10	0.00056212
flag1	11	0.00043858
field5	12	0.00041235
state	13	0.00037063
indicator1	14	0.00021101
domain1	15	0.00016681
flag2	16	0.00004364
field2	17	0.00001207
indicator2	18	0.00000606

(The score of variable "hour2" is 0.00023614)

Traditional method **selected attributes:** field1, hour1, hourdiff, amount, flag3, field3, field4, flag5, zip1, and flag4

Table 11. Results of Information Gain Variable Ranking

A.3 Result of Classification Confusion Matrix (A)

	Logistic Regression Confusion Matrix		**Naviebayes Confusion Matrix**		**RBF Confusion Matrix**		**NBtree Confusion Matrix**	
	Classified as **0**	Classified as **1**	Classified as **0**	Classified as **1**	Classified as **0**	Classified as **1**	Classified as **0**	Classified as **1**
Cluster 1	TP=6383	FP=20	TP=6386	FP=17	TP=6394	FP=9	TP=6400	FP=3
	FN=54	TN=20	FN=61	TN=13	FN=68	TN=6	FN=68	TN=6
Cluster 2	TP=5460	FP=17	TP=5468	FP=9	TP=5457	FP=20	TP=5459	FP=18
	FN=75	TN=15	FN=86	TN=4	FN=77	TN=13	FN=78	TN=12
Cluster 3	TP=10065	FP=41	TP=10017	FP=89	TP=10041	FP=65	TP=10034	FP=72
	FN=251	TN=58	FN=241	TN=68	FN=259	TN=50	FN=246	TN=63
Cluster 4	TP=9083	FP=28	TP=9071	FP=40	TP=9081	FP=30	TP=9078	FP=33
	FN=167	TN=40	FN=165	TN=42	FN=172	TN=35	FN=168	TN=39
Cluster 5	TP=10798	FP=31	TP=10515	FP=314	TP=10804	FP=25	TP=10773	FP=56
	FN=315	TN=82	FN=259	TN=138	FN=373	TN=24	FN=295	TN=102
Cluster 6	TP=7569	FP=9	TP=7559	FP=19	TP=7563	FP=15	TP=7571	FP=7
	FN=48	TN=6	FN=47	TN=7	FN=45	TN=9	FN=50	TN=4
Cluster 7	TP=7790	FP=0	TP=7743	FP=47	TP=7784	FP=6	TP=7785	FP=5
	FN=45	TN=8	FN=45	TN=8	FN=46	TN=7	FN=46	TN=7
Cluster 8	TP=3010	FP=78	TP=2937	FP=151	TP=2921	FP=167	TP=2966	FP=122
	FN=242	TN=222	FN=223	TN=241	FN=230	TN=234	FN=241	TN=223
Cluster 9	TP=8476	FP=17	TP=8159	FP=334	TP=8464	FP=29	TP=8463	FP=30
	FN=201	TN=32	FN=131	TN=102	FN=200	TN=33	FN=184	TN=49
Cluster 10	TP=5312	FP=37	TP=5270	FP=79	TP=5300	FP=49	TP=5325	FP=24
	FN=79	TN=58	FN=58	TN=69	FN=81	TN=46	FN=75	TN=52
Cluster 11	TP=8990	FP=2	TP=8929	FP=63	TP=8982	FP=10	TP=8979	FP=13
	FN=31	TN=14	FN=30	TN=15	FN=36	TN=9	FN=31	TN=14
Cluster 12	TP=9364	FP=8	TP=9307	FP=65	TP=9360	FP=12	TP=9360	FP=12
	FN=27	TN=14	FN=34	TN=7	FN=31	TN=10	FN=30	TN=11

A.3 Result of Classification Confusion Matrix (B)

	Logistic Regression Confusion Matrix		**Naivebayes Confusion Matrix**		**RBF Confusion Matrix**		**NBtree Confusion Matrix**	
	Classified as **0**	Classified as **1**	Classified as **0**	Classified as **1**	Classified as **0**	Classified as **1**	Classified as **0**	Classified as **1**
Proposed Algorithm	TP=92240	FP=288	TP=91361	FP=1227	TP=92151	FP=437	TP=92193	FP=395
	FN=1535	TN=569	FN=1380	TN=714	FN=1618	TN=476	FN=1512	TN=582

Table 12. Result of Classification Confusion Matrix

A.4 Result of Classification (A)

	Classifier	Number of correct classification	Number of incorrect classification	Accuracy(%)	AUC(%)	Sensitivity(%)	Specificity(%)
Cluster11	Logistic Regression	9,004	33	99.6	95.2	99.7	87.5
	Naviebayes	8,944	93	99.0	96.0	99.7	19.2
	RBFNetwork	8,991	46	99.5	87.7	99.6	47.4
	NBtree	8,993	44	99.5	94.5	99.7	51.9
cluster12	Logistic Regression	9,378	35	99.6	94.6	99.7	63.6
	Naviebayes	9,314	99	98.9	94.0	99.6	9.7
	RBFNetwork	9,370	43	99.5	90.4	99.7	45.5
	NBtree	9,371	42	99.5	96.6	99.7	47.8
cluster6	Logistic Regression	7,575	57	99.3	85.6	99.4	40.0
	Naviebayes	7,566	66	99.1	86.5	99.4	26.9
	RBFNetwork	7,572	60	99.2	87.3	99.4	37.5
	NBtree	7,575	57	99.3	84.4	99.3	36.4
cluster7	Logistic Regression	7,798	45	99.4	97.1	99.4	100.0
	Naviebayes	7,751	92	98.8	81.3	99.4	14.5
	RBFNetwork	7,791	52	99.3	83.0	99.4	53.8
	NBtree	7,792	51	99.3	86.9	99.4	58.3
cluster1	Logistic Regression	6,403	74	98.9	90.5	99.2	50.0
	Naviebayes	6,399	78	98.8	85.7	99.1	43.3
	RBFNetwork	6,400	77	98.8	87.9	98.9	40.0
	NBtree	6,406	71	98.9	89.6	98.9	66.7
cluster2	Logistic Regression	5,475	92	98.4	92.9	96.6	46.9
	Naviebayes	5,472	95	98.3	87.2	98.5	30.8
	RBFNetwork	5,470	97	98.3	87.1	98.6	39.4
	NBtree	5,471	96	98.3	85.0	98.6	40.0

A.4 Result of Classification (B)

	Classifier	Number of correct classification	Number of incorrect classification	Accuracy(%)	AUC(%)	Sensitivity(%)	Specificity(%)
cluster8	**Logistic Regression**	3,232	320	91.0	88.2	92.6	74.0
	Naviebayes	3,178	374	89.5	86.5	92.9	61.5
	RBFNetwork	3,155	397	88.8	85.9	92.7	58.4
	NBtree	3,189	363	89.8	88.7	92.5	64.6
cluster5	**Logistic Regression**	10,880	346	96.9	87.4	97.2	72.6
	Naviebayes	10,653	573	94.9	84.6	97.6	30.5
	RBFNetwork	10,828	398	96.5	81.4	96.7	49.0
	NBtree	10,875	351	96.9	89.9	97.3	64.6
cluster3	**Logistic Regression**	10,123	292	97.2	92.0	97.6	58.6
	Naviebayes	10,085	330	96.8	89.6	97.7	43.3
	RBFNetwork	10,091	324	96.9	85.8	97.5	43.5
	NBtree	10,097	318	96.9	83.7	97.6	46.7
cluster9	**Logistic Regression**	8,508	218	97.5	91.8	97.7	65.3
	Naviebayes	8,261	465	94.7	91.3	98.4	22.4
	RBFNetwork	8,497	229	97.4	85.5	97.7	53.2
	NBtree	8,512	214	97.5	85.1	97.9	62.0
cluster10	**Logistic Regression**	5,370	106	98.1	90.5	98.5	61.1
	Naviebayes	5,339	137	97.5	88.4	98.9	46.6
	RBFNetwork	5,346	130	97.6	86.6	98.5	48.4
	NBtree	5,377	99	98.2	81.0	98.6	68.4
cluster4	**Logistic Regression**	9,123	195	97.9	89.0	98.2	58.8
	Naviebayes	9,113	205	97.8	87.1	98.2	51.2
	RBFNetwork	9,116	202	97.8	82.7	99.6	53.8
	NBtree	9,117	201	97.8	84.0	98.2	54.2

Table 13. Results of classification

Appendix B. Coding Reference

B.1 Meta-cluster with K-means algorithm

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
import java.util.Vector;

public class KmeansCluster {

    /**Kmeans main process
    * @param Map<String, Map<String, Double>> DataminingContest2009
    * @param int K
    * @return Map<String,Integer>
    * @throws IOException
    */
    private Map<String, Integer> doProcess(
    Map<String, Map<String, Double>> DataminingContest2009, int K) {
     // TODO Auto-generated method stub

    //0. Read all variables of DataminingContest2009
    String[] testSampleNames = new String[DataminingContest2009.size()];
    int count = 0, tsLength = DataminingContest2009.size();
```

```
Set<Map.Entry<String, Map<String, Double>>> DataminingContest2009 =
DataminingContest2009.entrySet();

for(Iterator<Map.Entry<String, Map<String, Double>>> it =
allTestSampeleMapSet.iterator(); it.hasNext(); ){

 Map.Entry<String, Map<String, Double>> me = it.next();

testSampleNames[count++] = me.getKey();

 }


//1. The selection of initial point algorithm is uniformly separated

Map<Integer, Map<String, Double>> meansMap = getInitPoint(allDataminingContest2009,
K);

double [][] distance = new double[tsLength][K];//distance[i][j]


//2. Initializes the K cluster

int [] assignMeans = new int[tsLength];

Map<Integer, Vector<Integer>> clusterMember = new
TreeMap<Integer,Vector<Integer>>();

Vector<Integer> mem = new Vector<Integer>();

int iterNum = 0;

while(true){

 System.out.println("Iteration No." + (iterNum++) + "----------------------");


//3. To calculate the distance each point and each cluster cente

for(int i = 0; i < tsLength; i++){

 for(int j = 0; j < K; j++){

 distance[i][j] =
getDistance(DataminingContest2009.get(testSampleNames[i]),meansMap.get(j));

 }

 }
```

```
//4. Find the clustering center for each point
int[] nearestMeans = new int[tsLength];
for(int i = 0; i < tsLength; i++){
 nearestMeans[i] = findNearestMeans(distance, i);
 }

//5. Judging the cluster numbers of all current points whether reach to nearest cluster, if reached the maximum number of iterations, and then end it
int okCount = 0;
for(int i = 0; i <tsLength; i++){
 if(nearestMeans[i] == assignMeans[i]) okCount++;
 }
System.out.println("okCount = " + okCount);
if(okCount == tsLength || iterNum >= 10) break;
//6. If the conditions are not met, need to make another iteration of clustering. Then modify clustering members and the clustering information of each point
clusterMember.clear();
for(int i = 0; i < tsLength; i++){
 assignMeans[i] = nearestMeans[i];
if(clusterMember.containsKey(nearestMeans[i])){
 clusterMember.get(nearestMeans[i]).add(i);
 }
else {
 mem.clear();
mem.add(i);
Vector<Integer> tempMem = new Vector<Integer>();
tempMem.addAll(mem);
```

```
clusterMember.put(nearestMeans[i], tempMem);


}
 }

//7. Recalculate the center of each cluster
for(int i = 0; i < K; i++){
 if(!clusterMember.containsKey(i)){
continue;
 }
Map<String, Double> newMean = computeNewMean(clusterMember.get(i),
DataminingContest2009, testSampleNames);
Map<String, Double> tempMean = new TreeMap<String, Double>();
tempMean.putAll(newMean);
meansMap.put(i, tempMean);
 }
 }

//8. Generate clustering results and returns
Map<String, Integer> resMap = new TreeMap<String, Integer>();
for(int i = 0; i < tsLength; i++){
 resMap.put(testSampleNames[i], assignMeans[i]);
 }
return resMap;
 }
 /**Computing the new center of current cluster by using vector average
 * @param clusterM   Distance of all point to the clustering center
 * @param DataminingContest2009
```

```
 * @param testSampleNames
 * @return Map<String, Double>  New clustering center vector
 * @throws IOException
 */
private Map<String, Double> computeNewMean(Vector<Integer> clusterM,
Map<String, Map<String, Double>> DataminingContest2009,
String[] testSampleNames) {

// TODO Auto-generated method stub
double memberNum = (double)clusterM.size();
Map<String, Double> newMeanMap = new TreeMap<String,Double>();
Map<String, Double> currentMemMap = new TreeMap<String,Double>();
for(Iterator<Integer> it = clusterM.iterator(); it.hasNext();){
 int me = it.next();
currentMemMap = DataminingContest2009.get(testSampleNames[me]);
Set<Map.Entry<String, Double>> currentMemMapSet = currentMemMap.entrySet();
for(Iterator<Map.Entry<String, Double>> jt = currentMemMapSet.iterator(); jt.hasNext();){
 Map.Entry<String, Double> ne = jt.next();
if(newMeanMap.containsKey(ne.getKey())){
 newMeanMap.put(ne.getKey(), newMeanMap.get(ne.getKey()) + ne.getValue());
 }
else {
 newMeanMap.put(ne.getKey(), ne.getValue());
 }
 }
 }
 Set<Map.Entry<String, Double>> newMeanMapSet = newMeanMap.entrySet();
```

```
for(Iterator<Map.Entry<String, Double>> jt = newMeanMapSet.iterator(); jt.hasNext();){

Map.Entry<String, Double> ne = jt.next();
newMeanMap.put(ne.getKey(), newMeanMap.get(ne.getKey()) / memberNum);
 }
return newMeanMap;
 }
 /**Find out the nearest cluster center from the current point
 * @param double[][]
 * @return i
 * @throws IOException
 */
private int findNearestMeans(double[][] distance,int m) {
 // TODO Auto-generated method stub
double minDist = 10;
int j = 0;
for(int i = 0; i < distance[m].length; i++){
 if(distance[m][i] < minDist){
 minDist = distance[m][i];
j = i;
 }
 }
return j;
 }

/**Calculate the distance of the two points
 * @param map1
```

```
     * @param map2
     * @return double
     */
    private double getDistance(Map<String, Double> map1, Map<String, Double> map2) {
     // TODO Auto-generated method stub
    return 1 - computeSim(map1,map2);
     }
      /**Get initial point of kmeans algorithm
     * @param k
     * @param Map<String, Map<String, Double>> DataminingContest2009
     * @return Map<Integer, Map<String, Double>>
     * @throws IOException
     */
    private Map<Integer, Map<String, Double>> getInitPoint(Map<String, Map<String,
Double>> DataminingContest2009 , int K) {

    // TODO Auto-generated method stub
    int count = 0, i = 0;
    Map<Integer, Map<String, Double>> meansMap = new TreeMap<Integer, Map<String,
Double>>();
    System.out.println("The initial point of this time clustering file：");
    Set<Map.Entry<String, Map<String,Double>>> DataminingContest2009 =
DataminingContest2009.entrySet();
    for(Iterator<Map.Entry<String, Map<String,Double>>> it =
DataminingContest2009.iterator();it.hasNext();){

    Map.Entry<String, Map<String,Double>> me = it.next();
    if(count == i * DataminingContest2009.size() / K){
```

```
meansMap.put(i, me.getValue());
System.out.println(me.getKey() + " map size is " + me.getValue().size());
i++;
 }
count++;
 }
return meansMap;
 }

/**Output the clustering results to a file
 * @param kmeansClusterResultFile
 * @param kmeansClusterResult
 * @throws IOException
 */
private void printClusterResult(Map<String, Integer> kmeansClusterResult, String
kmeansClusterResultFile) throws IOException {

// TODO Auto-generated method stub
FileWriter resWriter = new FileWriter(kmeansClusterResultFile);
Set<Map.Entry<String,Integer>> kmeansClusterResultSet =
kmeansClusterResult.entrySet();
for(Iterator<Map.Entry<String,Integer>> it = kmeansClusterResultSet.iterator();
it.hasNext(); ){
 Map.Entry<String, Integer> me = it.next();
resWriter.append(me.getKey() + " " + me.getValue() + "\n");
 }
resWriter.flush();
resWriter.close();
```

```
    }
    public void KmeansClusterMain(String testSampleDir) throws IOException {
```

B.2 Extract each cluster's instances from the output file

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import weka.clusterers.SimpleKMeans;
import weka.clusterers.ClusterEvaluation;
import weka.core.Instances;
import weka.core.converters.ArffLoader;
public class Kmeans {
public static void startCluster(String inputfilename,String outputfilename) {
Instances ins = null;
    SimpleKMeans KM = null;
    ClusterEvaluation cl = null;
    File file = new File(inputfilename);
    ArffLoader loader = new ArffLoader();
    File f = new File(outputfilename);
    try {
    c1 = new ClusterEvaluation();
    loader.setFile(file);
    ins = loader.getDataSet();
    ins.deleteAttributeAt(0);// Ignore the first column
    KM = new SimpleKMeans();
    KM.setNumClusters(15);
```

```
KM.buildClusterer(ins);
C1.setClusterer(KM);
C1.evaluateClusterer(new Instances(ins));
double[] d = cl.getClusterAssignments();// Get the class label for each column
if (f.exists()) {
System.out.print("cunzai");
} else {
System.out.print("bucunzai");
f.createNewFile();
}
BufferedWriter output = new BufferedWriter(new FileWriter(f));
String dataline = "";
output.write(cl.clusterResultsToString());
for(int i = 0 ; i < d.length;i++){
dataline = d[i]+"";
output.write(dataline);
output.newLine();
System.out.println(dataline);
}
output.close();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (Exception e) {
// TODO Auto-generated catch block e.printStackTrace();
}
}
```

```
}
```

B.3 Unsupervised discretization

```
import java.io.*;
import weka.core.*;
import weka.filters.Filter;
import weka.filters.supervised.attribute.Discretize;
        /**
        * Shows how to generate compatible train/test sets using the Discretize
        * filter.
        *
        */
       public class DiscretizeTest {

         /**
          * loads the given ARFF file and sets the class attribute as the last
          * attribute.
          *
          * @param filename    the file to load
          * @throws Exception  if somethings goes wrong
          */
         protected static Instances load(String filename) throws Exception {
           Instances       result;
           BufferedReader  reader;

           reader = new BufferedReader(new FileReader(filename));
           result = new Instances(reader);
```

```
  result.setClassIndex(result.numAttributes() - 1);
  reader.close();

  return result;
}

/**
 * saves the data to the specified file
 *
 * @param data        the data to save to a file
 * @param filename    the file to save the data to
 * @throws Exception  if something goes wrong
 */
protected static void save(Instances data, String filename) throws Exception {
  BufferedWriter  writer;

  writer = new BufferedWriter(new FileWriter(filename));
  writer.write(data.toString());
  writer.newLine();
  writer.flush();
  writer.close();
}
 /**
 * Takes four arguments:
 * <ol>
 *   <li>input train file</li>
 *   <li>input test file</li>
```

```
*   <li>output train file</li>
*   <li>output test file</li>
* </ol>
*
* @param args        the commandline arguments
* @throws Exception  if something goes wrong
*/
public static void main(String[] args) throws Exception {
  Instances     inputTrain;
  Instances     inputTest;
  Instances     outputTrain;
  Instances     outputTest;
  Discretize    filter;

  // load data (class attribute is assumed to be last attribute)
  inputTrain = load(args[0]);
  inputTest  = load(args[1]);

  // setup filter
  filter = new Discretize();
  filter.setInputFormat(inputTrain);

  // apply filter
  outputTrain = Filter.useFilter(inputTrain, filter);
  outputTest  = Filter.useFilter(inputTest,  filter);

  // save output
```

```
            save(outputTrain, args[2]);
            save(outputTest,  args[3]);
        }
}
```

B.4 NominalAttributeValueConverter

```
package utility;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.HashMap;
import weka.core.Attribute;
import weka.core.AttributeStats;
import weka.core.FastVector;
import weka.core.Instance;
import weka.core.Instances;
public class NominalAttributeValueConverter {

        private final String WILDCARD_LABEL = "OTHER"; //The wildcard attribute value representing uninteresting values.
        private Instances template; //The template data set representing the attributes with interesting values.
        private HashMap<String, ArrayList<String>> inputAttributeSet;

        public Instances getInput() {
            return template;
        }

        public void setInput(Instances input) {
```

```
        this.template = input;
}

public NominalAttributeValueConverter(Instances input) {
        setInput(input);
}

public Instances getAttributeModifiedInstances(Instances input) {
        if ( input.numAttributes() != template.numAttributes() ) {
                System.err.println("Input and template dataset not compatible. Aborting..");
                System.exit(1);
        }

        Instances output = createModifiedInstancesFormat();
        for ( int i = 0; i < input.numInstances(); i++ ) {
                Instance newInst = new Instance(input.instance(i));

                for ( int j = 0; j < output.numAttributes(); j++ ) {
                        Attribute oldAttr = input.attribute(j);
                        if ( input.classAttribute().equals(oldAttr) )
                                continue;

                        if ( oldAttr.isNominal() ) {
                                Attribute newAttr = output.attribute(j);
                                double val = newInst.value(j);

                                if ( ((Double)val).equals(Instance.missingValue()) )
```

```
                                newInst.setMissing(j);
                            else {
                                if
    ( newAttr.indexOfValue(oldAttr.value((int)val)) == -1 )
                                    newInst.setValue(j,
newAttr.indexOfValue(WILDCARD_LABEL));
                                else
                                    newInst.setValue(j,
newAttr.indexOfValue(oldAttr.value((int)val)));
                            }
                        }
                    }
                    output.add(newInst);
                }
                return output;
            }

            private Instances createModifiedInstancesFormat() {
                Instances outputFormat;
                FastVector attrs = new FastVector();

                for ( int i = 0; i < template.numAttributes(); i++ ) {
                    Attribute oldAttr = template.attribute(i);
                    if ( oldAttr.isNumeric() ||
    template.classAttribute().equals(oldAttr) )
                        attrs.addElement(oldAttr.copy());
                    else {
                        AttributeStats stats = template.attributeStats(i);
```

```
                int[] valCounts = stats.nominalCounts;
                FastVector newAttributeVals = new FastVector();

                for ( int j = 0; j < oldAttr.numValues(); j++ ) {
                    if ( valCounts[j] != 0 )
                        newAttributeVals.addElement(oldAttr.value(j));
                }

                //At last, add the wildcard attribute value
                newAttributeVals.addElement(WILDCARD_LABEL);

                attrs.addElement(new Attribute(oldAttr.name(), newAttributeVals));
            }
        }

        //Create the output instance with desired format
        outputFormat = new Instances(template.relationName(), attrs, 0);
        outputFormat.setClassIndex(template.classIndex());
        return outputFormat;
    }
    public static void main(String[] args) {
        //UNDISCLOSED
    }
}
```