Fraud detection in bank transactions
ensemble of local sparse linear classifiers

Zhu, Z.

Award date:
2017

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Data Mining Group

Fraud Detection in Bank Transactions: Ensemble of Local Sparse Linear Classifiers

Master's Thesis

Zhanjie Zhu

Supervisors:

prof. dr. M. Pechenizkiy	TU/e
W. van Ipenburg MSc.	Rabobank

Assessment Committee:

prof. dr. M. Pechenizkiy	TU/e
W. van Ipenburg MSc.	Rabobank
dr. R. Medeiros de Carvalho	TU/e

Public Version

Publishing date: 31 Aug 2017

Abstract

In this thesis, we aim to develop and apply machine learning techniques for fraud detection in bank transactions. The business context of fraud detection in bank transactions has several challenges (e.g. high dimensional and highly imbalanced data) and model requirements (e.g. low false positive rate and good interpretability) that are not easy to solve and satisfy by simple machine learning techniques. We formulate the research problem as a supervised binary classification problem and our goal is to develop a binary classifier that can address the above challenges and satisfy the requirements.

Our approach is a combination of several state-of-art techniques (e.g. down-sampling and cost sensitive approach for highly imbalanced problem, L1-regularization for high dimensional problem, and using PAUC instead of AUC as model evaluation metric for low false positive rate requirement) and a novel clustering-based ensemble of sparse linear classifiers. Our main contributions of the clustering-based ensemble approach are the CCOvA (Class Crossing One-vs-All) classifier training scheme and SC (Soft Combination) scheme for combining local classifiers based on the instance's distances to different cluster centers. Compared with the two common classifier training schemes (i.e. One-vs-One and One-vs-All), the CCOvA scheme has higher predictive performance when combined with the SC scheme. Furthermore, if the obtained clusters have meaningful interpretations (e.g. each cluster corresponds to one sub-class), the clustering-based ensemble model is also easy to interpret since the interpretations of the local classifiers can be linked to the obtained clusters, and the SC scheme can be interpreted as giving different levels of trust to local specialized classifiers based on distance information. The main limitation of such approach is its high dependency on the clustering results.

We evaluate the above combined approach on a real transaction dataset provided by Rabobank. The single L1-regularized linear SVM model (with a stable feature selection technique) achieves PAUC_{100} performance of 3.75×10^{-5}, and our clustering-based ensemble approach improves the performance to 5.16×10^{-5} (by 40%) while keeping the final ensemble model relatively easy to interpret. Besides, the clustering-based ensemble model also outperforms the simple Bagging ensemble of linear classifiers. As a comparison, the state-of-art Random Forest achieves a slightly higher performance of 5.22×10^{-5} on the same dataset. By choosing a suitable classification threshold, our clustering-based ensemble model achieves FPR = 44 per 1 million and TPR = 0.55, which can provide relatively accurate predictions while keeping the FPR substantially below the workload limit defined by Rabobank.

Acknowledgement

Firstly I would like to thank my supervisors Prof. Mykola Pechenizkiy (at TU/e) and Werner van Ipenburg (at Rabobank). On the TU/e's side, Mykola gave me lots of valuable feedback and ideas on the thesis (e.g. how to validate an idea more reasonably), his constructive suggestions helped me a lot to improve the arguments and structure of my thesis. On the Rabobank's side, Werner van Ipenburg and Jan W. Veldsink helped me a lot from the start of the project and inspired me many new ideas from both academic and business perspectives. I am very grateful for their interest in my work and their support throughout the project, including but not limited to a strong machine for extensive experiments. I would also like to thank the whole KYC research group and the Rabobank's data scientists, with whom we shared our research progress and new ideas every week. Lastly, I would like to thank my family for always supporting me throughout my life.

This graduation project was a very nice experience and it was a good start of my journey to become a data scientist.

Zhanjie Zhu
Eindhoven, July 2017

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we briefly describe the background of fraud detection in banking industry, then we summarize some challenges and model requirements related to the fraud detection problem. We also summarize the approaches (chosen state-of-art approaches and our proposed approach) that are used in this thesis and our obtained results. Finally, we show the outline of this thesis.

1.1 Background

With the advancement of e-commerce and internet banking, huge amount of online transactions are happening every day (most are legal transactions but some are frauds), which makes automated fraud detection techniques more and more important for financial industry. Currently many banks and financial organizations are using rule-based systems for fraud detection, which heavily rely on human effort to create sophisticated rules and investigate suspicious transactions when the transactions violate some established rules and get alerts from the rule-based system. Thanks to the recent rapid development in data mining and machine learning, many state-of-art algorithms exhibit extraordinary performance and show great potentials to reduce or even replace human effort in identifying suspicious transactions in the financial industry.

As one of the major banks in the Netherlands, Rabobank handles millions of transactions every day and strives to identify suspicious transactions accurately and efficiently based on advanced machine learning algorithms. For research purpose, Rabobank has aggregated large volumes of anonymized transaction data, including confirmed abnormal transactions and normal transactions. In this project, we develop and apply machine learning techniques to identify abnormal transactions in the transaction dataset provided by Rabobank. As we discuss in Chapter 2, we formulate the fraud detection problem as a supervised binary classification problem, i.e. we try to classify whether a given transaction is abnormal or normal, but not considering the specific types of the abnormal transactions.

1.2 Summary of Challenges and Model Requirements

Firstly, a common challenge of fraud detection problem in banking industry is the *large volumes of data* (usually known as "Volume" of the five Vs in Big Data). Millions of transactions are happening every day, and the transaction data usually has been aggregated over years, which may cause problems in the phase of model training and testing due to limitations of time and computing resources. Secondly, the ratio of abnormal and normal transactions is usually *highly imbalanced*, since there are only dozens of transactions identified as abnormal every day while millions of transactions happening each day. In such a highly imbalanced situation, the abnormal transactions can be easily overwhelmed by the large amount of normal transactions and cause some common machine learning algorithms to be ineffective. Furthermore, the transaction data are *high dimensional*, i.e. each transaction contains thousands of features (variables) that may be

relevant to identifying suspicious transactions, these features are extracted from the rule-based system or manually identified by the domain specialists. Although these features are considered as relevant to some extents, most of them are weak signals and some of them are artifacts (false predictors) that give no practical use in reality (e.g. some hidden features that are equivalent to the timestamp), thus feature engineering techniques are required to filter out the artifacts. Thousands of features can be considered as relatively high dimensional, hence require special treatments. Lastly, as discussed by [6], another common challenge of supervised anomaly detection is that it is usually difficult to obtain representative and fine-grained labels for the abnormal class, and there can be multiple types of abnormal transactions that are very different to each other. In the binary classification problem setting, the abnormal transactions only have a main class label, i.e. abnormal, and the other transactions are simply regarded as normal if they have not been identified as abnormal within a certain period of time (such as one month from the transaction timestamp). The fact that *sub-classes may exist in the two main classes* may reduce the performance of some classification algorithms such as linear classifiers, and we propose a novel clustering-based ensemble approach to address this problem.

Besides facing the above challenges of the transaction data, there are also some requirements for the fraud detection techniques, e.g. the employed fraud detection techniques should generate *less false alerts* and have *good interpretability*. The requirements in the context of business problem are described in Section 2.1 in more details.

1.3 Summary of Approaches and Results

Summary of approaches

Here we summarize some chosen state-of-art approaches and our novel approach that are used to address the above challenges of the transaction data (i.e. large volumes, highly imbalanced, high dimensional, and may contain sub-classes in each of the two main classes) and satisfy the above realistic requirements of the model (i.e. producing less false alerts and easy to interpret). The chosen state-of-art approaches (abbreviated as SOA) and our novel approach (abbreviated as Novel) are shown in Table 1.1.

As we can see in the table, we use random sampling for model training and we use simple machine learning models that have fast training and prediction speed to tackle the challenge of large volumes of data.

To address the highly imbalanced problem, we use both down-sampling (Section 3.3.2) and cost sensitive approach (Section 4.2.2).

To address the high dimensional problem, we use linear classifiers with L1 regularization (Section 4.2.1) and a stable feature selection method based on sampling and averaging the feature vector (Section 4.2.5).

To address the challenge of "sub-classes exist in each main class," we propose a novel clustering-based ensemble approach, i.e. we perform clustering within each class separately, train local sparse linear classifiers using the CCOvA (Class Crossing One-vs-All) scheme, and combine the local classifiers by the SC (Soft Combination) scheme based on distance information to improve predictive performance (Chapter 5).

To satisfy the model requirement of producing less false alerts, firstly we use cost sensitive approach (Section 4.2.2), secondly we use PAUC as model evaluation metric (Section 3.3.3), and lastly we choose appropriate classification thresholds (Section 3.3.5).

To satisfy the model requirement of good interpretability, we use the above novel approach (clustering-based ensemble) with sparse linear classifiers as the base models so that each local classifier uses only a small number of features. Furthermore, each local classifier trained by the CCOvA scheme can be interpreted as specialized in identifying a certain obtained *cluster*. Hence, if the obtained clusters have meaningful and practical interpretations (e.g. each cluster corresponds to one sub-class of abnormal transactions), the final ensemble model is also easy to interpret since the interpretation of the local classifiers can be linked to the obtained clusters, and the SC scheme

Table 1.1: Summary of chosen approaches corresponding to each challenge and model requirement ("SOA" in Types means state-of-art approaches and "novel" means our proposed approach)

Challenges and Requirements	**Approaches**	**Types**
Large volumes of data	1. Random sampling for model training 2. Simple models (e.g. sparse linear classifiers) with fast training and prediction speed	SOA
Highly imbalanced	1. Down-sampling the normal class 2. Cost sensitive approach	SOA
High dimensional	1. L1 regularization to train sparse linear models 2. Stable feature selections based on sampling and averaging the feature vector	SOA
Sub-classes exist in each main class	1. Perform clustering within each class separately 2. Train local sparse linear classifiers using the "Class Crossing One-vs-All" scheme 3. Combine the local classifiers based on distance information to improve classification performance	Novel
Less false alerts	1. Cost sensitive approach 2. PAUC as model evaluation metric 3. Choosing appropriate model thresholds	SOA
Easy to interpret	1. Sparse linear classifiers (small number of features) 2. Interpret the local classifiers as specialized in identifying each of the obtained clusters (if the obtained clusters have meaningful interpretations) 3. Interpret the "Soft Combination" as giving different levels of trust to local specialized classifiers	1. SOA 2. Novel 3. Novel

can be simply interpreted as giving different levels of trust to the local specialized classifiers based on distance information.

Summary of results

On a real transaction dataset provided by Rabobank, our experiment results show that the PAUC is indeed more reasonable than the whole AUC as a model evaluation metric considering the realistic requirement of low false positive rate. Furthermore, the experiments also show that a single L1-regularized linear SVM model (with the stable feature selection) achieves PAUC_{100} performance of 3.75×10^{-5}, and our clustering-based ensemble approach significantly improves the performance to 5.16×10^{-5} (by 40%). Besides, the clustering-based ensemble model also outperforms the simple Bagging ensemble of linear classifiers (with average rule). As a comparison, the state-of-art Random Forest achieves a slightly higher performance 5.22×10^{-5} on the same dataset. By choosing a suitable classification threshold, our clustering-based ensemble model achieves FPR = 44 per 1 million and TPR = 0.55, which can provide relatively accurate predictions while keeping the FPR substantially below the workload limit defined by Rabobank. As for the interpretability, we also show that the clustering-based ensemble model trained on this transaction dataset is still relatively easy to interpret.

Besides, the experiments also show that the main limitation of the clustering-based ensemble approach is that the predictive performance and interpretability of the final ensemble model highly depend on the clustering results. Although we can overcome such limitation by trying different clustering settings and evaluating the obtained clusters with several metrics, it still requires more human effort and intervention than other off-the-shelf algorithms such as Random Forest.

1.4 Thesis Outline

This thesis is organized as follows:

- In Chapter 2 (Problem), firstly we introduce different aspects (e.g. conditions, objectives, challenges and requirements) of fraud detection in the context of business problem, and we show the conceptual workflow of fraud detection. Then we formulate our research problem according to the business problem settings. Finally, we briefly review some state-of-art solutions regarding the data challenges and model requirements, and we introduce our choices of the state-of-art solutions and also our proposed novel approach to address different challenges and requirements.
- In Chapter 3 (High Level Aspects of Our Approach for Fraud Detection), firstly we show the workflow of the R&D steps (e.g. model training and testing). Then we describe some high level aspects such as feature engineering techniques and model training settings that aim to address different challenges and requirements.
- In Chapter 4 (State-of-art Classification Techniques), firstly we describe some state-of-art ensemble methods (such as Random Forest as a Bagging method and XGBoost as a Boosting method), and we show some specific techniques (such as down-sampling when growing each tree) that can help address the above challenges and requirements. Then we introduce two L1-regularized and cost sensitive linear classifiers (i.e. L1-SVM and L1-LR) that are used as base classifiers in our following clustering-based ensemble approach.
- In Chapter 5 (Clustering-based Ensemble of Local Sparse Linear Classifiers), we propose a novel clustering-based ensemble approach. Firstly, we introduce the intuitions and main idea of such approach. Secondly, we break down this approach into different components and explain each component in details. Finally, we show an example of such ensemble approach and review related work to this clustering-based ensemble approach.
- In Chapter 6 (Experiment), firstly we show the experimental settings (e.g. dataset description). Then we show the experiment results of the state-of-art classification techniques and our proposed clustering-based ensemble approach. Finally, we give conclusions on the experiments.
- In Chapter 7 (Conclusion), we conclude our contributions in terms of academic and business values. And we discuss the limitations and future work.

Chapter 2

Problem

In this chapter, firstly we describe four different aspects (e.g. conditions, objectives, challenges and requirements) of fraud detection in the context of business problem. Secondly, we show the conceptual workflow of fraud detection in the business context. Then we show different possible data mining problem formulations of fraud detection (e.g. supervised, semi-supervised, and unsupervised approaches), and we formulate our research problem considering all the above four aspects of the business problem. Finally, we briefly review state-of-art solutions regarding the data challenges and model requirements, and we introduce our choices of the state-of-art solutions and also our proposed novel approach to address the challenges and satisfy the requirements.

2.1 Business Problem

In the context of bank transactions, fraud detection techniques refer to the techniques of identifying abnormal transactions from a stream of transaction data. There are different conditions, objectives, challenges and requirements regarding fraud detection in different business problem settings. For example, abnormal transactions may have different types and meanings, e.g. the transactions sent by stolen accounts can be considered as abnormal and the transactions involved in money laundering can also considered as abnormal. Furthermore, different types of data in the past can be provided for the fraud detection task, e.g. both confirmed abnormal transactions and normal transactions in the past can be provided, or just the confirmed normal transactions can be provided. As for the objectives of fraud detection, some business problem settings may require the employed techniques to identify completely unseen types of abnormal transactions, some may only need to identify similar abnormal transactions as the given abnormal ones in the past.

Besides the common challenges as we have discussed in Section 1.2, different business settings may have different requirements for the employed fraud detection techniques. Two important requirements that we consider in our business problem setting are *less false alerts* and *easy to interpret*. The false alerts are defined as the false predictions that we predict to be abnormal but are actually normal in reality. The model preference towards producing less false alerts is mainly due to the high human costs of manual investigations and the customer dissatisfaction when their transactions are falsely identified as abnormal. The interpretability of a machine learning model is critical in the banking or financial industry, since these industries are highly regulated and the domain specialists who are going to use these models always want to know exactly why and how a transaction is identified as abnormal, such transparent knowledge about the model can give the domain specialists and industry regulators confidence to trust and use the model in production. Hence, the machine learning models that are known to be "black box" like Artificial Neural Network are usually avoided considering the importance of interpretability in these highly regulated industries (there are ongoing research to make neural network interpretable but they are not the focus in this thesis).

Besides the above two requirements, the employed fraud detection techniques are also required

to not just give an alert when a new transaction is predicted to be abnormal, but also provide a score for each prediction that can indicate the decision confidence so that the domain specialists can investigate the potential abnormal transactions in a top-down manner (start from the most confident prediction).

In this thesis, we only consider the fraud detection problem (in the context of bank transactions) with the following specific conditions, objectives, challenges and requirements:

- **Conditions**
  1. "Abnormal" and "normal" are defined by the bank, and there may be different sub-classes contained in the abnormal and normal class.
  2. Both confirmed abnormal and normal transactions in the past need to be provided by the bank.
  3. Features that are considered to be relevant to fraud detection need to be provided for each transaction.
- **Objectives**
  1. Identify new abnormal transactions that have similar behaviours or patterns as the given abnormal transactions in the past.
  2. Unnecessary to identify the potential sub-types of the transactions (e.g. either it is money laundering or stolen account).
  3. Unnecessary to identify completely unseen and new types of abnormal transactions.
- **Challenges**
  1. Large volumes of data are provided (hundreds of millions of transactions in the provided case).
  2. High dimensional: lots of features are provided (thousands of features in the provided case), most of them are weak signals, some of them are artifacts.
  3. Highly imbalanced: lots of normal transactions are provided, but only a small number of abnormal transactions in the past are provided (the average ratio in the provided case is $10 : 1$ million).
  4. Sub-classes may exist in the two main classes (e.g. there may be different sub-types of abnormal transactions).
- **Requirements**
  1. Generate a small number of false alerts (less than 100 false alerts per 1 million transactions) while correctly identify more than half of the abnormal transactions. More precisely, FPR $<$ 100 per 1 million and TPR $>$ 0.5 (FPR and TPR are defined in Section 3.2).
  2. Easy to interpret to the domain specialists and/or regulators.
  3. Not just classify if a transaction is abnormal, but also provide a score that can indicate the decision confidence.

2.2 Conceptual Workflow of Fraud Detection

Here we describe a conceptual workflow of fraud detection in Figure 2.1, which is a conceptual abstraction in the business context to make the following explanations (such as the definition of True Positive) much easier. As we can see in the figure, the dashed box on the left side is related to the research and development of fraud detection techniques, and the other part is related to the application of fraud detection techniques in reality.

In the research and development phase, we retrieve data from the confirmed database (where each transaction has a confirmed label) and perform some R&D steps (e.g. model training and testing) to get the final classification model. These specific R&D steps are the focus of the following chapters, and the more detailed workflow of such R&D steps is shown in Figure 3.1.

The buffer database and confirmed database (along with the pre-defined time period and transfer rules) act as the connection between the application phase and R&D phase. The newly generated transactions from the transaction stream are stored in the buffer database for a pre-defined period of time (2 weeks in this example). If the buffered transactions are identified as abnormal within this period, they will be labelled as abnormal and immediately moved to the confirmed database. If the buffered transactions have not been identified as abnormal in the pre-defined period, they will be labelled as normal and moved to the confirmed database at the end of the time period. If a normal transaction in the confirmed database get identified as abnormal subsequently, its label in the confirmed database will be changed to abnormal immediately.

The application phase starts from the transaction data stream. To apply the final classification model that we get from the R&D phase, we process the transactions from the data stream in real-time or in a batch manner, and we pre-process the data to get the new transformed data (similar as in the R&D phase). Then we apply the classification model on the new transformed data, and we will get alerts for potential abnormal transactions with confidence scores. The alerts indicate that the transactions are identified as potentially abnormal, and the scores indicate the prediction confidence, based on which the domain specialists can perform manual top-down investigations on the potential abnormal transactions. The manually investigated transactions with their confirmed labels are then stored in the confirmed database.

In reality, the classification model needs to be updated periodically since there are always new types of abnormal transactions occurring. To update the model, we just restart the R&D phase from the confirmed database again (sine the confirmed database consistently gets newly confirmed abnormal and normal transactions).

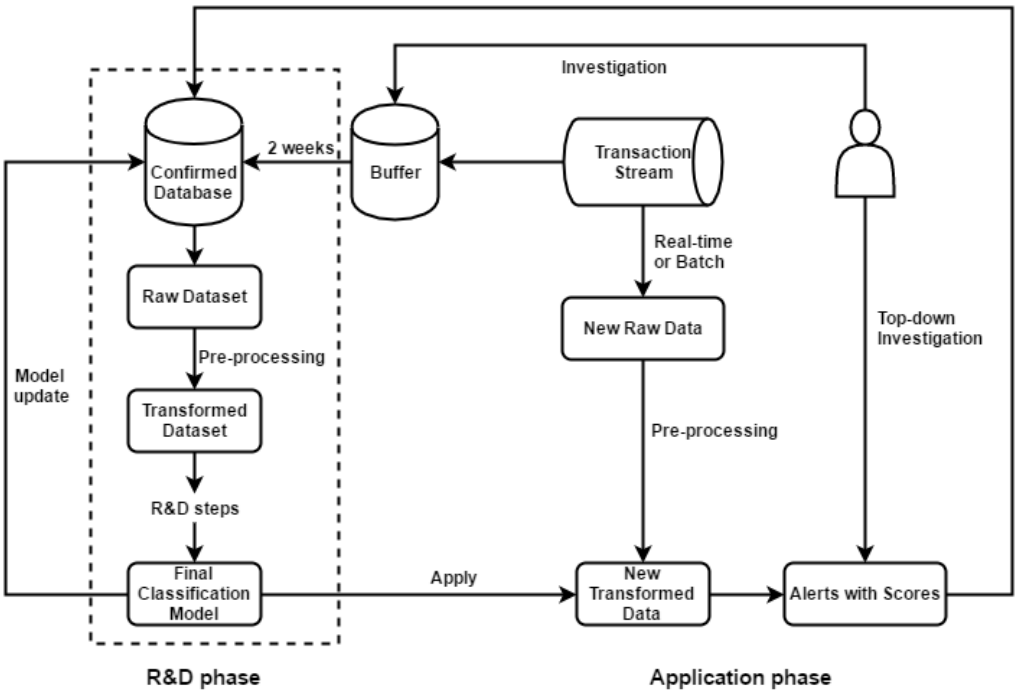


Figure 2.1: Conceptual workflow of fraud detection in R&D phase and application phase

2.3 Research Problem

As discussed in [24], there are different data mining problem formulations for fraud detection depending on different conditions and objectives of the business problem. The data mining formulations of the fraud detection problem can be mainly categorized into three types: supervised approach, semi-supervised approach, and unsupervised approach. In general, we can regard the fraud detection as the problem of finding patterns in data that do not conform to the normal behaviour. The supervised data mining approaches require labelled data for both the normal class and the abnormal class, the semi-supervised approaches in most fraud detection domains only require labelled data for the normal class, and the unsupervised approaches do not require any labelled data. The supervised approaches aim to identify new transactions that have similar abnormal behaviour and patterns as the given abnormal transactions in the past. The semi-supervised approaches only take the normal data as input, and aim to identify any pattern that does not conform to the given normal behaviour. The unsupervised approaches do not take any abnormal or normal behaviour as input, but just aim to identify the unusual patterns that do not conform to the usual behaviour (in this case, the unusual patterns are not guaranteed to be abnormal).

In the case of our business problem where both confirmed abnormal and normal transactions are provided by the bank, we formulate the research problem as a supervised data mining problem. Furthermore, considering that the objective is to identify similar abnormal behaviour or patterns as the given abnormal transactions in the past, but not necessary to identify the potential sub-types or completely unseen new types of abnormal transactions, we formulate the research problem as a *supervised binary classification problem*. More specifically, considering the conditions, objectives, challenges and requirements in the context of our business problem, we formulate the research problem as follows:

Given the X variables (predictor variables) and the Y variable (label) in the transaction dataset that has the four challenges as we have discussed above (i.e. large volumes of data, high dimensional, highly imbalanced, and sub-classes may exist in the two main classes), our research problem is to develop a *binary classifier* f such that:

1. $f : X \to \mathbb{R}$, the score $f(x)$ can indicate the confidence of the classification. The label y_i of the ith transaction is predicted to be abnormal if $f(x_i) > 0$, otherwise it is predicted to be normal.

2. f is required to have FPR $<$ 100 per 1 million and TPR > 0.5 (the TPR and FPR are defined in Section 3.3.3).

3. f is easy to interpret.

2.4 Related Work on Solutions to the Challenges and Requirements

In this section we briefly review some state-of-art solutions to address the above data challenges (i.e. large volumes of data, high dimensional, highly imbalanced, and sub-classes may exist in the two main classes) and satisfy the model requirements (i.e. low false positive rate, and easy to interpret).

Large volumes of data

For the challenge of large volumes of data, it is related to two Vs (Volume and Velocity) of the five Vs (i.e. Volume, Velocity, Variety, Veracity, and Value) in Big Data. The transaction data provided by banks are usually in large volumes, since the data have been aggregated over years and also the velocity of data generation is very fast (millions of transactions each day). Such large volumes of data may cause problems in the phase of model training and testing due to the limitations of time and computing resources, and the very fast data generation may also

cause problems when the fraud detection need to be done in (nearly) real-time. To deal with the challenge of large volumes, random sampling can be used to reduce the model training time, but the sampling method may cause information loss. Another option is to add more computing resources or use big data tools such as Hadoop with distributed computing clusters for model training and testing. Besides, simple classification models such as linear classifiers can be used considering the fast training and prediction speed of such simple classifiers. In our approach, we use both down-sampling method and sparse linear classifiers as base models, which are fast to train and test. As we have mentioned, down-sampling method may cause information loss or sampling bias, so we need to use appropriate techniques to address the information loss problem. In our case, for tree-based models we choose to down-sample the normal data *when growing each tree* (Section 4.1), and for sparse linear classifiers we use a so-called stable feature selection method to deal with the sampling bias based on taking multiple samples and averaging the feature vector (Section 4.2.5).

High dimensional

For the challenge of high dimensional data, in some literature high dimensional problems may refer to the situations where the number of data instances is much smaller than the number of features, but in our case high dimensional only refers to the situations where the number of features is relatively larger than common problems (e.g. more than 1000 features), and the features may include lots of weak signals and irrelevant features. Hence, the high dimensional problem in our case is actually a feature selection problem such that the used features can achieve a good balance between predictive performance and interpretability. As shown in a survey of feature selection methods [21], feature selections can be mainly categorized into three types: filter, wrapper and embedded methods. Filter methods select features based on the intrinsic properties of the features (such as correlation coefficient with the dependent variable) regardless of the learning algorithm. Wrapper methods use the learning algorithm as subroutine and evaluate different feature subsets based on the performance of the learning algorithm, examples of wrapper methods include recursive feature elimination and genetic algorithms. Embedded methods have the feature selection process embedded in the learning algorithm, examples of embedded methods include decision trees and L1 regularization. In our approach, we mainly use embedded methods since they have better predictive performance than filter methods and usually faster training speed than the wrapper methods. More specifically, we use L1-regularized linear classifiers or tree-based models to tackle high dimensional problem, and we use a so-called stable feature selection method for linear classifiers to filter out noisy features based on sampling and averaging the feature vectors (Section 4.2.5).

Highly imbalanced

For the challenge of highly imbalanced classes (i.e. there are lots of normal transactions but only a few abnormal transactions in the dataset), besides choosing more appropriate evaluation metrics (such as the AUC instead of the accuracy), the state-of-art approaches to tackle the highly imbalanced problem can be mainly categorized into three types: down-sampling the majority class, over-sampling the rare class, and cost-sensitive learning [33]. Examples of over-sampling the rare class include the SMOTE [8], which generates new and non-replicated rare class examples by interpolating the nearest rare class examples. However, SMOTE may generate meaningless data points when the features are categorical. In our approach, we choose a more appropriate evaluation metric (i.e. PAUC in Section 3.3.3), and we use both down-sampling the majority class and cost sensitive learning to tackle the highly imbalanced problem. We explain the down-sampling approach in Section 3.3.2 and the cost sensitive learning in Section 4.2.2.

Sub-classes within each main class

For the challenge of sub-classes may exist in each main class, it refers to the problem that the complex regions formed by the sub-classes within each main class may reduce the classification

performance of some classification models (especially simple classifiers like linear classifiers). State-of-art solutions to tackle this problem include using non-linear classifiers such as decision trees that can find non-linear complex decision boundaries. However, as we discuss in Section 5.1, using a single decision tree may lead to over-fitting in reality, and using many decision trees (like Random Forest based on instance sampling and feature sampling) may make the model harder to interpret (compared with a single decision tree). Hence, we propose a clustering-based ensemble approach for (sparse) linear classifiers to have better predictive performance than single linear classifiers while keeping the final ensemble model possibly easy to interpret (depending on the interpretability of the obtained clusters), i.e. we first perform clustering within each main class, then we train local classifiers using the "Class Crossing One-vs-All" scheme, and finally we combine the local classifiers based on the distance information (the instance's distances to different cluster centers). We explain this ensemble approach in Chapter 5 in more details.

Model requirements

For the model requirement of low false positive rate, similar as tackling the highly imbalanced problem, we use PAUC as evaluation metric (Section 3.3.3), and we use both down-sampling and cost sensitive learning in our approach. Besides, we also need to choose appropriate classification thresholds for our chosen model to achieve preferred false positive rate (Section 3.3.5). For the requirement of good interpretability, we again use our novel clustering-based ensemble approach as mentioned above, i.e. we can interpret each local classifier as specialized in identifying a certain obtained cluster (but it still depends on the interpretability of the obtained clusters). The choice of using L1-regularized sparse linear classifiers as the base classifiers in the ensemble approach can also improve the interpretability. Furthermore, we can interpret the weighted sum combination as giving different levels of trust to the local specialized classifiers. For the requirement of not just outputting the discrete labels but also the confidence scores, we only use classification models that can generate such scores, such as SVM model and logistic regression model. And this requirement also relates to the choice of classifier combination schemes in our ensemble approach (Section 5.8), since some common combination schemes such as majority voting cannot generate continuous confidence scores as output when the number of votes is small.

Chapter 3

High Level Aspects of Our Approach for Fraud Detection

In this chapter, we describe the high level aspects of our approach that aim to address different data challenges and model requirements defined in the research problem. These high level aspects can be combined with different classification models that are described in the next two chapters.

In the first section, we show the workflow of the R&D steps (e.g. model training and testing), then we describe each component (in red) in the workflow one by one in the following two sections.

In the second section, we describe some feature engineering techniques such as data discretization that need to be performed before training the classification models.

In the third section, we introduce high level aspects of our model training and testing (e.g. evaluation metrics). Each of these aspects aims to address a corresponding data challenge or requirement. More specifically, we split the dataset by time to avoid unrealistic testing results, we down-sample the majority class to (partly) address the highly imbalanced challenge, we choose PAUC as our model evaluation metric to address the highly imbalanced challenge and the requirement of low false positive rate, and we choose an appropriate classification threshold to achieve a preferred false positive rate.

3.1 R&D Workflow

The workflow of our R&D procedures is shown in Figure 3.1. As we can see, the R&D workflow starts from the confirmed database (where all the transactions have been confirmed as abnormal or normal), and then we pre-process the raw dataset (such as some feature engineering techniques) to get the transformed dataset. Then we split the transformed dataset into training set and testing set. For the training set, we further perform down-sampling on the normal class and retain all the abnormal data to get the model training set. On the other hand, the testing set is left for final model evaluation. For the model training set, we train our selected model and tune the hyper-parameters by cross validation and using the PAUC as evaluation metric. Then we apply the tuned model on the testing set to have a final model evaluation. If the model perform well on the testing set, we retrain this model on the whole transformed dataset (also down-sample the normal class and retain all the abnormal data). Finally, we select an appropriate classification threshold based on the transformed dataset and the final model. Then the final model with the chosen threshold can be used in the application phase.

This workflow can be combined with different classification models that are described in the next two chapters. There are different steps in this R&D workflow addressing different data challenges and requirements. We show the steps (in red) in the following sections one by one.

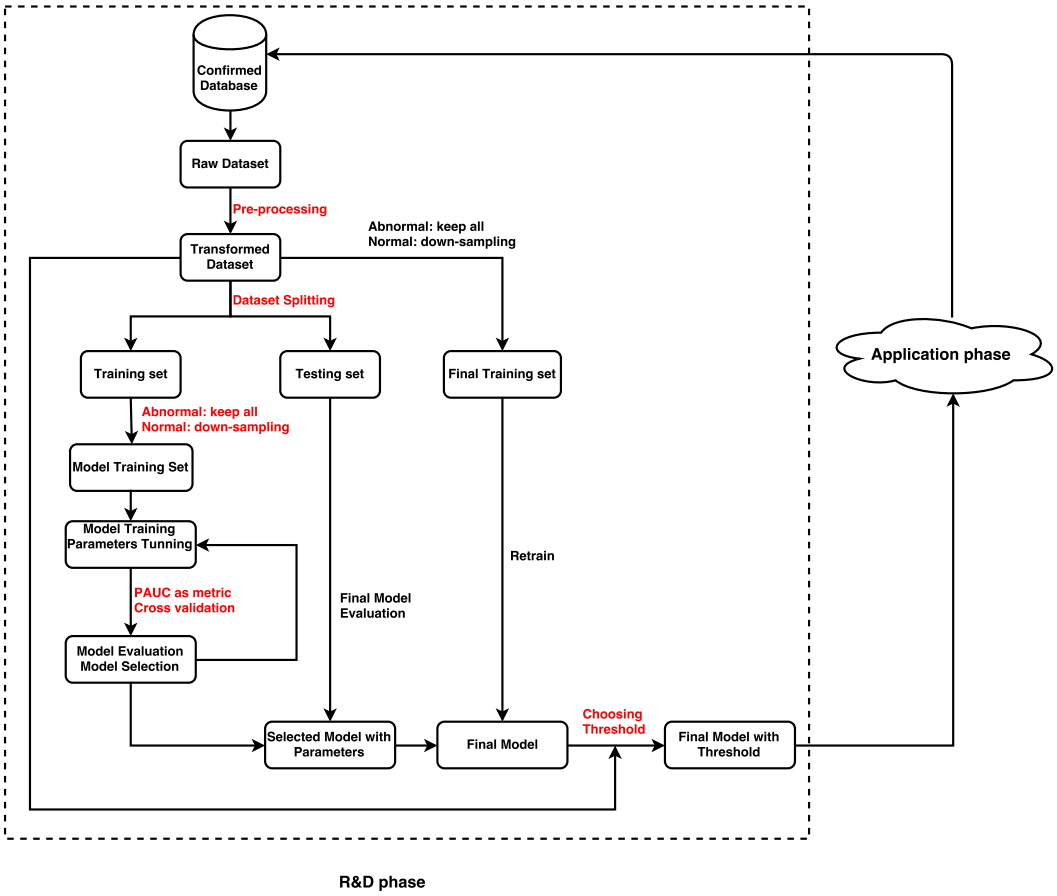


Figure 3.1: R&D workflow of fraud detection

3.2 Feature Engineering Techniques

We show some feature engineering and pre-processing techniques in this section. These techniques are used to transform the raw dataset into transformed dataset, which is then used for subsequent model training and testing.

3.2.1 Data types determination

The features used in the raw transaction dataset can be mainly categorized into three types: categorical (e.g. country code), numerical (e.g. transaction amount), and textual (e.g. beneficiary address). The categorical features usually have relatively small number of unique values, while the numerical and textual features have much more unique values. Hence, we can first determine whether a feature is categorical based on an aggregated sample, i.e. we can aggregate the data in a time range (e.g. one week), count the number of unique values for each feature, and determine the feature type to be categorical if the number of unique values is smaller than a pre-defined level (e.g. 200). The features that have a relatively large number of unique values are determined to be numerical or textual based on the proportion of numerical values, e.g. if a feature has more than 200 unique values in the aggregated sample *and* more than 80% of the unique values are numerical, it is determined to be numerical, otherwise it is determined to be textual. After determining the data types for each feature, we apply different discretization rules for features of different data types as follows.

3.2.2 Data discretization

Based on the determined data types, we can apply different data discretization rules for different data types as follows. For categorical features, we can simply sort the categories according to ASCII order if the majority of categories are non-numerical or according to numerical order if the majority of categories are numerical. Then we can encode the categories into different numbers. For both numerical and textual features, we can use *frequency-based binning* method to bin and encode the values into numbers, the only difference between binning the numerical and textual features is how we sort the values, i.e. we sort the numerical features according to numerical order and sort the textual features according to ASCII character order. The frequency-based binning method aims to produce bins that have the same (or similar) frequency of the contained original values. We describe the data discretization method in more details in Section 6.1.1.

3.2.3 Removing artifacts and single-valued features

Here artifacts refer to the features that can perfectly predict the dependent variable but useless in reality, e.g. a feature that is equivalent to the timestamp may give perfect predictions when the training set and testing are split by time. We can identify potential artifacts by the correlation coefficient with the dependent variable, i.e. the features that have high correlation coefficient may be artifacts. We also require domain knowledge to investigate those potential artifacts.

Single-valued features refer to the features that have only one single value among a large enough sample. We remove the single-valued features because they cannot provide information for classification and it can speed up the model training (if the training implementation does not include this feature).

3.2.4 One-hot encoding

Some classification models such as tree-based models can handle categorical data in nature, but some models such as linear classifiers cannot. Hence, we need to convert all the categorical features to one-hot encoding (dummy variables) for models like linear classifiers. Firstly, we identify the categorical features that have exactly 2 unique values in the aggregated sample, and convert each of them into a single 0-1 binary variable (maps the larger bin value to 1 and the smaller one to 0). Secondly, we convert all the features that have at least 3 unique values in the aggregated sample into one-hot encoding, i.e. a single variable with n observations and p unique values is converted to p variables with n observations, and each observation of the p variables indicate the presence 1 or absence 0 of the corresponding value. After this one-hot encoding conversion, the predictor variables have 1 single variable for each binary variable, and p variables for each variable with p unique values ($p \geq 3$).

3.3 High Level Aspects of Model Training

In this section, we describe the high level aspects of the model training (e.g. dataset splitting, model evaluation metrics) in our approach. These high level aspects can be combined with all the different base classification models that are described in the next two chapters.

3.3.1 Dataset splitting by time

After applying the above pre-processing techniques, we split the transformed dataset into training set and testing set, which is usually done by stratified random sampling. However, the stratified random sampling may make the model testing results too optimistic and unrealistic in our case, since it is not uncommon to have very similar transactions happening in a short period of time (e.g. 4 very similar transactions triggered by the same person happened in a few minutes), if we split the dataset by stratified random sampling, some of the very similar transactions that fall

into the testing set will easily get classified correctly since their similar counterparts fall into the training set.

To address the above problem, we can split the time span of the dataset into two consecutive parts, then we take the transactions in the first part of the time span as training set and the transactions in the second part as testing set. For example, if the total time span of a dataset is 5 months, then we take the transactions that happened in the first 4 months as training set, and take the transactions that happened in the last month as testing set. In this way of splitting, the very similar transactions happening in short time can only fall into either the training set or testing set, which makes the testing results more realistic and reliable. This time-based dataset splitting manner is used in all of our experiments.

There are even more realistic evaluation schemes such as prequential evaluation (test the model on new instances and retrain the model by including the new instances, and so on). However, we only consider this static training and testing manner in this thesis due to time limitations.

3.3.2 Down-sampling the normal class

Considering the highly imbalanced challenge of the transformed dataset, we perform down-sampling on the normal class data and retain all the abnormal class data in the training set. The down-sampling ratio depends on different classification models that we use in our approach. For example, we down-sampled the normal transactions from 150 million to around 2.7 million as inputs for the random forest model. Furthermore, we can perform down-sampling in different places, e.g. we perform down-sampling when growing each tree for the Random Forest model in order to avoid too much information loss (Section 4.1.2). We will specify and justify the down-sampling ratio for each model in the experiment section.

3.3.3 Model evaluation metrics

Many evaluation metrics are available for binary classification models, e.g. confusion matrix, precision, recall, accuracy, F-measure and AUC. Hereby we formally define several evaluation metrics that are related to our binary classification problem, and we discuss which of them is the most suitable in our case.

First we define GTP (ground truth positive), GTN (ground true negative), TP (true positive), FP (false positive), TN (True negative) and FN (false negative), because many evaluation metrics depend on the these concepts. The definitions of these concepts are related to how we confirm the labels of transactions in the business problem settings. As we have discussed in the business problem (Section 2.2), the newly generated transactions from the stream are stored in the Buffer Database for a pre-defined period of time (let's assume x days). If the buffered transactions are identified as abnormal within this period, they will be labelled as abnormal and *immediately* moved to the Confirmed Database. If the buffered transactions have not been identified as abnormal in x days, they will be labelled as normal and moved to the Confirmed Database *at the end* of the x days. If a normal transaction in the Confirmed Database get identified as abnormal subsequently, its label in the Confirmed Database will be changed to abnormal immediately.

For model evaluations in the R&D phase (workflow shown in Figure 3.1), the definitions of the above concepts are simple and direct because the dataset used for R&D is taken from the Confirmed Database, where the label of each transaction are considered as confirmed according to the above rule. The GTP is defined as the number of transactions that have been confirmed as abnormal according to the above rule, and the GTN is the number of transactions that have been confirmed as normal according to the above rule. The TP is defined as the number of transactions that have been confirmed as abnormal *and* predicted to be abnormal, the FP is defined as the number of transactions that have been confirmed as normal *and* predicted to be abnormal, and similar for the definitions of TN and FN.

For the application phase, the model evaluation process is shown in Figure 3.2. Assume that we are in Place 1 (so the new transactions keep coming in from the stream) and we have used the Confirmed Database to train a model that we can apply to new transactions, now we can choose

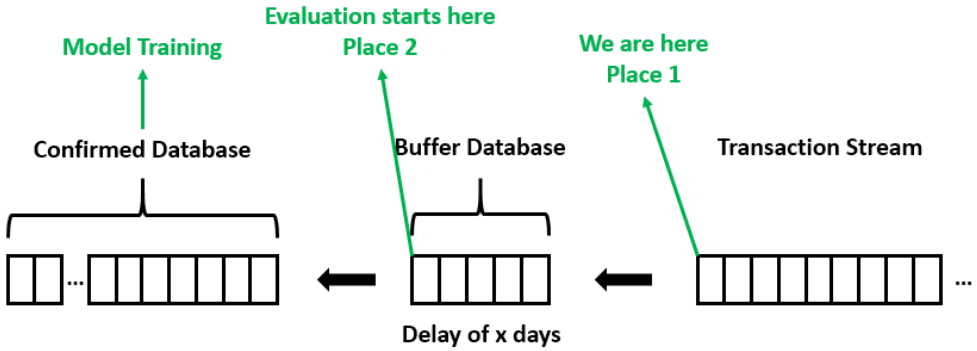


Figure 3.2: Model evaluation in application phase

two different places to start evaluating the model in reality. If we start the model evaluation in Place 1, it means we apply and evaluate the model only on new transactions, which is quite intuitive but we need to wait for at least x days for the transactions to be confirmed. Hence, instead we can start the model evaluation in Place 2 (the earliest transactions in the Buffer Database), since these transactions are not used for model training and also close to be confirmed, which means that we don't have to wait for too many days to start seeing the evaluation results. Furthermore, a transaction can only get evaluated if it is in the Confirmed Database, so we can apply the model on the transactions starting from Place 2 and keep track of them, and we can evaluate them once they get into the Confirmed Database (according to the above rule). The definitions of the above concepts (e.g. GTP, GTN, TP, FP) are the same as in the R&D phase, the only difference is that in the application phase we only evaluate the transactions that are entered into the Confirmed Database and we start evaluating the transactions from the Place 2.

Table 3.1: Confusion matrix

	PP (Prediction Positive)	**PN (Prediction Negative)**
GTP (Ground Truth Positive)	TP (true positive)	FN (false negative)
GTN (Ground Truth Negative)	FP (false positive)	TN (true negative)

Now we formally define several relevant evaluation metrics based on the definitions of the above concepts:

- Confusion Matrix: defined in Table 3.1
- Precision $= \frac{TP}{PP} = \frac{\text{True Positive}}{\text{Prediction Positive}} = \frac{\text{True Positive}}{\text{True Positive+False Positive}}$
- Recall $= \frac{TP}{GTP} = \frac{\text{True Positive}}{\text{Ground Truth Positive}} = \frac{\text{True Positive}}{\text{True Positive+False Negative}}$
- Accuracy $= \frac{TP+TN}{all} = \frac{\text{True Positive+True Negative}}{\text{Total Population}} = \frac{\text{True Positive+True Negative}}{\text{Ground Truth Positive+Ground Truth Negative}}$
- F1-score $= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision+Recall}}$
- TPR (true positive rate) $= \frac{TP}{GTP} = \frac{\text{True Positive}}{\text{Ground Truth Positive}}$ (same as Recall)
- FPR (false positive rate) $= \frac{FP}{GTN} = \frac{\text{False Positive}}{\text{Ground Truth Negative}}$
- ROC-curve: the curve of TPR (y-axis) against FPR (x-axis) when the classification threshold is varied.
- AUC: the whole area under the ROC-curve along the whole span of FPR (x-axis) from 0 to 1.

- PAUC_n: the partial area under the ROC-curve along the span of FPR (x-axis) in the range $0 \leq x \leq n \times 10^{-6}$, where $n \times 10^{-6}$ is a predefined level indicating the maximum allowed FPR value in reality.

The single metrics such as precision, recall, TPR and FPR are not enough to provide a balance for the classification model, e.g. a model that predict all transactions to be positive can yield recall or TPR to be 1, but the precision and FPR are very bad. The F1-score can be viewed as the harmonic mean of the precision and recall, it is usually used to balance the precision and recall rate, but it is not suitable for our case since there is a realistic requirement of FPR and the F1-score cannot explicitly control the FPR. The accuracy is widely used and simple, but it is also not suitable for the highly imbalanced class, e.g. if a dataset has 95% transactions as normal (negative), then a model that predict all transactions to be negative can yield accuracy 0.95, which is very high but not reasonable since missing the positive ones can be costly in reality.

The above metrics only require the classification models to have discrete predictions (class labels) as output. In contrast, the AUC requires the binary classification models to generate continuous (or nearly continuous) scores that can indicate the confidence of the model's predictions, e.g. the probability output of a binomial logistic regression model, or the distance to the decision boundary of a linear SVM (support vector machine) model. The AUC can be interpreted as a single number that can summarize the classification model's overall performance on the testing dataset, since the ROC-curve is generated by varying the classification threshold and it provides a way to choose a preferred balance between TPR and FPR.

However, considering that there is a realistic requirement of the classification model to have FPR less than a predefined level (in our case, the model is required to have FPR $\leq$ 100 per 1 million), the whole AUC is still not the best metric in our case since the FPR (x-axis) of the whole AUC can range from 0 to 1. The area when FPR is larger than 100 per 1 million (on the right side) is useless to us and it can influence the whole AUC value. Besides the whole AUC, we may also consider to use the single point TPR when the FPR is exactly (or very close to) the predefined level. However, the single point TPR is also not suitable since our model is required to have FPR *at most* the predefined level, but not necessary to be exactly equal to it. For example, if the TPR keeps the same when the FPR varies from 0.05 to 0.1, then we should choose the point at FPR = 0.05 instead of the point at 0.1.

We illustrate the differences between the PAUC, the whole AUC and the single point TPR at predefined FPR in two imagined scenarios (as shown in Figure 3.3 and Figure 3.4). In both figures, we can see two ROC curves of two classifiers respectively[1]. In Figure 3.3, we prefer classifier 1 over classifier 2 since our predefined requirement of FPR is at most 0.1 and the TPR of classifier 1 is consistently higher than classifier 2 when FPR is less than 0.1. However, in this imagined scenario, the metric of the whole AUC suggests to choose classifier 2 since it also includes the area on the right part, which is useless to us and affects the whole AUC value. In contrast, the PAUC (when FPR $\leq$ 0.1) can help us select the right classifier. In Figure 3.4, we may prefer classifier 1 over classifier 2, since theoretically classifier 1 can achieve TPR = 0.93 with FPR only 0.05, on the other hand, classifier 2 can only achieve TPR = 0.98 with FPR = 0.2 (we sacrifice FPR 0.15 for the gain of TPR 0.05). However, the metric of single point TPR at the predefined FPR level 0.2 suggests us to choose classifier 2 since it has a higher TPR at that single point compared with classifier 1. In contrast, the PAUC (when FPR $\leq$ 0.2) still provides us with the right choice again.

In summary, the PAUC is more suitable than all the above metrics including the whole AUC and the single point TPR at predefined FPR level. In our case, the bank prefers classification models to have FPR $\leq$ 100 per 1 million, since typically there are around 1 million transactions everyday, then there are at most 100 false alerts generated everyday on average if the FPR requirement is satisfied. For *final model evaluation and comparison* on the testing set, we use PAUC_{100} (FPR $\leq$ 100 per 1 million) considering the realistic requirement. For the model evaluation *when tuning the hyper-parameters* (as shown in the next subsection), considering that the partial area when FPR $\leq$ 100 per 1 million may be too small and noisy, we choose a larger area PAUC_{300} (FPR $\leq$ 300

[1]Such exact shapes of ROC curves are not likely to occur in reality, we just use them for illustrations.

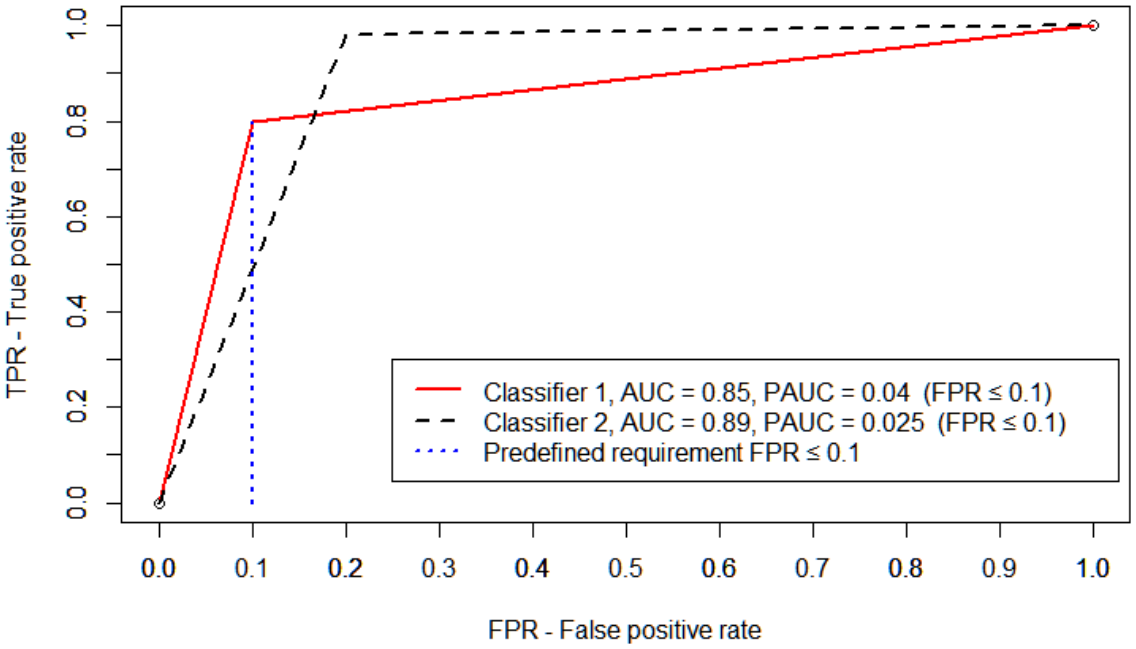


Figure 3.3: Comparison between the whole AUC and the PAUC

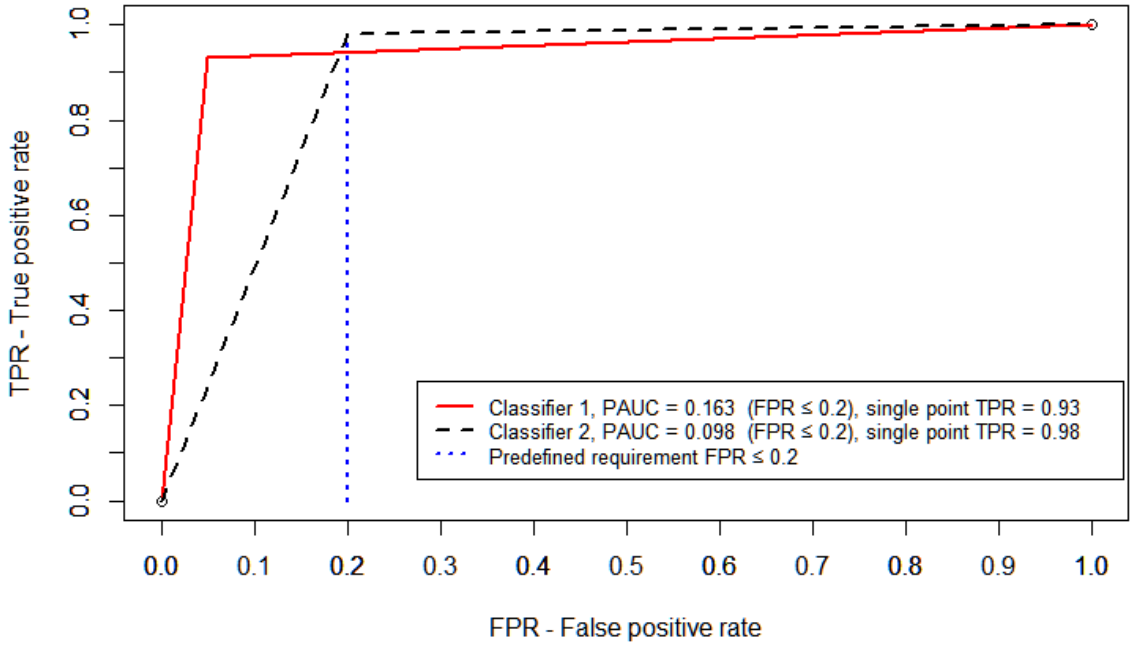


Figure 3.4: Comparison between the PAUC and the single point TPR

per 1 million) and use this evaluation metric for hyper-parameter tuning consistently in all of our experiments.

3.3.4 Hyper-parameter tuning by cross validation

In the phase of hyper-parameter tuning, we use 5-fold stratified cross validation with 2 repetitions, i.e. we split the training dataset into 5 folds by random stratified sampling (we use stratified sampling due to the highly imbalanced classes), train the model with specific parameters on each 4 different folds of the 5 folds, test the model on the rest fold (using the PAUC_{300} as evaluation metric when tuning), and take the average of the 5 evaluation metric values. Then we perform the above process again with another random stratified splitting of the training set and again take the average of the evaluation metric values. Finally, the average evaluation metric value PAUC_{300} can be viewed as an unbiased evaluation of that specific model (with those specific parameters), and it can be used to select the best hyper-parameters for different classification models.

3.3.5 Choosing threshold

For the binary classification models that can output (nearly) continuous confidence scores, we need to select a suitable score threshold such that the model can achieve the preferred TPR and FPR. Since we use the PAUC as model evaluation metric, which is generated by varying the classification threshold and constraining the range of FPR, the default threshold (e.g. majority vote ratio 0.5 in Random Forest model) used by the chosen model does not guarantee to achieve the preferred TPR and FPR.

After we have chosen the best model with the best hyper-parameters according to the PAUC metric (by cross validation), we have a ROC curve as in the above Figure 3.4. Then we can select a point on the curve with the preferred balance between TPR and FPR (also satisfying the FPR constraint) according to domain knowledge. After selecting a point on the ROC curve, we can take the corresponding FPR value to guide the process of choosing classification threshold. Let's say we select the turning point with FPR = 0.05 in the above Figure 3.4, the next step is to calculate an approximate threshold such that the model can get the chosen FPR. Since the testing dataset is only used for model testing and final evaluation, it cannot be used for choosing threshold.

Hence, we aggregate all the *normal transactions* in the last few weeks of the transformed dataset, apply the chosen classification model on those normal data, and get the score which is (nearly) the top X% among all the calculated confidence scores, where X% is the FPR of the chosen point (5% in the example of Figure 3.4). It is reasonable to choose the threshold in this way, because (1) the FPR is only concerned with the ground-truth normal data (2) usually only a small part of the normal transactions in the transformed dataset are used for model training since the down-sampling ratio is very small (e.g. 1 million out of 1 billion normal transactions), thus there are lots of out-of-bag samples in the aggregated normal transaction sample, which can avoid poor choices of the threshold.

Chapter 4

State-of-art Classification Techniques

In this chapter, we describe several state-of-art classification models (both linear and non-linear classifiers) and different techniques that can adapt those classification models in order to address the challenges and requirements defined in the research problem. We focus on non-linear classifiers in the first section and linear classifiers in the second section.

Firstly, we describe the two most popular ensemble methods, i.e. bagging and boosting. Corresponding to bagging and boosting respectively, we describe two state-of-art tree-based ensemble models, i.e. Random Forest and XGBoost (Extreme Gradient Boosting), both of which are non-linear tree-based classifiers. We choose the Random Forest model since (1) it has good predictive performance and robustness against over-fitting; (2) it is widely used in the industry; (3) it can act as a representative model of the bagging method in our later comparisons. We choose XGBoost since it can act as a representative example of boosting, and it also exhibits great predictive performance.

Secondly, we introduce the L1 regularization and cost sensitive approach, which are combined with two base linear classifiers (linear SVM and logistic regression) to address the high dimensional and highly imbalanced problem in our case. Lastly, we show a feature selection method for linear classifiers that can filter out the noisy features based on sampling and averaging the feature vectors.

4.1 Tree-based Ensemble Models (non-linear)

In this section, we describe two ensemble methods, i.e. bagging and boosting. And we also introduce two state-of-art tree-based ensemble models, i.e. Random Forest and XGBoost. Each of them acts as a representative model of bagging and boosting. Furthermore, we show how to address the highly imbalanced problem in our case based on down-sampling the normal data and adjusting the imbalanced ratio *when growing each tree*.

4.1.1 Ensemble Methods: Bagging & Boosting

Bagging was proposed by [4]. It is based on a simple but effective idea, i.e. it draws multiple bootstrap samples (random samples with replacement), trains models on the obtained samples, and then it takes average of the results (e.g. majority vote) predicted by those models. Bagging aims to reduce the variance, and it is more suitable for models with low bias but high variance. The models trained over the multiple bootstrap samples are independent to each other, which enables the implementation to be parallelized easily.

Boosting works in a different direction compared with Bagging, i.e. it starts from a weak classifier and tries to improve the weak classifier into a stronger one. The boosting method improve the weak classifier by identifying the instances where the weak classifier performs poorly

and giving more weights to those instances in order to predict them better in the next round. Different as Bagging, the Boosting method aims to reduce the bias, and it is more suitable for models with high bias but low variance. The training manner of Boosting is sequential, i.e. the training of a new model depends on the previous model, which is also different as Bagging's parallel training manner.

4.1.2 Random Forest

Random Forest was proposed by [5]. In addition to the idea of bagging, it adds an additional layer of randomness by making each node to split among only a subset of the features. In more details, the originally proposed Random Forest first draws multiple bootstrap samples (same size as the original dataset), then it trains independent tree models for each bootstrap sample. For each node of each tree, it selects a random subset of the original features, from which the tree model chooses the best feature and the best split at that node. All the trees are grown with the maximum depth without pruning such that the model has low bias but high variance, and the high variance of each single tree is reduced by the idea of bagging.

Many variants of Random Forest have been proposed to improve its performance in different problems, some of them aim to solve the imbalanced class problem, which is also one of the challenges in our case. For example, the BRF (Balanced Random Forest) and WRF (Weighted Random Forest) were proposed to solve the imbalanced class problem [9]. The BRF follows the sampling approach, however, it does not down-sample the majority class *before* applying the Random Forest, instead it down-samples the majority class *when growing each tree* such that the number of majority and minority instances are exactly the same. The WRF follows the cost sensitive approach, i.e. it incorporates the costs in two places, one is the Gini criterion when splitting each node, and the other one is the weighted majority vote when making predictions.

In our case where the transaction dataset is also highly imbalanced, we wanted to follow the WRF (cost sensitive) approach. However, the WRF approach is not implemented in the RandomForest package in R [20]. Hence, we followed the idea of the BRF approach and slightly changed it such that it fits in our case. The BRF follows the data sampling approach to tackle the imbalanced class problem. We show the two places where we can perform sampling when applying the Random Forest algorithm in Figure 4.1.

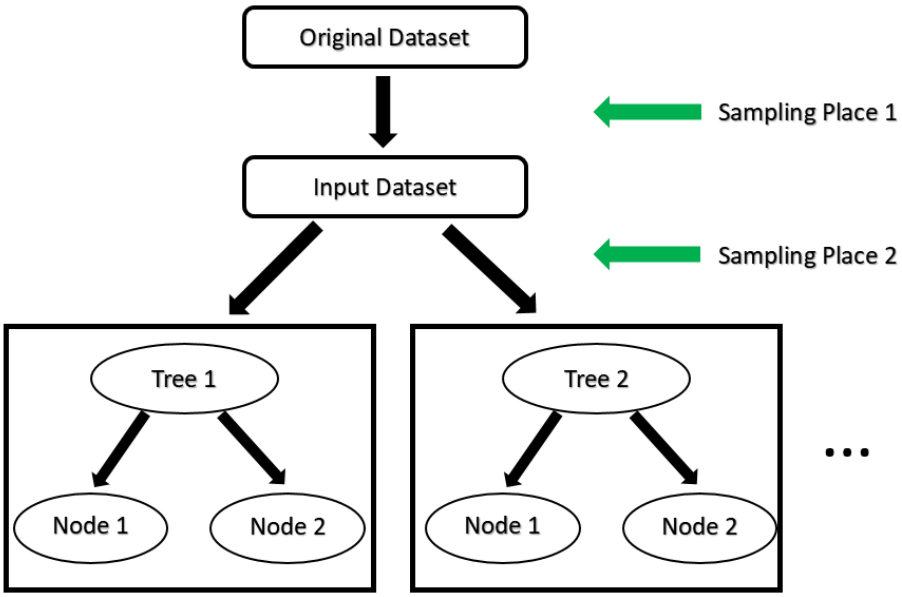


Figure 4.1: Two places in Random Forest where we can perform sampling

As we can see, if we perform sampling at the place 1 (before applying the Random Forest algorithm), we may lose too much information if we down-sample the majority class here. Instead, if we perform sampling at the place 2 (when growing each tree), we won't lose information if we grow *large enough number of trees* in the forest while keeping the class balance when growing each tree.

The BRF approach performs down-sampling at the place 2 such that number of instances in each class are exactly the same. However, the model in our case is required to have very low FPR, so the exactly balanced ratio (1:1) between normal and abnormal data when growing each tree does not guarantee to have the best performance in terms of the PAUC metric. Hence, we regard the ratio between the normal and abnormal data when growing each tree as a hyper-parameter, and we choose the best ratio that gives the highest PAUC when tuning the parameters with cross validation. In short, we follow the same idea of BRF to adjust the imbalanced ratio when growing each tree, but we don't restrict the ratio to be exactly 1, instead we allow the model to tune the best ratio.

4.1.3 XGBoost (Extreme Gradient Boosting)

The XGBoost, a variant of tree-based gradient boosting model, was proposed by [10]. The idea of gradient boosting came from [13], who found out that the Boosting can be interpreted as an algorithm optimized for a specific cost function (e.g. Adaboost can be viewed as an optimization over the exponential loss function). As discussed by [23], the traditional GBM (gradient boosting machine) model uses gradient descent (first order of Taylor expansion) of the function space, while the XGBoost model uses Newton method (second order of Taylor expansion). Moreover, the XGBoost model employs more effective regularization of individual trees such as the number of terminal nodes and the L2 regularization on the leaf weights in its objective function.

More specifically, the step-wise optimization problem is to find f_t at each step such that it can minimize $\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$, where l is the chosen differentiable loss function, $\hat{y}_i^{(t-1)}$ is the prediction of the previous step, and Ω is the regularization term $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda||w||^2$ (T is the number of leaves and w is the weight vector of the leaves). Using the second order approximation, the above minimization problem can be approximated as follows:

$$\text{Minimize } \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda||w||^2 \tag{4.1}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$. An optimal w can then be solved for each tree structure in this optimization problem, and the value of the above equation can be used to evaluate a given tree structure, hence it can also be used to evaluate the split of each node. A more detailed description is shown in [10].

XGBoost also integrates different instance weights into the model, for example, if we denote the instance weights as w_i, the only difference between the new optimization problem with w_i and the above original optimization problem is that g_i and h_i become $\hat{g}_i$ and $\hat{h}_i$, where $\hat{g}_i = w_i g_i$ and $\hat{h}_i = w_i h_i$, which means that the instance weights are taken into account when finding the best split on each node. In our case where the data is highly imbalanced, we can use cost sensitive approach by utilizing the instance weights in XGBoost, i.e. we can set the weights for all negative instances as 1, and the weights for all positive instances as r. Then we can take this r as parameter and tune the best r according to the evaluation metric (PAUC).

4.2 Sparse Linear Classification Models

In this section, we introduce two common regularization techniques (L1 and L2 regularization) to address the high dimensional problem. And we introduce cost sensitive learning to solve the highly imbalanced problem. Then we show two base linear classifiers (linear SVM and logistic regression) combined with the above two techniques (L1 regularization and cost sensitive learning). Finally, we describe a feature selection method for linear classifiers based on sampling and averaging the feature vector.

4.2.1 L1 and L2 regularization

Regularization in general aims to reduce the overfitting in machine learning model, and it can take different forms in different models, e.g. number of leaves in a tree model. The L1 and L2 regularization are the two most common and widely used regularization terms for linear classifiers. The L2 regularization imposes L2 norm (i.e. sum of squares) constraints on the parameters that need to be optimized, while the L1 regularization imposes L1 norm (i.e. sum of absolute) constraints. The L2 regularization was proposed in the Ridge regression [15], and the L1 regularization was proposed in the LASSO regression [30]. The objective functions of Ridge regression and LASSO regression are shown in the following:

$$\text{Ridge regression (L2 regularization): } min \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2 \tag{4.2}$$

$$\text{LASSO regression (L1 regularization): } min \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j| \tag{4.3}$$

Now to graphically compare the L1 and L2 regularization, we first convert the above minimization problems to their equivalent forms according to the Lagrangian multiplier and KKT condition:

$$\text{Ridge regression (L2 regularization): } min \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 \text{ s.t. } \sum_{j=1}^{p}\beta_j^2 \leq c \text{ for some c} \tag{4.4}$$

$$\text{LASSO regression (L1 regularization): } min \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 \text{ s.t. } \sum_{j=1}^{p}|\beta_j| \leq c \text{ for some c} \tag{4.5}$$

The comparison of L2 regularization and L1 regularization are shown in Figure 4.2. As we can see, the absolute sum constraints imposed by the L1 regularization is much easier to reach the "corners" compared with the square sum constraints imposed by L2 regularization, which means that using L1 regularization is more likely to give sparse models (i.e. some feature parameters are exactly 0, so the corresponding features are completely dropped). Hence, the L1 regularization can be viewed as an internal feature selection method, and we can combine it with different loss functions (such as the logistic loss in logistic regression) in the objective function. In practice, the regularization parameter λ (or shrinkage parameter) controls the trade-off between the magnitude of regularization and the loss from the loss function.

In our case where the data are high dimensional and the model is required to be easy-to-interpret, we consistently use L1 regularization since it can produce sparse models (using just a few features) so that the model is easy-to-interpret, and the trade-off between regularization and classification loss can be adjusted as preferred.

4.2.2 Cost sensitive learning

The objective function that the model tries to optimize usually consists of two parts, i.e. loss function and regularization. The cost sensitive learning is related to the component of loss function in the objective function. Many originally proposed models (e.g. logistic regression) give the same

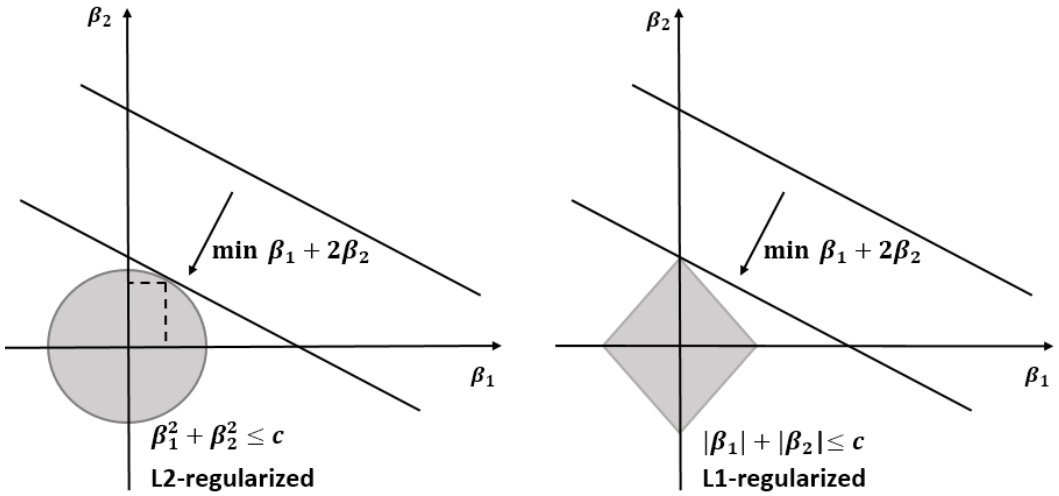


Figure 4.2: Comparison of L2 regularization and L1 regularization. Figure remade from [30]

Table 4.1: Cost matrix for binary classification

	PP (Prediction Positive)	**PN (Prediction Negative)**
GTP (Ground Truth Positive)	C_{TP} (true positive)	C_{FN} (false negative)
GTN (Ground Truth Negative)	C_{FP} (false positive)	C_{TN} (true negative)

costs when the model misclassify an actual positive or an actual negative instance. However, this is usually not true in reality, and the different misclassification costs highly depend on the domains. For example, missing a hacking attack in cyber-security can be much more expensive than misclassifying a normal service request as an attack. Hence, the cost sensitive learning aims to take the different misclassification costs into consideration and integrate the costs in the loss function. The cost matrix for binary classification is shown in Table 4.1. In practice, the costs for true positive and true negative classifications are 0 ($C_{TP} = C_{TN} = 0$) since they are correct. The costs for false positive C_{FP} and false negative C_{FN} are usually different in reality and depend on domain knowledge. When integrating the cost matrix in the loss function, we can decompose the loss function into two parts: one for the loss of misclassifying the actual positives (attached with the cost C_{FN}), and another one for the loss of misclassifying the actual negatives (attached with the cost C_{FP}).

In our case, the transaction data are highly imbalanced and our model is required to have low FPR (FPR $\leq$ 100 per 1 million). We use the cost sensitive learning approach to address the highly imbalanced problem, but we do not have accurate domain knowledge about the misclassification costs. By also considering that we have a requirement of FPR and we use the PAUC as model evaluation metric, we can regard the two costs C_{FN} and C_{FP} as hyper-parameter and tune the best costs according to the PAUC such that the highly imbalanced problem is addressed and also satisfying the FPR requirement.

4.2.3 L1-regularized cost sensitive linear SVM (L1-SVM)

The SVM (support vector machine) is a maximum margin classifier, i.e. if the data are linearly separable, then the model tries to find the best hyper-plane that can separate the two classes of data and has the maximum margin (distances to the two parallel hyper-planes on each side). Mathematically, it can be expressed as $minimize\ \frac{1}{2}||\vec{w}||^2$ subject to $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$ for every i, here minimizing $||\vec{w}||^2$ is equivalent to maximizing $\frac{2}{||\vec{w}||^2}$ (the distances between the two separating hyper-planes). If the data are not linearly separable, some slack variables ϵ_i can be introduced in the constraints: $minimize\ \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\epsilon_i$ subject to $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \epsilon_i$, and $\epsilon_i \geq 0$ for every

i. The above minimization problem with constraints is equivalent to the following minimization problem without constraints by using the hinge loss:

$$\text{minimize } \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n} max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i + b)) \tag{4.6}$$

As discussed in the above two subsections, we use the L1 regularization and cost sensitive approach to tackle the high dimensional and imbalanced problems. More specifically, we change the L2 norm of $\vec{w}$ in the objective function to L1 norm, and we decompose the loss function into two parts corresponding to the false negative and false positive loss. The linear SVM model with L1 regularization and cost sensitive approach is shown in the following:

$$\text{minimize } \sum_{i=1}^{p} |w_i| + C_p \sum_{y_i=1} max(0, 1 - (\vec{w} \cdot \vec{x}_i + b))^2 + C_n \sum_{y_i=-1} max(0, 1 + (\vec{w} \cdot \vec{x}_i + b))^2 \tag{4.7}$$

In this model, we can see that it uses L1 regularization, and it decomposes the loss functions into two parts corresponding to the loss of false negative and false positive ($y_i = 1$ means the instance i is ground truth positive, and $y_i = -1$ means the instance i is ground truth negative). We also use squared hinge loss to penalize more on the severely misclassifying instances, which means that this model focus more on observations with large negative margins. The two parameters C_p and C_n control the relative costs of misclassifying the actual positive and actual negative instances. And the regularization also depends on the magnitude of these two parameters.

4.2.4 L1-regularized cost sensitive logistic regression (L1-LR)

The logistic regression for binary classification expresses the probability of an instance being positive as the logistic function of the variables' linear combination, as follows:

$$\begin{aligned} p(y_i = 1|x = \vec{x}_i) &= \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x}_i + b)}} \\ p(y_i = -1|x = \vec{x}_i) &= 1 - p(y_i = 1|x = \vec{x}_i) = \frac{1}{1 + e^{(\vec{w} \cdot \vec{x}_i + b)}} \end{aligned} \tag{4.8}$$

Hence, the likelihood function can be combined into one: $p(y = y_i|x = \vec{x}_i) = \frac{1}{1+e^{-y_i(\vec{w} \cdot \vec{x}_i + b)}}$. Then maximizing the likelihood function is actually equivalent to minimizing the loss function, as follows:

$$\begin{aligned} &\text{maximize } \log \prod_{i=1}^{n} p(y = y_i|x = \vec{x}_i) = -\sum_{i=1}^{n} \log(1 + e^{-y_i(\vec{w} \cdot \vec{x}_i + b)}) \text{ where } y_i \in \{-1, 1\} \\ &\text{minimize } \sum_{i=1}^{n} L(y_i, f(x_i)) = \sum_{i=1}^{n} \log(1 + e^{-y_i(\vec{w} \cdot \vec{x}_i + b)}) \text{ where } y_i \in \{-1, 1\} \end{aligned} \tag{4.9}$$

Similar as how we integrate the L1 regularization and cost sensitive in the linear SVM, we can add the L1 regularization term and decompose the loss function into two parts, as follows:

$$\text{minimize } \sum_{i=1}^{p} |w_i| + C_p \sum_{y_i=1} \log(1 + e^{-(\vec{w} \cdot \vec{x}_i + b)}) + C_n \sum_{y_i=-1} \log(1 + e^{(\vec{w} \cdot \vec{x}_i + b)}) \tag{4.10}$$

As we can see in the objective function, the two parameters C_p and C_n can control the relative costs of FP and FN, and also control the magnitude of regularization. And we can see that the only difference compared with the above SVM case is the loss function, the above SVM model uses a squared hinge loss while this logistic regression model uses a logistic loss.

4.2.5 Stable feature selection

The above two base linear classifiers can be applied on the dataset after down-sampling the normal data and converting to the one-hot encoding. Here we describe a feature selection technique for linear classifiers. The main idea of this feature selection comes from [29], who applied the L1-SVM model to mine the medical equipment log for predictive maintenance. They selected "stable" features by performing following steps:

1. Draw N random samples (each sample with same number of instances) from the majority class, denote them as M_i $(i = 1, 2, \ldots, N)$
2. Combine the above samples with all the minority data respectively. If we denote the minority data as D, then the combined samples are $C_i = M_i \cup D$
3. Train the L1-regularized linear classifiers on each C_i, and denote the corresponding feature vector as $f_{i,j}$ (the jth feature of the ith model, $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, p$)
4. Take the sum of all the feature vectors, and denote the sum of feature vector as F (so we have $F_j = \sum_{i=1}^{N} f_{i,j}$)
5. Only keep the top X features with large enough absolute values of F_j (i.e. sort the values of $|F_j|$ and only keep the largest X ones)
6. Finally train the model on the final training set using only the features kept in the above step

We can explain that this stable feature selection technique is reasonable because of the following reasons:

1. If the model is only trained over one sample that only accounts for a small proportion of the total dataset, there is sampling bias such that the sample does not reflect the real behavior of the total population, which inevitably causes our model to learn this sampling bias. By drawing multiple samples and training multiple models in the above way, we can reduce the sampling bias.
2. The sign of the feature weights of linear classifiers (like the L1-SVM) can be interpreted as making the instance more likely to be in the positive or negative class. The features with unstable signs (sometimes positive and sometimes negative) are considered to be noisy, since the effect of the features on the classification score should be either positive or negative consistently. Taking the sum of the feature vectors can filter out the features with unstable signs, since if the feature is sometimes positive and sometimes negative, it's likely to sum up to 0 in the combined feature vector.
3. The magnitude (absolute values) of the feature weights reflects the feature's influence on the final classification score. Hence, the features with small absolute values can be filtered out since they have less influence on the model compared with the features with larger absolute values.

In our case where we are using one-hot encoding for our features, we may further improve this feature selection idea to make it more suitable for our model. We only change the step 4 and 5 in the above feature selection technique as follows:

- In step 4, we also calculate the standard deviation for each feature among the feature vectors, and we denote the vector of the standard deviation for each feature as SD (so we have $SD_j = sd\{f_{1j}, f_{2j}, \ldots, f_{Nj}\}$).
- In step 4, since the original calculation ($|F|$ as absolute sum) is for the features at the one-hot encoded level, we also calculate the OF_j at the original feature level as the sum of $|F_i|$ who belongs to the original feature j. So we have $OF_j = \sum_{i \in j} |F_i|$, where $i \in j$ means the i one-hot encoded feature belongs to the j original feature.

- In step 5, the one-hot encoded features with large enough absolute values F and small enough standard deviation SD are kept. And the original features with too small OF values are completely filtered out.

As we can see in the above 3 changes, we consider the "stable" features as having large enough absolute sum values and small standard deviation, so the features that vary a lot across different samples are considered to be not stable and should be filtered out. Furthermore, we adjust this feature selection technique according to our one-hot encoding setting, i.e. we filter our the features not just based on the one-hot encoded feature level, but also the original feature level, which can enable us to have more explicit controls over the number of original features that are used in the final model.

Chapter 5

Clustering-based Ensemble of Local Sparse Linear Classifiers

In this chapter, we describe a novel clustering-based ensemble approach to improve the predictive performance of single linear classifiers for binary classification problems where the two main classes may have well-separated sub-classes, and we show that the final ensemble model is still easy to interpret if the obtained clusters have meaningful interpretations. In short, our approach is to first perform clustering *within* each class (the abnormal and normal class), then train multiple local sparse linear classifiers using the "Class Crossing One-vs-All" scheme, and finally combine the classification scores from the local classifiers based on the instance's distances to different cluster centers.

In the following sections, we first illustrate the intuitions and the main idea of this approach, then we describe each component of this approach, i.e. clustering schemes, clustering algorithms, base linear classifiers, classifier training schemes, score pre-processing before combining the classifiers, and classifier combination schemes. Finally, we show an example of such ensemble approach and review related work that are similar to this clustering-based ensemble approach.

5.1 Intuitions

As we can see in the experiment section, the "stable feature selection" technique described in the previous section indeed can improve the predictive performance of a single sparse linear classifier. However, its predictive performance is still not good enough compared with some state-of-art classification techniques such as Random Forest. The main limitation of a *single* linear classifier is its poor performance when the data are not linearly separable to a large extent, which is quite common in reality (also in our case). Furthermore, in this binary classification problem setting where the labels are quite coarse (as we have discussed about the challenges of the data in Chapter 1), the transactions of each main class may have more fine-grained sub-classes that are very different from each other, so the data of each class may form complex regions, which may make it even harder to separate the two classes by a single linear classifier (in essence a single hyper-plane in the feature space). In contrast, the Random Forest can deal with those complex regions quite well, since the base model (i.e. decision tree) is designed to identify the complex regions by dividing the feature space into axis-parallel hyper-rectangles and the bagging ensemble can reduce the overfitting problem that a single decision tree may encounter.

In terms of interpretability[1], a single decision tree has excellent interpretability, but a Random Forest (an ensemble of many trees) makes it harder to interpret how a decision is being made due to the randomness (instance sampling and feature subset sampling) included when growing

[1]It is not our focus to compare different models in terms of interpretability, because interpretability is rather subjective, and it remains an open question to measure interpretability [3]

each tree, and such randomness is the key point to help the model avoid over-fitting. Although we can calculate some feature importance indexes, such as the average Gini index reduction for each feature, the exact behavior and impact of each feature are not clear due to the complex interactions (lots of deep trees in the Random Forest model) between the features. On the other hand, a single linear classifier also has excellent interpretability. In our one-hot encoding setting for linear classifiers (e.g. linear SVM), the feature weights can be simply interpreted as the effect of the categorical values on the classification scores. If a one-hot encoded feature has a positive (negative) weight, then it means the presence of that categorical value makes the instance more (less) likely to be abnormal, and the magnitude (absolute value) of the feature weight implies the magnitude of the feature's influence on the classification score. However, a single linear classifier does not perform well on complex data that are not linearly separable.

Hence, we want to build an ensemble of a small number of linear classifiers such that the final ensemble model has higher predictive performance than a single linear classifier while keeping the final ensemble model easy to interpret, which means that we try to draw several "reasonable" lines (or hyper-planes) instead of just one line (or hyper-plane) in the feature space. This small set of linear classifiers should also relate to the fact that sub-classes may exist in each main class. If each of those linear classifiers can be interpreted as a "specialized" classifier that aims to identify a specific sub-class, and the combination of those local classifiers is based on some locality information (e.g. distance information), then the final ensemble model is possible to have good interpretability. The idea of "clustering-based ensemble of local sparse linear classifiers" is illustrated in Figure 5.1. As we can see in the figure, the "+" represents abnormal data (in

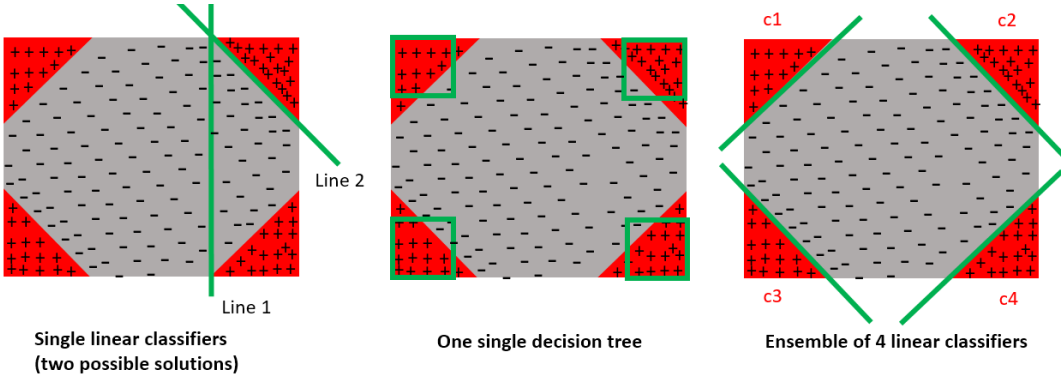


Figure 5.1: Illustration of the clustering-based ensemble idea, where + is abnormal, and - is normal data

red color), the "-" represents normal data (in grey color), and our binary classification task is to train a model to separate the abnormal and normal ones. In this imagined scenario, there are 4 sub-classes (clusters) in the abnormal class, they are well separated from each other and located in the four corners. If we consider 3 models (i.e. one single linear classifier, decision tree, and ensemble of linear classifiers) in the above scenario, they have different behaviours respectively as follows:

1. If the model is just one single linear classifier, which is equivalent to drawing one single line in a two-dimensional feature space, there is no perfect solution that can separate all the "+" from the "-" without any misclassification. For example, if we draw the "line 1" (as shown in the figure), we can correctly classify two "+" corners on the right side, but also misclassify two corners of "+" and some "-" region. If we draw the "line 2", we can classify one "+" corner and all the "-" data correctly, but also sacrifice all the other three "+" corners. As we have discussed before, the cost sensitive approach may prefer line 2 over line 1 if the costs of misclassifying a normal instance is very high, and in our case where there is a FPR requirement on the model, we may also prefer the line 2. All in all, in this situation, one single linear classifier is not good enough due to the complex regions formed by the four

well-separated sub-classes of the abnormal data.

2. If the model is a decision tree, which is able to divide the two-dimensional feature space into axis-parallel rectangles, then we may end up with classifying all the four rectangles located in the four corners as abnormal. Again, the volume of the four rectangles (a larger rectangle can include more TP but also more FP) depends on the misclassification costs in the cost sensitive learning. In our case where the FPR is required to be very low, the four rectangles are likely to be small. All in all, the decision tree can do a reasonably good job in terms of the predictive performance, but as we have discussed before, one single tree can get overfitting easily in reality, and the Random Forest becomes harder to interpret compared with the single decision tree model.

3. If we can train and combine several "reasonable" linear classifiers instead of just one single linear classifier, we can imitate the behavior of a decision tree to improve the predictive performance. The question left is how to train and combine several linear classifiers "reasonably". In this imagined scenario, our solution is to first perform clustering in the abnormal class, and we get 4 clusters located in each of the four corners. Then for each pair of the abnormal cluster and all of the normal data (each pair of the corner and the center region), we train a local linear classifier (preferably also sparse) separately, which results in 4 local linear classifiers. When a new data instance comes in, we evaluate this instance using all of the 4 classifiers, then combine their classification scores based on the instance's distances to each of the 4 abnormal cluster centers. In this way, we can improve the predictive performance since the sub-problems (each of the 4 local classifiers) created by performing a suitable clustering within each main class become more linearly separable and the combination can be viewed as distributing different levels of trust to local experts based on the locality information of the instances. Furthermore, if the obtained clusters have meaningful interpretations (e.g. corresponding to different types of abnormal transactions), each of the local classifiers can be interpreted as specialized in identifying a certain sub-class, which makes each of them easy to interpret. And we can keep the final ensemble model still easy to interpret by interpreting the combination as giving different levels of trust to those local specialized classifiers based on distance information. In short, we train the local linear classifiers "reasonably" by performing clustering to properly divide each main class and then train the classifiers on the clustering results using a proper scheme ("Class Crossing One-vs-All" as we explain later), and we combine their results "reasonably" by exploiting their distances to the cluster centers ("Soft Combination" scheme as we explain later).

In the above imagined scenario, our approach (i.e. clustering-based ensemble of local sparse linear classifiers) can give a perfect solution without making any misclassification, and the final ensemble model is still easy to interpret. Although it's an imagined scenario, the fact that the two main classes (the abnormal class and normal class) may contain sub-classes that are very different from each other encourages us towards this clustering-based ensemble approach, and the sub-problems created by performing a suitable clustering within each main class should become more linearly separable. The experiment results on the transaction dataset also supports our approach by showing improvement in predictive performance while still being relatively easy to interpret, although the interpretability of the final ensemble model also highly depends on the clustering results. We formulate the main idea in the following section.

5.2 Main Idea

The main idea of our approach (i.e. clustering-based ensemble of local sparse linear classifiers) can be decomposed into several components (e.g. clustering schemes, classification schemes). The components and the workflow of the approach are shown in Figure 5.2. As we can see in the

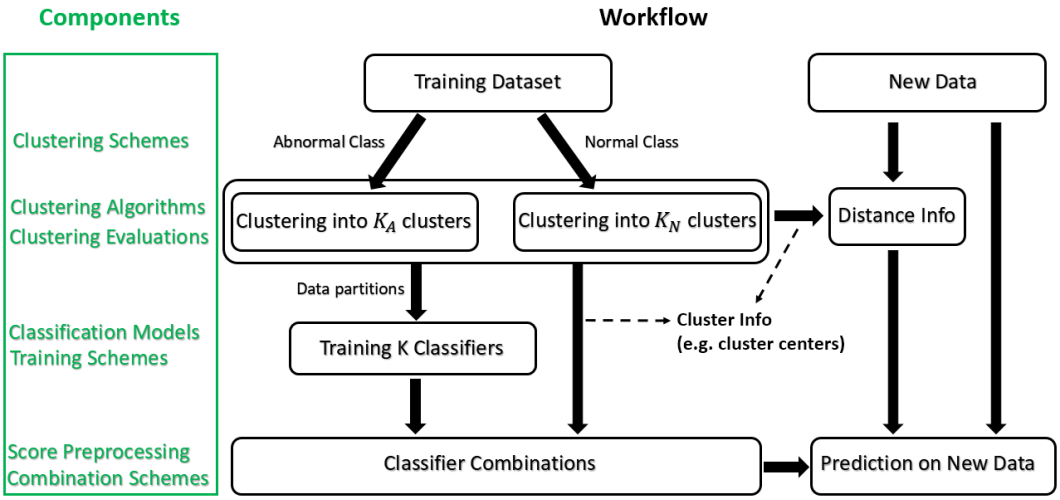


Figure 5.2: Components and workflow of the clustering-based ensemble approach

workflow, firstly the training set is split by the main class labels, and then we perform clustering inside each main class. This step is related to the clustering schemes in the components, since besides "clustering inside each class" we can also perform clustering in the whole training set (mixed class). When performing the clustering, we need to choose a suitable clustering algorithm and evaluation metric. Then the next step is to train local classifiers based on the clustering results (i.e. partitions of the data), which is related to the classification models and classifier training schemes in the components (e.g. we can choose different base classifier models and different training schemes). Finally, we need to combine the local classifiers based on the clustering information (e.g. cluster centers), which is related to the score pre-processing (usually we need to pre-process the scores from different local classifiers) and combination schemes (i.e. the rules to combine the classification scores from the local classifiers) in the components. After we create such an ensemble model, we can make predictions on new data based on the model and the distance information (i.e. the instance's distances to different cluster centers).

In the following sections, we describe each component in more details (i.e. clustering schemes, clustering algorithms and evaluation, classification models, classification training schemes, score pre-processing and classifier combination schemes). Then we show the pseudo code as an example of our clustering-based ensemble approach. Finally, we review and discuss related work that are similar to this clustering-based ensemble approach.

5.3 Clustering Schemes

The reason that we perform clustering is to find out the local structure of the data (e.g. different sub-classes inside each main class) such that the sub-problems created by the clustering results (i.e. partitions of the data) become more linearly separable, which should in return improve the predictive performance of the original binary classification task. Depending on our later choices of clustering algorithms and classification models, there are two main types of clustering schemes:

- Clustering the whole training dataset (data of mixed classes)
- Clustering inside each main class separately

The first scheme (i.e. Clustering the whole training dataset) has the advantage that the discovered locality is directly related to both main classes (e.g. the abnormal transactions and normal transactions are close to each other in a cluster trained by this scheme), so we may just build a local classifier for each of the discovered cluster that may contain data of both main classes. However, this is also its disadvantage since the data of both main classes have strong locality in one cluster does not mean the (linear) classifier can separate them well. And we can imagine if a linear classifier can separate the data of both main classes very well, the data of both main classes do not have to exhibit locality at all (the two main classes do not have to be close to each other in the feature space). Another disadvantage of the first clustering scheme is that the ratio between the two main classes in each cluster can be even more imbalanced than the original problem and give more challenges for the classification task.

We only focus on the second scheme (i.e. Clustering inside each main class) in our approach since it can discover different clusters inside each main class, and the classifiers trained later can be interpreted as "specialized" classifiers if the obtained clusters have meaningful interpretations.

5.4 Clustering Algorithms and Evaluations

Here we discuss some clustering algorithms, dissimilarity measures and evaluation metrics that are closely related to our clustering-based ensemble approach, and we refer to [36] for a more detailed survey of clustering.

1. Clustering algorithms
   - K-means
   - K-medoids
   - K-modes
   - Self-organizing map (SOM)
2. Dissimilarity measures, between points $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$
   - Euclidean distance: $\sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$
   - Manhattan distance: $\sum_{i=1}^{n}|p_i - q_i|$
   - Hamming distance: $\sum_{p_i \neq q_i} 1$ (number of distinct value pairs)
3. Clustering evaluations
   - External index: Adjusted Rand Index
   - Internal index: the average Silhouette index
   - Others: cluster instances distribution, time distribution, and interpretations by domain experts

Clustering algorithms

We can see that most of the listed clustering algorithms are hard partitional (i.e. each data instance can only belong to one cluster), which makes the later classifier training task much easier. The K-means is the most widely used and simplest, it first randomly selects K points as cluster centers and iteratively updates by assigning the closest points and calculating the mean points as new cluster centers until it converge. Compared with K-means, the K-medoids only considers actual data instances as cluster centers instead of the mean points, and it is usually more robust to noise. The K-modes is a variant of K-means designed for clustering categorical data [16], since the K-means is based on Euclidean distance and it is not suitable for categorical data. The K-modes updates the clusters in a similar manner, but it calculates the modes of the categories instead of the mean points as cluster centers. All of the above three algorithms (i.e. K-means, K-medoids

and K-modes) are hard partitional and based on cluster centers, which perfectly fits our ensemble approach. Similarly, the SOM can also be viewed as a clustering algorithm to partition the data based on cluster centers (node centers). The difference between SOM and the previous algorithms is that SOM uses a neural-network-like manner to update the centers [19].

Besides the above four clustering algorithms that are all hard-partitional and center-based, we have also considered some alternative algorithms, e.g. Fuzzy C-means, DBSCAN, and subspace clustering. The Fuzzy C-means can be viewed as a soft-partitional version of K-means, which means that it allows each data instance to belong to different clusters (to different extents) and it is also center-based. If in later stage we want to train classifiers based on Fuzzy C-means clustering, we can use the cluster memberships of an instance as its weights and we need classification models that accept the instance weights as inputs. However, we prefer hard partitions of the data since we want a model that is simple and easy to interpret, so we do not use Fuzzy C-means in our ensemble approach. Different from all the above five clustering algorithms based on centers, the DBSCAN is also hard-partitional but based on density, it can group regions with high density (defined by the minimum points in a neighborhood) into clusters. The main advantage of DBSCAN is that it can find out arbitrary shape of clusters, whereas the algorithms like K-means are biased towards finding clusters of spherical shapes. Subspace clustering is also related to our ensemble approach in terms of its ability to find out not just the clusters, but also the subspace attached to each cluster. If we can establish a classifier training scheme to only train in the subspace attached with the clusters, we can easily meet the requirement of a sparse model. However, the subspace attached with the clusters are not guaranteed to be useful for classification, and also many subspace clustering algorithms are not hard partitional, so we do not focus on subspace clustering in our ensemble approach.

Dissimilarity measures

As for the dissimilarity measures, we consider 3 distances, i.e. Euclidean distance, Manhattan distance and Hamming distance. For example, we may use Euclidean distance for K-means, Manhattan distance for K-medoids, and Hamming distance for K-modes. The Manhattan distance is more robust to noises compared with Euclidean distance since it uses the L1-norm, which can reduce the effects of noises. The Hamming distances are used for categorical features, since the numerical distance notion is usually not applicable for categorical values. The Euclidean and Manhattan distance are still applicable in our case considering that our categorical features are encoded in float numbers in the range $(0, 1)$ such that there are some orderings for numerical categories preserved (e.g. 10 to 0.1, 100 to 0.2 and 1000 to 0.3). And as we can see in the experiment section, the clustering results of K-means (Euclidean distance) and K-modes (Hamming distance) on the transaction dataset are very similar in our case.

Clustering evaluations

As for the clustering evaluations, external indexes refer to the evaluation metrics that require the ground-truth labels, and the internal indexes refer to the ones that do not require ground-truth labels and usually calculated based on the internal structure or characteristics of the clusters (e.g. within-cluster distances). External indexes require the ground-truth labels that we usually don't have in reality, while the internal indexes can be biased towards a certain type of algorithms. Evaluating the clustering results is not an easy task considering that our clustering scheme is "clustering inside each main class" and we don't have subclass-labelled data for external evaluations. However, we can still use the main class labels with external index like ARI (Adjust Rand Index, defined in Appendix A.1) for evaluating the clustering results of *mixed data* in order to choose suitable clustering algorithms that can perform well on our data. More specifically, we can combine the abnormal data and some random samples of the normal data in the training set, keep their main class labels, perform clustering by a chosen algorithm and parameter (like cluster number K in K-means), and evaluate the clustering results using external index ARI with their main class labels. A clustering algorithm that is applicable to our data should roughly separate

the abnormal and normal data, which usually results in a higher ARI value. After we have chosen a suitable algorithm, we can perform the clustering inside each main class and use internal index like the average Silhouette index to evaluate, which is used to measure how tightly grouped the data in the clusters are. More specifically, we can use the average Silhouette index (defined in Appendix A.2) to choose the number of clusters (K) when performing K-means inside each main class.

Besides the above two metrics, we can also use another two metrics that are more related to realistic considerations: the *instance distribution* and *time distribution* in all obtained clusters. In our ensemble approach, we prefer clusters inside each main class that have balanced number of instances in order to avoid the too imbalanced problem for later classifier training, and we also prefer the time distribution within each cluster to be evenly spread (i.e. the time-stamps of the instances in each cluster should be widely and evenly spread in the total time span, but not only limited to a specific time), since the abnormal transactions in a very time-specific cluster may not happen again in the future.

All the above four metrics (i.e. the external index ARI, internal index average SI, instance distribution, and time distribution) can be easily quantified and measured. There is another metric, i.e. *interpretation by domain experts*, that is difficult to quantify and measure. However, as we show in the experiment section, the interpretability of the final ensemble model highly depends on the interpretations of the obtained clusters. The "Class Crossing One-vs-All" classifier training scheme that is described in the following sections can guarantee that each local classifier can be interpreted as "specialized" in identifying a certain obtained *cluster*, but the obtained clusters do not necessarily have meaningful and practical interpretations (e.g. different types of abnormal transactions). Hence, if we want to make the final ensemble model easy to interpret, we should also seek help from domain experts to check if the obtained clusters have meaningful interpretations.

5.5 Classification Models

As mentioned in Section 5.1 where we illustrated the intuitions of the clustering-based ensemble approach, we want to train and combine a small set of *linear* classifiers to imitate the behaviour of a decision tree, so we only consider linear classifiers (preferably sparse) as the base classification models in our ensemble approach. We consider the following two linear classifiers as shown in Section 4.2.3 and 4.2.4:

1. L1-regularized cost sensitive linear SVM

2. L1-regularized cost sensitive logistic regression

The L1 regularization is for internally selecting the features such that each local classifier is *sparse* enough and the total number of used features after combining all the local classifiers is not too large. On the other hand, the cost sensitive approach used in the two base linear classifiers aims to address the imbalanced problem that may have different imbalanced ratios when training local classifiers over different pairs of obtained clusters.

The linear SVM is a maximum-margin classifier, while the logistic regression models the probability of an instance being positive as the logistic function of the variables' linear combination. Both of them can output classification scores that can *indicate* the confidence of the classification. The classification scores of the linear SVM model can be interpreted as the distances from the classification hyper-plane (boundary), whereas the classification scores of the logistic regression can be viewed as the estimated probability of being positive. The different meanings of the classification scores in the base classification models drive the needs of pre-processing the scores before combining the classifiers. And we show in the experiment section that even if we use the same base classification model, the score pre-processing is still necessary.

5.6 Classifier Training Schemes

Now we need to choose a suitable classifier training scheme after we have performed the clustering inside each main class and selected a base classification model. Let's only focus on the hard-partitional and center-based clustering algorithms (e.g. K-means, K-medoids, K-modes and SOM). Assume that we have k^+ clusters ($k^+ \geq 1$) for the abnormal class (each denoted as $A_1, A_2, \ldots, A_{k^+}$) and k^- clusters ($k^- \geq 1$) for the normal class (each denoted as $N_1, N_2, \ldots, N_{k^-}$), so we have $k^+ + k^-$ hard partitions of the training dataset, and each of these partitions has a label attached to them (either abnormal or normal). We also denote the set of all abnormal clusters as A (so we have $A = \cup_{i=1}^{k^+} A_i$), the set of all normal clusters as N (so we have $N = \cup_{i=1}^{k^-} N_i$), and the set of all clusters as $T = A \cup N$. We also denote $T_i = A_i$ for $1 \leq i \leq k^+$ and $T_j = N_j$ for $(k^+ + 1) \leq j \leq (k^+ + k^-)$.

A classifier training scheme is a manner of training a set of classifiers on different combinations of the above obtained $k^+ + k^-$ data partitions. Some common classifier training schemes and our proposed scheme "Class Crossing One-vs-All" are shown in the following:

1. **One-vs-All** (binary): for each cluster $T_i \in T$, we train a classifier on T_i (as positive) and all the rest clusters $T \setminus T_i$ (as negative), i.e. A_1 as positive and $\{A_2, \ldots, A_{k^+}, N_1, \ldots, N_{k^-}\}$ as negative, and so on. So in this scheme we need to train $k^+ + k^-$ classifiers in total.

2. **One-vs-One** (binary): for each cluster pair $T_i, T_j \in T$ and $i < j$, we train a classifier on T_i (as positive) and T_j (as negative), i.e. A_1 as positive and A_2 as negative, A_1 as positive and A_3 as negative, and so on. So in this scheme we need to train $\frac{(k^+ + k^-)(k^+ + k^- - 1)}{2}$ classifiers in total. The classifiers that are trained over the clusters from the same main class (e.g. A_1 vs. A_2) can be ignored considering that our final task is just binary classification but not necessary to predict the specific types.

3. **Direct extension to multi-class** (multi): this scheme is only possible for the classification models that can be naturally extended for multi-classification, e.g. multinomial logistic regression. Instead of training binary classifiers (positive and negative), it can train multi-classifier on all the clusters at once using the cluster's labels as classification labels, i.e. $A_1, A_2, \ldots, A_{k^+}, N_1, N_2, \ldots, N_{k^-}$. So in this scheme we only need to train 1 multi-classifier.

4. **Class Crossing One-vs-All** (binary): we assume at least one of k^+ or k^- greater than 1 (otherwise we can only train 1 global classifier). For each $A_i \in A$ ($1 \leq i \leq k^+$), we train a binary classifier on A_i (as positive) and all normal clusters N (as negative). And for each $N_i \in N$ ($1 \leq i \leq k^-$), we train a binary classifier on all abnormal clusters A (as positive) and N_i (as negative). In the case where $k^+ = 1$ or $k^- = 1$, a global classifier is trained (A as positive and N as negative). So in this scheme we need to train $k^+ + k^-$ classifiers in total.

We can also represent the above binary schemes using a so-called coding matrix. The coding matrix representation of transforming a multi-class problem to binary classification problem was first proposed by [11], who restricted the elements of the matrix to be $\{+, -\}$ (it means that every subset has to be used for training, either labelled as positive or negative). The coding matrix was later extended by [1], who allowed the elements of the matrix to be $\{+, 0, -\}$, which means that some subsets can be ignored and not included in the training (when the element is 0). If we assume $k^+ = 3$ and $k^- = 2$, the coding matrix for "One-vs-All", "One-vs-One", and "Class Crossing One-vs-All" are shown in Table 5.1, 5.2, and 5.3 respectively. Each row of the code matrix represents each cluster, and each column represents the training of one binary classifier. For example, in Table 5.3, the first column means that the classifier f_1 is trained over A_1 (as positive) and all normal clusters (N_1 and N_2 as negative), ignoring the other two abnormal clusters A_2 and A_3. We can see that the code matrix provides a very clear representation of different binary classifier training schemes.

Table 5.1: Code matrix representation for One-vs-All scheme

	f_1	f_2	f_3	f_4	f_5
A_1	+	-	-	-	-
A_2	-	+	-	-	-
A_3	-	-	+	-	-
N_1	-	-	-	+	-
N_2	-	-	-	-	+

Table 5.2: Code matrix representation for One-vs-One scheme

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
A_1	+	+	+	+	0	0	0	0	0	0
A_2	-	0	0	0	+	+	+	0	0	0
A_3	0	-	0	0	-	0	0	+	+	0
N_1	0	0	-	0	0	-	0	-	0	+
N_2	0	0	0	-	0	0	-	0	-	-

Now we analyze the advantages and disadvantages of each classifier training scheme in our clustering-based ensemble approach as follows:

1. **One-vs-All** (binary): this scheme is the simplest and most widely used for reducing the multi-class problem to binary classification, it only needs to train a relatively small number of classifiers. However, there are a few disadvantages of this scheme: firstly, every local classifier trained by this scheme is hard to interpret, since in each local classifier there are clusters of the same class on both sides (e.g. A_1 is positive and A_2, A_3 are negative as shown in Table 5.1), but our ultimate task is only to classify whether the instance is abnormal (or normal). Secondly, the highly imbalanced problem becomes even more severe for every local classifier since there is only one cluster on one side and all the other clusters on the other side. Lastly, this scheme is hard to incorporate the distance information (instance's distances to different cluster centers), since every local classifier is trained over the whole training set (with different data partitions and training labels), for example, if an instance has the smallest distance to the cluster center of A_1, we cannot give preference over the local classifier f_1 since other classifiers such as f_2 is also trained over A_1 (although with different labels).

2. **One-vs-One** (binary): as mentioned above, we can disregard the classifiers trained over the clusters of same class (e.g. A_1 as positive and A_2 as negative) since our task is only binary classification of the two main classes. After removing them, every classifier left is trained over a cluster of one main class (as positive) and a cluster of another main class (as negative). The main disadvantage of this scheme is that we need to train a relatively large number of local classifiers even after removing the useless ones, for example, if we assume $k^+ = 5$ and $k^- = 5$, then we need to train 25 local classifiers. Moreover, each local classifier is also not easy to interpret, since they are trained over only one cluster of one main class

Table 5.3: Code matrix representation for Class Crossing One-vs-All scheme

	f_1	f_2	f_3	f_4	f_5
A_1	+	0	0	+	+
A_2	0	+	0	+	+
A_3	0	0	+	+	+
N_1	-	-	-	-	0
N_2	-	-	-	0	-

against one cluster of another main class.

3. **Direct extension to multi-class** (multi): this scheme is not applicable to every base classification model since it requires the model to be able to deal with multi-class directly. Some classification models can deal with multi-class naturally such as decision trees, some have direct extensions to multi-class such as the multinomial logistic regression and artificial neural network, but some classification model do not have such simple and direct extensions such as the SVM. Hence, we do not focus on this scheme.

4. **Class Crossing One-vs-All** (binary): this scheme has the following advantages compared with the above common schemes

   - Firstly, the local classifiers trained by this scheme are easy to interpret, i.e. each of them can be interpreted as *specialized* in identifying the corresponding cluster from the another main class, since on one side there is one cluster of one main class and on the other side there are all the data of the other main class. For example, as shown in Table 5.3, the local classifier f_1 is trained on A_1 as positive and all the other normal clusters (N_1, N_2) as negative, so this local classifier can be interpreted as a specialized classifier that can distinguish the abnormal cluster A_1 from all the normal data.
   - Secondly, we only need to train a relatively small number of classifiers $(k^+ + k^-)$ by this scheme, for example, if we assume $k^+ = 5$ and $k^- = 1$, then we only need to train 6 classifiers, and if we assume $k^+ = 5$ and $k^- = 5$, then we only need to train 10 local classifiers. Such a small number of local classifiers can also help to improve the interpretability of the final ensemble model.
   - Thirdly, this scheme is *additive* and *recyclable*. For example, assume that as a starting point we only want to perform clustering inside the abnormal class and keep the normal class as a whole (assume $k^+ = 3$ and $k^- = 1$), then we only need to train 4 classifiers (3 local classifiers and 1 global classifier). If in a later stage we also want to perform clustering inside the normal class (such that $k^- > 1$), we can just add the new classifiers corresponding to the new normal clusters and keep the existing 3 classifiers corresponding to the 3 abnormal clusters unchanged. Furthermore, if we assume that we already have 5 local classifiers for $k^+ = 3$ and $k^- = 2$, and now we want to perform a new clustering within the normal class such that $k^- = 3$, we can still keep the 3 classifiers corresponding to the $k^+ = 3$ and add the new ones corresponding to $k^- = 3$. This additive and recyclable feature can reduce the burden of model updating. In contrast, all the above common schemes (except the One-vs-All scheme) need to train completely new classifiers to replace all the old ones when there is any change on the clusters.
   - Lastly, this scheme is much easier to incorporate the distance information as we explain in the following section of combination schemes.

5.7 Classifier Score Pre-processing

Now assume that we have trained the local classifiers based on the Class Crossing One-vs-All scheme and the clustering results, then we can apply these local classifiers on any data instance and get a classification score for each local classifier, which can indicate the confidence of the classifications. Before combining the scores of local classifiers, we should pre-process the scores since they are not guaranteed to be in the same scale and usually they do not have the same mapping to the posterior probability. It's important to note the difference between the classification scores and the posterior probability: the classification scores can only indicate the confidence of classification but do not directly correspond to the posterior probability (except the classification models like logistic regression that explicitly express the probability estimates as scores). For example, the classification scores of the SVM model corresponds to the instance's distance to the classification hyper-plane (the decision boundary), and the higher the margin $y_i score_i$ is, the more confident

the classification is. The classification scores from different types of classifiers can have totally different meanings and scales, and even the classifiers that are of the same type but trained over different partitions of the training set can have different scales of classification scores. Hence, we can consider several score pre-processing techniques as follows:

- Simple scaling
  - Standardization: $\hat{s}_i = \frac{s_i - mean}{sd}$
  - Min-max scaling: $\hat{s}_i = \frac{s_i - min}{max - min}$
- Probability calibration
  - Platt's scaling (fits a sigmoid function)
  - Isotonic regression (fits an isotonic function)

For the simple scaling techniques, the standardization re-scales the scores to new scores with $mean = 0$ and $sd = 1$. The min-max scaling simply maps the original scores to new scores in the range of $[0, 1]$. It's important to note that we should not use the testing set for the the estimations of $mean, sd$ in standardization and min, max in the min-max scaling. In our experiment, we use the last three weeks of the training set (around 28 million rows) to estimate the above statistics.

For the techniques of probability calibration, the Platt's scaling proposed by [25] fits a sigmoid function to the scores, i.e. $P(y_i = 1|s_i) = \frac{1}{1+\exp(\beta_1 s_i + \beta_0)}$, we can think of it as fitting a logistic regression from the scores to the probability estimates. The Platt's scaling method is especially effective for models that have a sigmoid-like mapping function from scores to probability, e.g. the SVM model. For other models that the mapping function shape is unknown, it is suggested to use isotonic regression to fit a non-decreasing constant step function from scores to probability estimates [37]. A graphical comparison of Platt's scaling and Isotonic regression is shown in Figure 5.3.

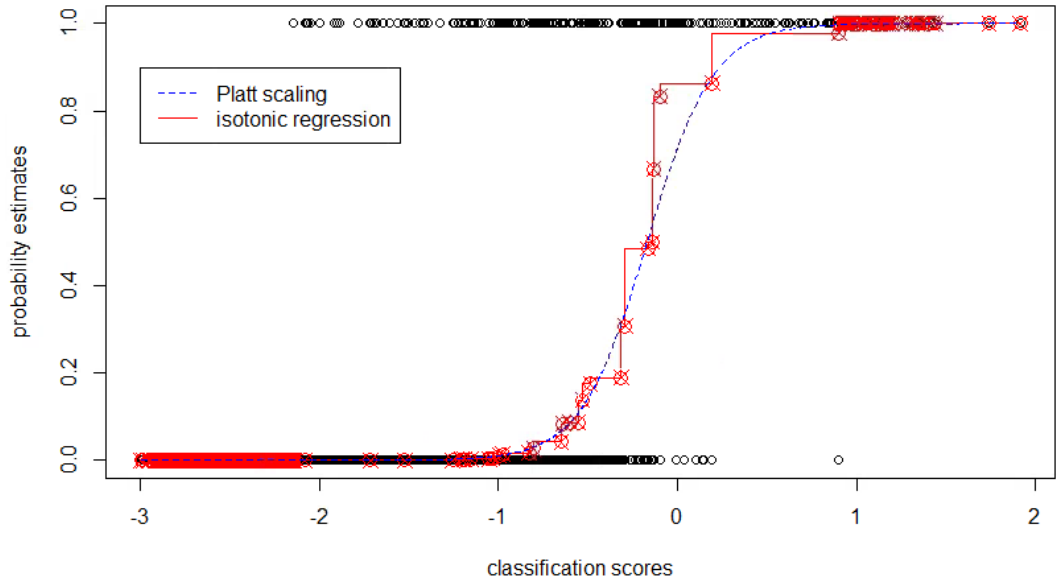


Figure 5.3: Comparison between Platt's scaling and Isotonic regression

As discussed by [22], the fitting of the sigmoid function or isotonic regression should be based on a calibration set, i.e. we should separate an independent set from the training set and use it for calibration to avoid a poor probability estimation. In our case, we use the last three weeks of the training set (around 28 million rows) to fit the calibration methods. This should provide a good

probability estimation since our model training set (e.g. 10K rows for training the SVM model) only uses a small down-sampling set of normal data in the training set (around 120 million rows), so we include lots of extra normal data that are out-of-bag (not included in the model training set). Furthermore, this large dataset for calibration can also provide a realistic ratio between abnormal data and normal data, which is impossible to achieve if we take the calibration set from the model training set.

5.8 Classifier Combination Schemes

After we have pre-processed the scores from different local classifiers, they should have approximately the same scale and/or same meaning as probability estimates (by probability calibration). Now we consider different schemes to combine the local classifiers. As shown in the following section of related work, many existing approaches that have similar idea to ours use simple rules to combine local classifiers like majority votes, which ignore the magnitude of the scores from local classifiers. In our approach, considering the model requirement to output (nearly) continuous scores (as discussed in Section 2.3), we only focus on the combination schemes that can combine the scores of local classifiers into one single score that can indicate the confidence of the combined classification. Furthermore, we incorporate the distance information (i.e. an instance's distances to different cluster centers) into the combination. In the following, first we introduce a method to calculate cluster memberships based on the distances to the cluster centers, then we show different combination schemes for the "Class Crossing One-vs-All" training scheme.

Given an instance's distances to different cluster centers $D = \{d_1, d_2, \ldots, d_n\}$, we can calculate the instance's membership $m(i, D)$ belonging to cluster i as follows:

$$m(i, D) = \frac{(\frac{1}{d_i})^m}{\sum_{d_j \in D}(\frac{1}{d_j})^m} \tag{5.1}$$

where m is called fuzzy membership degree (larger m results in crisper and less fuzzier memberships). This membership calculation is the same as the membership degree in Fuzzy C-means [2], it has several properties: the memberships sum up to 1 (i.e. $\sum_{i=1}^{n} m(i, D) = 1$), and a smaller distance results in a larger membership.

Assume that there are k^+ abnormal clusters and k^- normal clusters obtained by a clustering algorithm that is center-based (e.g. K-means). We denote the cluster centers of abnormal clusters as $a_1, a_2, \ldots, a_{k^+}$ and the centers of normal clusters as $n_1, n_2, \ldots, n_{k^-}$. Based on the "Class Crossing One-vs-All" scheme, we denote the local classifier corresponding to each abnormal cluster as $f_{a_1}, f_{a_2}, \ldots, f_{a_{k^+}}$ and each normal cluster as $f_{n_1}, f_{n_2}, \ldots, f_{n_{k^-}}$, we also train a global classifier on the whole training set (all the abnormal clusters as positive and all the normal clusters as negative), and we denote this global classifier as f_g. Now assume that we have a data instance, we can apply the local classifiers on the instance and pre-process the scores, and we denote the scores after pre-processing as $S^+ = \{s_{a_1}, s_{a_2}, \ldots, s_{a_{k^+}}\}$ and $S^- = \{s_{n_1}, s_{n_2}, \ldots, s_{n_{k^-}}\}$ respectively, and the set of all scores $S = S^+ \cup S^-$. We can also calculate the instance's distance to each cluster center based on the chosen dissimilarity metric in clustering algorithms (e.g. Euclidean distance in K-means, Hamming distance in K-modes), and we denote its distances to the abnormal cluster centers as $D^+ = \{d_{a_1}, d_{a_2}, \ldots, d_{a_{k^+}}\}$ and its distances to the normal cluster centers as $D^- = \{d_{n_1}, d_{n_2}, \ldots, d_{n_{k^-}}\}$, we also denote the set of all distances as $D = D^+ \cup D^-$. Now we show different combination schemes for our proposed "Class Crossing One-vs-All" training scheme as follows:

1. **Without distance information**:
   - **Min rule**: $s = min(S)$, where $S = \{s_{a_1}, s_{a_2}, \ldots, s_{a_{k^+}}, s_{n_1}, s_{n_2}, \ldots, s_{n_{k^-}}\}$
   - **Max rule**: $s = max(S)$
   - **Average rule**: $s = average(S)$

2. **With distance information**:

   - **Hard Combination with / without Global Classifier**: we take the score from the local classifier corresponding to the cluster that the instance has smallest distance to, i.e. $s = s_{j^*}$ where $j^* = \arg\min_j\{d_j \in D\}$ and $D = \{d_{a_1}, d_{a_2}, \ldots, d_{a_{k^+}}, d_{n_1}, d_{n_2}, \ldots, d_{n_{k^-}}\}$. We can also ask for the help of global classifier when the minimum distance is still too large, otherwise same as above, i.e. if $d_{j^*} > \alpha$ then $s = s_g$, otherwise $s = s_{j^*}$, where $j^* = \arg\min_j\{d_j \in D\}$. The α is a parameter to control how much our combined classifier relies on the local or global classifier.
   - **Hard-Soft Combination**: we treat the distances to abnormal clusters and normal clusters differently, first we take the two scores with the minimum distances in the abnormal clusters and normal clusters respectively, and take the weighted sum of the two scores based on the memberships calculated on the two minimum distances, i.e. $s = s_{j^+} \cdot m(1, D_{min}) + s_{j^-} \cdot m(2, D_{min})$, where $j^+ = \arg\min_j\{d_j \in D^+\}$, $j^- = \arg\min_j\{d_j \in D^-\}$, and $D_{min} = \{d_{j^+}, d_{j^-}\}$. The membership calculation is defined in Equation 5.1.
   - **Soft Combination**: we take the weighted sum of all the scores, and the weights are the cluster memberships, i.e. $s = \sum_{i=1}^{k^+ + k^-} s_i \cdot m(i, D)$, where D is the set of the distances between the instance and all the cluster centers.

Now we show the intuitions and relations of the above 3 combination schemes with distance information as follows. As we can see in the above combination schemes, the "Hard Combination without Global Classifier" can be viewed as a special case of "Soft Combination" if we set the fuzzy degree m to be very large as shown in Equation 5.1, since when the fuzzy degree is very large the cluster memberships become completely crisp (the membership corresponding to the minimum distance is close to 1 and other memberships are shrunk to 0), which is exactly the definition of the hard combination.

The "Hard-Soft" combination can be viewed as a compromise between completely hard and soft combinations. We can explain why this combination is reasonable in the following example: assume we have 3 abnormal clusters and 2 normal clusters, and we have trained the 5 local classifiers according to the "Class Crossing One-vs-All" scheme (as shown in Table 5.3). Now when we get a new data instance, we assume it has the smallest distances to the centers of abnormal cluster A_2 and normal cluster N_1 respectively, so we get the scores from the two corresponding classifiers f_2 and f_4 and combine them based on "Hard-Soft" scheme. We argue that this manner is reasonable because:

1. If the ground-truth label of the instance is abnormal, then using the score from the local classifier f_2 *should be beneficial* to our final result since the instance has the smallest distance to that abnormal cluster and the sub-problem should become more linearly separable (as illustrated in Figure 5.1). On the other hand, using the score from the f_4 *should not hurt* our final result since the f_4 is trained against the whole abnormal set (so it has "seen" all the abnormal patterns), and the membership value for the score of f_4 (corresponding to normal cluster N_1) should be smaller since we assume the ground-truth label of the instance is abnormal and it should be quite far away from normal clusters.

2. If the ground-truth label of the instance is normal, then we have the same reasoning but in this case f_4 *should be beneficial* to our final result and f_2 *should not hurt* our final result.

Hence, no matter what the ground-truth label of the instance is, one of our chosen local classifier in the "Hard-Soft" scheme can be beneficial and the other one should not hurt, which can improve our final combined results on average.

As for the "Soft" combination, it is a generalized version of the above "Hard-Soft" scheme, it is also reasonable because some clusters within the same main class can be close to each other, so we can take "advices" from all the local classifiers and give different levels of "trust" based on the distances, but not just take the one that has the smallest distance. The direct comparison between the distances to abnormal cluster centers and the normal cluster centers is also reasonable

since the clustering within each main class is performed in the same feature space using the same clustering algorithm and distance metric.

5.9 Example

To illustrate the whole workflow of our ensemble approach, we select an example with the following chosen components:

- Clustering scheme: Clustering inside each main class separately
- Clustering algorithm: K-means (Euclidean distance, $k^+ = 3, k^- = 2$)
- Classification model: L1-regularized cost sensitive linear SVM (abbreviated as L1-SVM)
- Classifier training scheme: Class Crossing One-vs-All
- Classifier score pre-processing: Platt's scaling (fits a sigmoid function)
- Classifier combination scheme: Soft Combination with fuzzy membership degree $m = 4$

The pseudo code of the above example is shown in the Algorithm 1. As we can see, the final ensemble model can output a classification score for each new data instance, then we can choose a classification threshold (as shown in Section 3.3.5) and use the classifier with the threshold to make binary classification.

Algorithm 1 Pseudo code of an example for our ensemble approach

procedure CLUSTERING(A, N, k^+, k^-) ▷ A: abnormal class, N: normal class
  perform K-means with $k = k^+$ within A, get clusters $A_1, A_2, \ldots, A_{k^+}$
  perform K-means with $k = k^-$ within N, get clusters $N_1, N_2, \ldots, N_{k^-}$
  $k \leftarrow k^+ + k^-$
  $C \leftarrow \{c_1, c_2, \ldots, c_k\} \leftarrow centers(A_1, A_2, \ldots, A_{k^+}, N_1, N_2, \ldots, N_{k^-})$ ▷ get the centers
  return $A_1, A_2, \ldots, A_{k^+}, N_1, N_2, \ldots, N_{k^-}, C$
end procedure

procedure CLASSIFIERTRAINING($A_1, A_2, \ldots, A_{k^+}, N_1, N_2, \ldots, N_{k^-}$)
  for $1 \leq i \leq k^+$ **do**
    Train the L1-SVM classifier f_i^+ on A_i (as positive) and N (as negative)
  end for
  for $1 \leq i \leq k^-$ **do**
    Train the L1-SVM classifier f_i^- on A (as positive) and N_i (as negative)
  end for
  $k \leftarrow k^+ + k^-$
  $F \leftarrow \{f_1, f_2, \ldots, f_k\} \leftarrow \{f_1^+, \ldots, f_{k^+}^+, f_1^-, \ldots, f_{k^-}^-\}$
  return F
end procedure

procedure CALIBRATION($F, CSET, k^+, k^-$) ▷ $CSET$: dataset for fitting the calibration
  $k \leftarrow k^+ + k^-$
  for $1 \leq i \leq k$ **do**
    fits the sigmoid function SG_i based on $f_i \in F$ and $CSET$
  end for
  $SG \leftarrow \{SG_1, SG_2, \ldots, SG_k\}$ ▷ SG: set of sigmoid functions for calibration
  return SG
end procedure

procedure TESTINGNEWDATA(F, SG, C, T, k, m) ▷ T: new data used for testing
▷ m: membership function as defined in Equation 5.1
  for each data instance t_i in T **do**
    for $1 \leq j \leq k$ **do**
      $s_{i,j} \leftarrow f_j(t_i)$ ▷ apply the classifier $f_j \in F$ on the instance t_i
      $\hat{s}_{i,j} \leftarrow SG_j(s_{i,j})$ ▷ apply the calibration sigmoid function SG_j on the score $s_{i,j}$
      $d_{i,j} \leftarrow d(t_i, c_j)$ ▷ calculate the distance between instance t_i and center $c_j \in C$
    end for
    $D_i \leftarrow \{d_{i,1}, d_{i,2}, \ldots, d_{i,k}\}$
    for $1 \leq j \leq k$ **do**
      $m_{i,j} \leftarrow m(j, D_i)$ ▷ calculate the memberships based on the set of distances
      $s_i \leftarrow \sum_{j=1}^{k} m_{i,j} \hat{s}_{i,j}$ ▷ calculate the combined score based on memberships
    end for
  end for
  $S \leftarrow \{s_1, s_2, \ldots, s_n\}$ ▷ the final scores for the testing set T
  return S
end procedure

5.10 Related Work on Clustering-based Ensemble

Similar idea of our clustering-based ensemble approach dated back to 2002, when Japkowicz [18] performed K-means clustering within each main class several times (for the sake of randomness) to have several sets of clusters. Then a decision tree (multi-classifier) was trained for each obtained set of clusters, and the classification results from those decision trees were then combined based on unweighted or weighted votes (probability output of the decision tree), which improved the classification accuracy. This approach falls in the "Direct extension to multi-class" category of classification training schemes, since decision tree is a multi-classifier in nature.

A complicated decision tree usually exhibits low bias and high variance characteristic. Vilalta et al. [32] proposed a framework to improve classification accuracy of low-variance classifiers (such as linear classifiers) by performing clustering inside each main class and merging the obtained clusters by greedy search. Then they trained a multi-classifier over the merged clusters and converted the cluster predictions back to the class predictions. Their results showed improvements in the performance of Naive Bayes, but no significant improvement for the linear SVM model, and their classifier training scheme was not discussed.

Fradkin [12] also used the clustering inside classes approach to improve the classification accuracy for linear classifiers (namely, logistic regression and linear SVM). The results showed significant accuracy improvement for the two linear classifiers by using the clustering approach, but no significant improvement for the non-linear SVM model. A multinomial logistic regression model and a multi-class SVM model were used, although not mentioned in the paper, here the multi-class SVM model being used should be the "One-vs-One" approach in LibSVM [7]. The classifier training schemes and combination schemes were not the focus in this work.

Wu et al. [34] used the clustering inside each main class approach with over-sampling (such that the obtained clusters are relatively balanced) to address the imbalanced problem. Similar as above, they used the multi-class linear SVM model ("One-vs-One" training scheme) and their results showed accuracy improvements for several real-world imbalanced datasets.

Similar as Japkowicz [18] who performed clustering several times to have different partitions of the training data, Rahman et al. [26] proposed an approach to generate different base classifiers based on performing clustering several times (called layered clustering) over the mixed data. Since they performed clustering on the mixed data, some clusters may contain examples of one single class (called atomic clusters) and some may contain mixed class data (called non-atomic clusters). They trained Neural Network models over the non-atomic clusters, and simply assigned the class label if the data instances were passed to the atomic clusters. The distances between the instance and different cluster centers were used to select appropriate clusters. They only considered fusion methods (majority votes) for the base classifiers that can produce discrete-valued class decisions.

Verma et al. [31] proposed a two layer approach Cluster-Oriented Ensemble Classifier (COEC) to improve the classification accuracy by clustering the data, learning different base classifiers to map instances to cluster confidence vectors, and finally training a fusion classifier to map the cluster confidence to the class confidence vector. More specifically, they have considered k-Nearest-Neighbor, Neural Network and SVM for the base classifiers, and the Neural Network for the fusion classifier. They also considered both clustering inside each class and clustering the whole dataset (mixed data), and they found the clustering within each class was significantly better than the latter one in their approach.

Xiao et al. [35] used a similar clustering-based ensemble approach in the domain of credit scoring (similar to our fraud detection domain). They also performed clustering within each class and used the "One-vs-One" scheme to train base classifiers. Then they combined the results of base classifiers by a weighted voting, and the weights are determined by evaluating the base classifiers' performance on the nearest neighbors of an instance.

In summary, most of the above related work use the clustering scheme "clustering inside each main class separately", which is the same as in our clustering-based ensemble approach. Some of them use base classifiers that have direct extension to multi-class such as decision trees and multinomial logistic regression, and some of them (implicitly) use the "One-vs-One" classifier training scheme for linear classifiers like linear SVM. In our ensemble approach, we propose a

classifier training scheme (i.e. Class Crossing One-vs-All) to improve the predictive performance when combined with the "Soft Combination" scheme based on the instance's distances to different cluster centers. Furthermore, each local classifier trained by the Class Crossing One-vs-All scheme can be interpreted as specialized in identifying a certain obtained cluster, which means that each local classifier is possible to have good interpretability if the obtained clusters have meaningful and practical interpretations (e.g. each obtained cluster may correspond to one type of abnormal transactions). For the classifier combination schemes, most of the existing work use majority voting to combine the base classifiers that output discrete-valued classification decisions, whereas we incorporate the distance information and propose several combination schemes for the base classifiers that can output continuous-valued scores. And we can interpret the "Soft Combination" (weighted sum) scheme as giving different levels of trust to each local specialized classifier based on the distance information such that the final ensemble model is possible to have good interpretability (although it still highly depends on the interpretability of the obtained clusters).

Chapter 6

Experiment

In this chapter, we perform different experiments on the real transaction dataset provided by Rabobank. More specifically, we apply and evaluate the previous state-of-art classification models (with specific techniques to address the challenges and requirements) and our novel clustering-based ensemble approach on the real transaction dataset. The goal of the experiment is to evaluate and compare those techniques in terms of predictive performance and interpretability. For predictive performance, as discussed in Section 3.3.3, we use the PAUC_{100} (FPR $\leq$ 100 per 1 million) as final model evaluation metric considering that the FPR is required to be smaller than 100 per 1 million in reality (it's important to note that PAUC_{300} is used for parameter tuning). For interpretability, as discussed in Section 5.1, it's rather subjective to define and compare interpretability between different classification models [3]. One metric that we can quantify is the number of features that are used by the classification model (e.g. when comparing two linear classifiers, the one using less features is usually easier to interpret). Besides the number of used features, we also compare the model structure in terms of interpretability, for example, we analyze the features used by each local classifier in our clustering-based ensemble model.

In the following sections, firstly we describe the experimental settings, e.g. dataset description, dataset splitting and hyper-parameter tuning settings that are applicable to all the following experiments. Secondly, we show the experiment results of the state-of-art classification techniques that we have discussed in Chapter 4, i.e. L1-SVM, L1-LR, stable feature selection, Random Forest and XGBoost. Then we show the experiment results of our proposed Clustering-based ensemble approach. Finally, we give conclusions on all the experiments.

6.1 Experimental Settings

In this section, we show the experimental settings including the dataset description, one-hot encoding setting, dataset splitting for training and testing set, and the hyper-parameter tuning settings that are applicable to all the following experiments.

6.1.1 Dataset description

Firstly we briefly describe the raw dataset that we get from the Confirmed Database, where every transaction has a label that is confirmed by the time delaying rule (as discussed in Section 2.2), then we describe the transformed dataset.

For the raw dataset: each row of the raw transaction dataset represents one transaction, and each transaction contains a label indicating whether the transaction has been confirmed as abnormal or normal, a timestamp indicating the transaction time, and a unique hash that can be used to identify the transaction. Besides the label, timestamp and hash, each transaction also has around 1300 features recording information that is considered as relevant to fraud detection.

These features can be numerical, categorical or textual, which need to be transformed before applying machine learning algorithms.

For the transformed dataset (anonymous and binned dataset), the following operations have been applied on the above raw dataset:

1. **Data labelling**: we denote the label of every abnormal transaction as 1, and the label of every normal transaction as 0. We also denote this label variable as Y (our target variable).

2. **Meta-information extraction**: we denote the two information features timestamp and meta-hash as I, and these two information variables should not be used to predict the target variable Y. Besides Y and I variables, all the other variables are denoted as X, which are used to predict the target variable.

3. **Data type determination**: we determine each X feature to be one of the three data types (i.e. categorical, numerical and textual). Firstly, one week of transaction data (the first week in January 2017) has been aggregated as a sample to generate statistics of each feature (e.g. how many unique values). Then according to the sample statistics, the feature type is determined as follows:

   - If a feature has less than 200 unique values, it is determined to be categorical (denoted by C).
   - Otherwise: if it has more than 80% of its values as numerical, it is determined to be numerical (denoted by N), or else it is determined to be textual (denoted by T).

4. **Data discretization (binning)**: We apply different binning rules to different data types. The binning processes corresponding to the three determined data types are shown in the following:

   - C **(categorical)** - one to one encoding: the unique values are first sorted according to ASCII character order, then the feature values are mapped to the bin values between 0.1 and 0.9 according to that order. Each category corresponds to one bin value, and all the values between two consecutive categories also corresponds to one bin value. For example, if there are 2 unique categories A and B, then $x = A$ corresponds to bin value 0.3, $A < x < B$ corresponds to bin value 0.5, and $x = B$ corresponds to bin value 0.7, so all the categories and their possible values in between are evenly spread in the range of (0.1, 0.9). Since categorical variable has at most 200 unique values, there are at most 400 bin values for each categorical variable.
   - N **(numerical)** - equal frequency based: the unique values are sorted according to numeric order, then the bins with different widths but approximately equal frequencies are generated. There are at most 800 bins (bin values ranging from 0.1 to 0.9, in unit of 0.001). Hence, except some very frequent and identical values, each bin is expected to contain value range that has total frequency around $1/800 = 0.125\%$. The bin id value is taken as the middle point of the bin, for example, if the smallest value 0 has frequency 87.5% (it takes 700 bins), the second smallest value 1 has frequency 0.25% (2 bins), and the third smallest value 2 has frequency 0.125% (1 bin), then the first bin id is $(0.1 + (0.1 + 700 * 0.001))/2 = 0.45$ (original values ranging from $0 \leq x < 1$), and the second bin id is $((0.1 + 700 * 0.001) + (0.1 + 702 * 0.001))/2 = 0.801$ (original values ranging from $1 \leq x < 2$), and so on.
   - T **(textual)** - equal frequency based: the unique values are sorted according to ASCII order, then applies the same frequency based binning process as N variables.
   - **Global binning setting**: besides the above binning process, the values that are smaller than the minimum value of the binning sample are mapped to the bin id 0.05, values that are larger than the maximum value of the binning sample are mapped to the bin id 0.95, empty value is mapped to 0, and the error value is mapped to 0.999. Hence,

Table 6.1: Example of the transformed dataset

I variables (information)		X variables of only C types				Y variable
Timestamp	**Meta-hash**	**C_001**	**C_002**	$\cdots$	**C_856**	**label**
12:00:00 AM 01/10/2016	Sf/dsfJKFds2	0.201	0.3	$\cdots$	0.95	1
12:00:05 AM 01/10/2016	Dfsf*Fdsfnxl	0.05	0.701	$\cdots$	0.342	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
23:59:50 AM 03/03/2017	bdf*Fdf2axd	0.678	0.701	$\cdots$	0.782	0

for all the variables, the binned values float numbers in unit of 0.001, ranging from 0 to 1.

5. **Removal of artifacts and useless variables**: we remove variables that are *exactly* equivalent to the timestamp but in different encoding (e.g. encoded in numbers), since such variables may give perfect but useless predictions when the time periods of the abnormal and normal transactions in the training set are very different.

6. **Anonymization**: we anonymize the original feature names by mapping them to a new set of anonymized feature names. And any information related to the customers have been removed in the binning stage (since all the features are binned to float numbers between 0 and 1).

In this project, we are only concerned with the C features (categorical), so we remove all the features of types N and T. There are 856 categorical features kept in the transformed dataset, each of them is in the range from 0 to 1. Besides, we only consider the transaction data in the time range from 1st October 2016 to 3rd March 2017. An example of the transformed dataset is shown in Table 6.1. In summary, there are around 150 million normal transactions and 1100 abnormal transactions in this transformed dataset. Each row contains two I variables (information features), 856 C type features as X predictors, and the binary target variable $Y = \{0, 1\}$. All the following experiments are concerned with the transformed dataset instead of the raw dataset.

6.1.2 One-hot encoding

As discussed in Section 3.2.4, we use one-hot encoding for linear classifiers since all the features of the transformed dataset are of categorical type. For other classification models such as tree-based models, we use the original representation without one-hot encoding because those tree-based models can handle categorical features in nature. The transformed dataset after converting to one-hot encoding resulted in a matrix with 24988 columns of 0-1 variables. We use the sparse matrix format to store and train models due to the sparsity of the one-hot encoded matrix.

6.1.3 Dataset splitting

As we have shown in the R&D workflow (Figure 3.1), we split the transformed dataset into training set and testing set *by time*. Considering that the total time span of the transformed dataset is from 1st October 2016 to 3rd March 2017 (approx. 5 months of data), we take the transactions that happened in the first 4 months (1st Oct. 2016 to 31st Jan. 2017) as training set, and make the transactions that happened in the last month (1st Feb 2017 to 3rd March 2017) as testing set. The statistics related to the dataset splitting is shown in Table 6.2. The training set is further down-sampled by different ratios and used for model training, and the testing set is used for final evaluation and comparison of different models. This setting of training and testing set is used for all the following experiments. There are better evaluation schemes such as prequential evaluation (test the model on new instances and retrain the model by including the new instances, and so on), but we only consider the above static manner in this thesis due to time limitations.

Table 6.2: Statistics of the training set and testing set after splitting by time

	Training set	**Testing set**
Timespan	4 months (1st Oct. 2016 to 31st Jan. 2017)	1 month (1st Feb 2017 to 3rd March 2017)
Abnormal	769 rows	333 rows
Normal	120 million rows	30 million rows

6.1.4 Hyper-parameter tuning settings

As discussed in Section 3.3.3 and Section 3.3.4, we use the PAUC_{300} (FPR $\leq$ 300 per 1 million) as model evaluation metric for *parameter tuning* (it's important to note that we use PAUC_{100} for final model evaluation and comparison). For the algorithms that have very fast training speed such as the two linear classifiers L1-SVM and L1-Logistic-Regression, we use 5-fold stratified cross validation with 2 repetitions for parameter tuning. For more time-consuming algorithms such as Random Forest and XGBoost, we use 5-fold stratified cross validation (with only 1 repetition) for parameter tuning due to limitations of time and computing resources.

6.2 Experimental Results of State-of-art Classification Techniques

In this section, we show the experiment results of the state-of-art classification techniques that are described in Chapter 4. Firstly, we start from the linear classifiers, i.e. we compare the predictive performance of L1-SVM and L1-LR on the transaction dataset. Then we compare the performance of L1-SVM with and without the stable feature selection technique. Finally, we show the experiment results of the two tree-based (non-linear) classifiers, i.e. Random Forest and XGBoost.

6.2.1 L1-SVM vs. L1-LR

The L1-SVM (L1-regularized cost sensitive linear SVM) and L1-LR (L1-regularized cost sensitive logistic regression) models are the two sparse linear classifiers that we use as base classification models in our ensemble approach. Here we compare the predictive performance of them just as a single classifier. For training the linear classifiers, we further down-sample the normal transactions to 10,000 and combine with all the 769 abnormal transactions in the training set. And we use one-hot encoding for all the categorical features as discussed above.

As discussed in Section 4.2, there are two parameters, i.e. misclassification costs for positive instances C_p and negative instances C_n, that we need to tune according to our tuning evaluation metric (PAUC_{300}). The relative magnitude of those two parameters controls the relative importance of classifying the positive instances and negative instances, the absolute magnitude controls the trade-off between regularization and loss function. In this experiment, for both models we tune the two costs in the Cartesian grid $C \times C$, where C belongs to a set of 43 elements in the range from 0.001 to 500, so there are $43 \times 43 = 1849$ combinations in total. We tune the above grid by 5-fold cross validation with 2 repetitions, and we record the mean values of the following: number of one-hot encoded features (since we are using one-hot encoding) that are used by the model (with non-zero feature weights), number of original features used by the model, PAUC_{300}, and the whole AUC. The detailed CV results of tuning the L1-SVM parameters are shown in Figure A.1 in Appendix.

For both models (L1-SVM and L1-LR), we select the best combination of C_p and C_n that has the highest mean PAUC_{300}, i.e. $C_p = 0.1$ and $C_n = 40$ for L1-SVM model (we denote this model as L1-SVM-0.1-40), and $C_p = 0.3$ and $C_n = 15$ for L1-LR model (we denote this model as L1-LR-0.3-15), then we train the models with the above selected parameters on the down-sampled

Table 6.3: Model details and testing results of L1-SVM and L1-LR

Models	No. of one-hot encoded features	No. of original features	$PAUC_{100}$	$PAUC_{300}$	AUC
L1-SVM-0.1-40	1137	390	3.33×10^{-5}	1.27×10^{-4}	0.975
L1-LR-0.3-15	848	365	2.89×10^{-5}	1.19×10^{-4}	0.983

training set (10,000 normal and 769 abnormal transactions), and test them on the whole testing set. The testing results (PAUC) and model details of the two models are shown in Figure 6.1 and Table 6.3. From the above comparison between L1-SVM and L1-LR, we can conclude that:

1. The PAUC is indeed more suitable than the whole AUC used for model evaluation in our case, since as we can see in Table 6.3, the PAUC of L1-SVM is consistently higher than that of L1-LR when the FPR is required to be less than 100 per 1M, but the whole AUC of L1-LR is higher than L1-SVM. And please note that both models are tuned by the $PAUC_{300}$ metric.

2. L1-SVM has higher predictive performance than L1-LR on this real transaction dataset. In terms of the $PAUC_{100}$, the L1-SVM is almost 20% better than the L1-LR, and both models use similar number of original features.

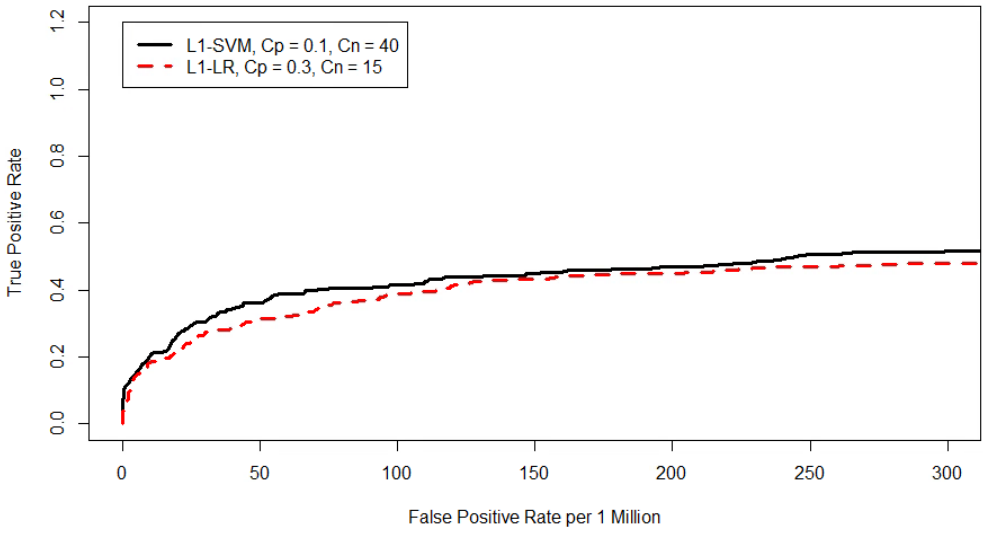


Figure 6.1: Model testing results: L1-SVM vs. L1-LR

6.2.2 L1-SVM with stable feature selection

Here we compare the predictive performance of L1-SVM with and without stable feature selection. As discussed in Section 4.2.5, the stable feature selection is done by first creating multiple random samples, training models on the samples, then averaging the feature vectors and filtering the features. In this experiment, we create 100 training samples by taking random samples of 10,000 normal transactions and combining with all the 769 abnormal transactions in the training set, so each training sample has 10,769 transactions (same as the above experiment setting). We tune the two parameters C_p and C_n for each training sample as above (the only difference is that we tune the parameters in a smaller grid due to time limitations), and we train 3 models for each sample using the top 3 sets of parameters according to the $PAUC_{300}$ metric. Then we have 300 feature vectors in total, and we take the average of these 300 feature vectors and take the absolute values

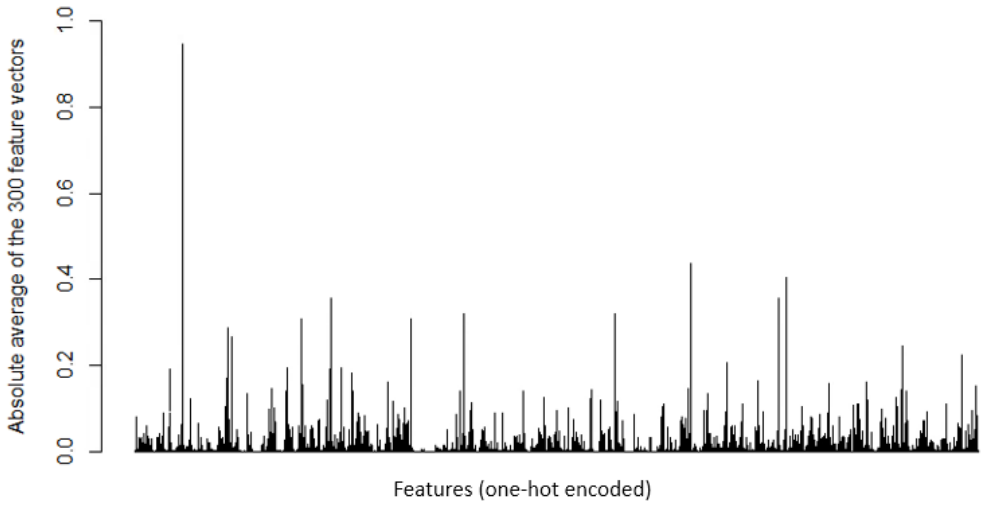


Figure 6.2: Absolute values of the average of the 300 feature vectors

Table 6.4: Model details and testing results of L1-SVM with and without feature selection

Models	No. of one-hot encoded features	No. of original features	$PAUC_{100}$	$PAUC_{300}$	AUC
L1-SVM-FS-0.06-5	429	210	3.75×10^{-5}	1.35×10^{-4}	0.983
L1-SVM-0.1-40	1137	390	3.33×10^{-5}	1.27×10^{-4}	0.975
L1-SVM-0.15-1.5	643	254	3.44×10^{-5}	1.31×10^{-4}	0.981

of the averaged vector. The bar chart of the absolute average vector is shown in Figure 6.2. As we can see in the figure, there are a few outstanding features that are consistently important in all the models trained on the 100 random samples, and many of others are less important (quite many of them are even consistently 0). As an example, there are around 1300 one-hot encoded features that have absolute average value more than 0.01 (which is quite small already). Hence, we only keep the top 1500 one-hot encoded features, and the top 300 original features with respect to the absolute sum values (as discussed in Section 4.2.5), which results in a final selected set of features consisting of 1494 one-hot encoded features.

Using only this selected set of stable features, we tune and train the L1-SVM model on the same training set (the one that was used to compare L1-SVM and L1-LR), we get the best parameter set $C_p = 0.06$ and $C_n = 5$ and we denote this model as L1-SVM-FS-0.06-5 (FS means feature selection). This model with feature selection uses 429 one-hot encoded features and 210 original features. For a more reasonable comparison, we also select the parameter set $C_p = 0.15$ and $C_n = 1.5$ for the L1-SVM model *without* feature selection (denoted as L1-SVM-0.15-1.5), which uses similar number of features compared with the number of features used by the L1-SVM-FS-0.06-5. Finally, we test the above models on the testing set. The PAUC graph of the testing results are shown in Figure 6.3, and the model details and testing results are shown in Table 6.4.

From the testing results, we can conclude that:

1. Without the stable feature selection, the L1-SVM model that uses less features (more regularization) is possibly better than the one that uses much more features, since the more regularized model is less likely to overfit the data.

2. Stable feature selection can improve the predictive performance of linear classifiers (L1-SVM in this case). We can see that the PAUC metric of the L1-SVM model with feature selection is better than the two models without feature selection (including a model that uses similar number of features).

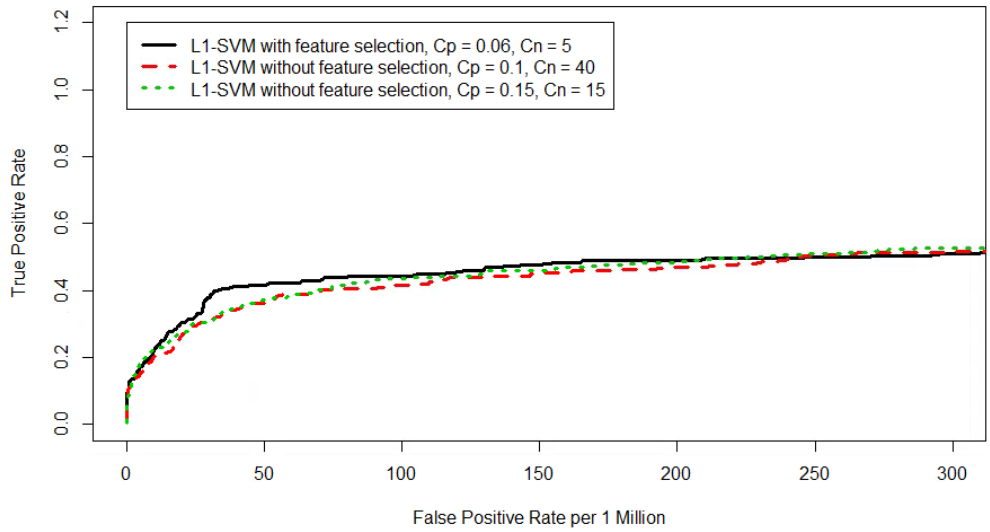


Figure 6.3: Model testing results: L1-SVM with feature selection vs. without feature selection

Table 6.5: Testing results of Random Forest and XGBoost

Model	$PAUC_{100}$	$PAUC_{300}$	AUC
Random Forest	5.22×10^{-5}	1.76×10^{-4}	0.981
XGBoost	4.7×10^{-5}	1.66×10^{-4}	0.986

6.2.3 Random Forest vs. XGBoost

For training the Random Forest model, we down-sample the normal data in the training set from 120 million rows to around 2.7 million rows and combine them with all the abnormal data (769 rows) in the training set. In this experiment, we use the H2O Distributed Random Forest implementation [14] to train the Random Forest model because of its better scalability and higher efficiency. There are two main hyper-parameters that we need to tune, i.e. *mtry* (number of features to consider on each node) and the *sample_rate_per_class* (down-sample ratio per class when growing each tree). Due to time limitations, we only tune the best *mtry* using all the other default parameters in a grid from 5 to 40 (step size 5), and we chose the $mtry = 25$ according to the evaluation metric $PAUC_{300}$. Using this best *mtry*, we then fix the down-sampling ratio for abnormal class when growing each tree to be 0.7, and tune the best down-sampling ratio (0.6) for normal class. Using $mtry = 25$ and $sample\_rate\_per\_class =$ (normal 0.6, abnormal 0.7), we keep other parameters as default and grow 500 trees. Then we test this Random Forest model on the testing set, and the performance (PAUC graph) is shown in Figure 6.4.

For training the XGBoost, we further down-sample the normal data in the training set from 120 million rows to around 1 million rows and combine them with all the abnormal data (769 rows) in the training set. There are many parameters in the XGBoost algorithm, including the cost sensitive parameter *scale_pos_weight* that controls the relative weights between positive and negative examples. However, tuning this model is not our main focus in this thesis due to time limitations, thus we use all default parameters with $nrounds = 1000$ and $early\_stopping\_rounds = 10$ to avoid overfitting. We end up with a model consisting of 123 trees, then we test this XGBoost model on the testing set, and the performance (PAUC graph) compared with the above Random Forest model and the L1-SVM model with feature selection (L1-SVM-0.1-40) is shown in Figure 6.4. The PAUC and AUC values of the above RF and XGBoost model (on the testing set) are shown in Table 6.5.

From this experiment, we can conclude that:

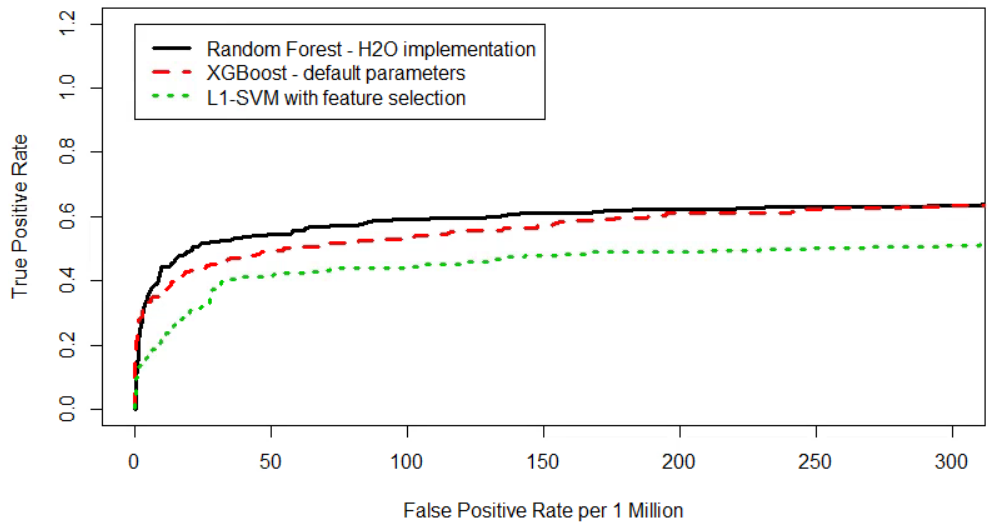


Figure 6.4: Random Forest vs. XGBoost

1. Both tree-based ensemble models (i.e. Random Forest and XGBoost) achieve great predictive performance on this real transaction dataset. However, as discussed in Section 5.1, these tree-based ensemble models are relatively hard to interpret compared with a single decision tree or linear classifier.

2. Both tree-based ensemble models (non-linear classifiers) outperform the L1-SVM model (linear classifier) significantly. The obtained Random Forest model is 60% better than the L1-SVM-0.1-40 in terms of the $PAUC_{100}$ metric.

3. In Table 6.5 we can observe that the RF model has higher $PAUC_{100}$ and $PAUC_{300}$ than the XGBoost, but it has lower AUC than the XGBoost, which again confirm that the PAUC is a suitable evaluation metric considering the low FPR requirement in reality.

We take the above Random Forest model as a representative of tree-based ensemble models to compare with our proposed clustering-based ensemble approach in the following experiments.

6.3 Experimental Results of Clustering-based Ensemble Approach

In this section, we show the experimental results of our proposed approach, i.e. clustering-based ensemble of local sparse linear classifiers. As shown in Chapter 5, our approach can be decomposed into 6 components, i.e. clustering schemes, clustering algorithms, classification models, classifier training schemes, classifier score pre-processing, and classifier combination schemes. Since there are 6 different components and many different combinations in our ensemble approach, to make it more clear for the comparisons of following experiments, we create and show the notations of different choices for those components in Table 6.6. In this table, the clustering algorithms are not shown since we choose and fix our clustering algorithm to be K-means as shown in the next subsection.

In the following sections, firstly we show the experimental results about clustering on the transaction dataset, which are then used in the following experiments. Secondly, we fix the classifier training scheme to be CCOvA (Class Crossing One-vs-All) and the base classifiers to be L1-SVM, and we compare the predictive performance of different component choices in our ensemble approach (i.e. different clustering choices, different score pre-processing methods and different classifier combination schemes). Thirdly, we compare the performance of the CCOvA classifier

Table 6.6: Notations for different choices and components in the clustering-based ensemble approach

Components	Choices	Notations
Classifier training schemes	One-vs-All	OvA
	One-vs-One	OvO
	Direct Multi-Classifiers	DMC
	Class Crossing One-vs-All	CCOvA
Clustering choices	Only clustering within abnormal class i.e. $K_A = 3, K_N = 1$	K_{31}
	Clustering within abnormal and normal class separately, i.e. $K_A = 3, K_N = 2$	K_{32}
Classification Models	L1-SVM	L1-SVM
	L1-LR	L1-LR
Score Pre-processing	No pre-processing	#
	Standardization	SD
	Platt's Scaling	PS
Classifier combination	Min rule	MIN
	Max rule	MAX
	Average rule (mean rule)	AVG
	Hard Combination with or without Global	HC(G)
	Hard-Soft combination with fuzzy membership degree k	HS_k
	Soft Combination with fuzzy membership degree k	SC_k

training scheme with two common training schemes (i.e. OvA and OvO). Fourthly, we validate the idea of this clustering-based ensemble approach by examining the effects of clustering and classification separately, and we also show the main limitations of such approach. Finally, we compare the predictive performance of our proposed clustering-based ensemble approach with some state-of-art classification techniques (e.g. Random Forest and the simple Bagging of linear classifiers), and we also analyze the interpretability of the obtained clustering-based ensemble model.

6.3.1 Clustering

In this subsection, we first select a suitable clustering algorithm that can work well on the real transaction dataset, then we apply this chosen algorithm to perform clustering within the abnormal class and normal class separately. To select a suitable algorithm, we mix the abnormal and normal data (keeping their class labels), apply a clustering algorithm on this mixed dataset and evaluate the clustering results based on the external index ARI (Adjusted Rand Index) as discussed in Section 5.4. A reasonably good clustering algorithm should be able to roughly separate the two main classes of the mixed dataset, and we also have the class labels to use external index ARI, although the main class labels are coarse.

First we randomly select 2300 normal transactions and combine them with all the 769 abnormal transactions in the training set, then we perform K-means clustering with Euclidean distance, and K-medoids (PAM - Partitioning Around Medoids in this case) with Manhattan distance on this mixed dataset. We also choose different K (number of clusters) and calculate the adjusted Rand index (external index using the main class labels) for different choices of K. The different ARI values for K-means and K-medoids with different choices of K are shown in Figure 6.5. As we can see in the figure, the ARI values of K-means (with Euclidean distance) are consistently higher than that of K-medoids (with Manhattan distance). For both algorithms, the ARI index suggests

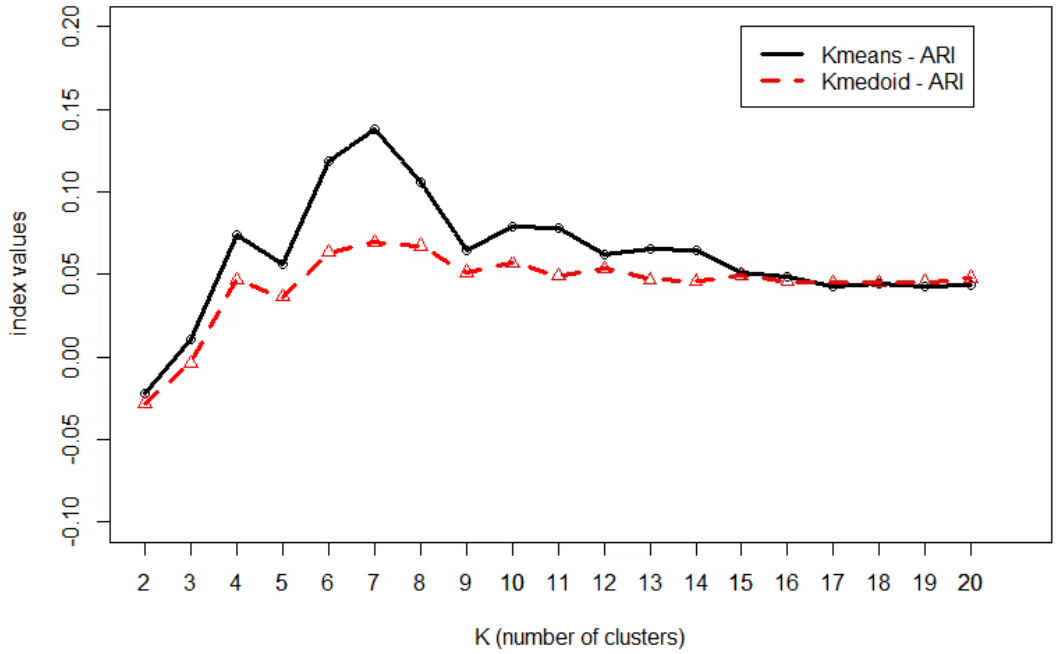


Figure 6.5: ARI values for K-means and K-medoids with different choices of K

Table 6.7: K-means (K=7) clustering results on the mixed data

		Clusters						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Classes	**0**	237	19	744	641	14	36	616
	1	62	153	25	84	221	137	87

K=7, so we can run both algorithms with K=7 on the mixed dataset, and we show the clustering results of K-means (K=7) and K-medoids (K=7) in Table 6.7 and Table 6.8. As we can see in the two tables, the K-means with K=7 is more capable of separating the two main classes apart, and the K-means with K=7 suggests there are 3 clusters (2, 5 and 6) for the abnormal class. On the other hand, the K-medoids can only get two clusters for the abnormal class (6 and 7), and the other abnormal data are mixed with normal clusters. Hence, we conclude that K-means with Euclidean distance is more suitable than K-medoids with Manhattan distance on this transaction dataset.

Now we perform K-means clustering (with Euclidean distance) inside each main class separately. The training set (same as the one used for training L1-SVM models) has 769 abnormal transactions and 10000 normal transactions. First, we perform K-means clustering within the abnormal class, then within the normal class. As discussed in Section 5.4, we don't have sub-type labels for clustering inside each main class, so we can only rely on internal index such as the average SI (silhouette index) and some other metrics such as the instance distribution and time distribution of the clusters. Now for the 769 abnormal transactions in the training set, we run the

Table 6.8: K-medoids (K=7) clustering results on the mixed data

		Clusters						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Classes	**0**	547	636	534	332	239	8	11
	1	20	151	102	136	61	133	166

Table 6.9: Instance distribution and time distribution of K-means inside abnormal class when K=2 and K=3

		No. of instances	Min_time	Max_time	Mean_time	SD_time
K = 2	**Cluster 1**	185	20161010	20170131	20170104	27 days
	Cluster 2	584	20161003	20170131	20161207	33 days
K = 3	**Cluster 1**	251	20161003	20170131	20161204	36 days
	Cluster 2	334	20161005	20170127	20161212	29 days
	Cluster 3	184	20161010	20170131	20170104	27 days

K-means with different K (number of clusters), and we show the SI values for different K in Figure 6.6. The SI values suggest to choose K=2, however, the obtained two clusters have quite imbal-

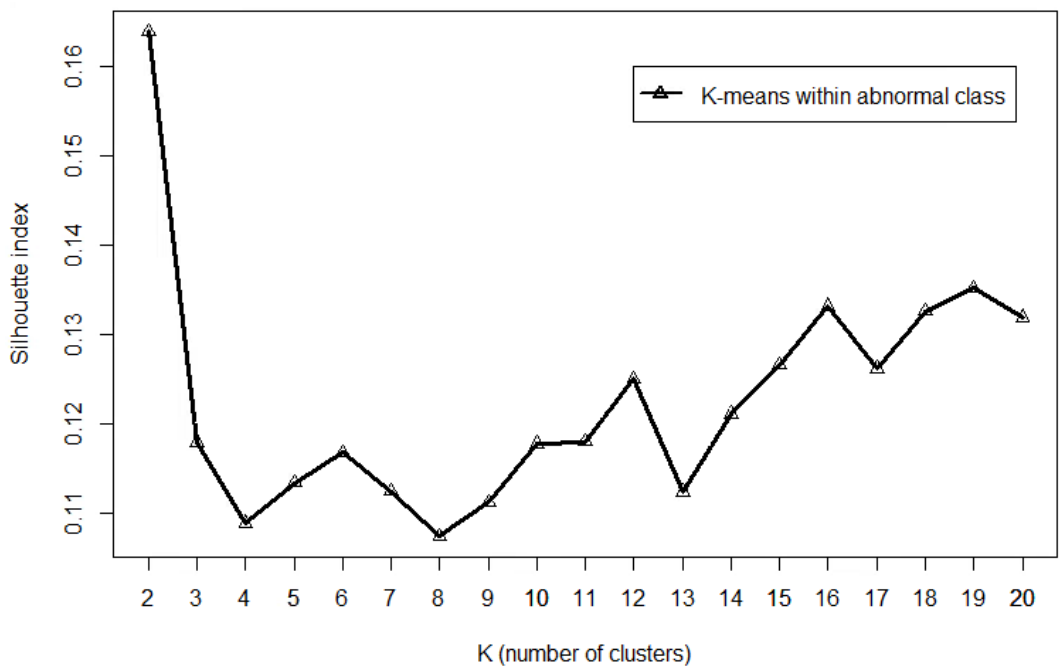


Figure 6.6: Average Silhouette index values for performing K-means with different choices of K inside the abnormal class

anced instance distribution (cluster 1 has 185 instances and cluster 2 has 584 instances). Hence we choose K=3, which results in three clusters that have more balanced numbers of instances and more spread time distributions. The instance distribution and time distribution of the obtained clusters when K=2 and K=3 are shown in Table 6.9. From the table, we can observe that K=3 is a better choice since the resulted clusters have more balanced number of instances, which is beneficial to our ensemble approach at later stage.

So far we have performed K-means clustering (K=3) with Euclidean distance within the abnormal class, one may argue that the Euclidean distance is not that suitable for our transaction dataset, whose features are considered as categorical. Hence, we also experiment with K-modes (as shown in Section 5.4), which is designed for clustering categorical data. We apply K-modes (with K=3) on the abnormal data, and we compare the clustering results of K-means and K-modes in Table 6.10. As we can see in the table, the clustering results of K-means and K-modes only differ by less than 10%. Hence, we only use K-means with Euclidean distance as our clustering algorithm in the following experiments, and we have already chosen K = 3 for the abnormal class. Some of the following experiments of our ensemble approach only require to cluster the abnormal

Table 6.10: Clustering result comparison between K-means and K-modes, when K = 3

		K-means		
		Cluster 1	**Cluster 2**	**Cluster 3**
K-modes	**Cluster 1**	245	7	2
	Cluster 2	0	303	13
	Cluster 3	6	24	169

class and keep the normal class as a whole, but some of them also require to cluster the normal class. For clustering the normal data (10,000 normal transactions in the training set), we choose K-means with K = 2 considering similar reasons as above. The clustering results of K=2 for the normal class are shown in Table A.1 in Appendix.

Besides, as discussed in Chapter 5, the interpretability of the final ensemble model highly depends on the interpretability of the obtained clusters. Hence, we also check the interpretability of the clustering results (3 abnormal clusters) with the Rabobank domain experts, and one possible interpretation of those obtained 3 clusters is that each of them corresponds to one type of transaction sources with their typical fraud patterns.

In summary, we choose K-means clustering with Euclidean distance as our clustering algorithm, and we have perform K-means with K=3 for abnormal class (3 abnormal clusters) and K=2 for normal class (2 normal clusters). Some of the following experiments only require the 3 abnormal clusters and keep the normal class as a whole (denoted as K_{31}), some of them require both of the 3 abnormal clusters and the 2 normal clusters (denoted as K_{32}). The information (such as instance memberships, cluster centers) of the above 3 abnormal clusters and/or 2 normal clusters are used for the following classifier training in our ensemble approach.

6.3.2 Comparisons within the CCOvA scheme

In this subsection, we fix the classifier training scheme to be CCOvA (Class Crossing One-vs-All) and the base classifiers to be L1-SVM model, and we compare the predictive performance of different clustering choices (i.e. K_{31} or K_{32}), different score pre-processing methods (SD or PS), and different classifier combination schemes (HC, HS_k and SC_k). The meanings of the above notations are shown in Table 6.6. Firstly we show the details of how we train the local classifiers for K_{31} and K_{32}, then we show the experimental results of the above different combinations.

Since we choose the classifier training scheme to be CCOvA, we need to train 4 classifiers (3 local, 1 global) for K_{31} and 5 classifiers (5 local) for K_{32}. It is important to note that we do not use the stable feature selection technique in this ensemble approach to train local classifiers, except that we reuse the single global L1-SVM classifier with stable feature selection for convenience. For K_{31}, we train 3 L1-SVM classifiers for the three abnormal clusters (K_{31}) by making each abnormal cluster as positive and the whole normal class as negative, and 1 global L1-SVM classifier for the whole abnormal class and normal class (we reuse the one trained in Section 6.2.2). For training those 3 local classifiers separately, we also tune the parameters by 5-fold cross validation with 2 repetitions as in the previous experiments, the only difference is that we explicitly restrict the number of original features used by each local classifier to be smaller than 200 (the average number of used features is recorded when doing cross validations), which can make each local classifier more sparse and regularized. To train the 5 local classifiers (K_{32}) for the three abnormal clusters and two normal clusters, we can reuse the previous 3 local classifiers and just need to add 2 more additional local classifiers for the two normal clusters, since our CCOvA scheme is additive (as shown in Section 5.6). For training the two additional classifiers, we use the same training process as above by making each normal cluster as negative and the whole abnormal class as positive. The model details of these 5 local classifiers and the single global classifier are shown in Table 6.11.

According to the classifier training scheme of CCOvA, the ensemble classifier for K_{31} is composed of 3 local classifiers (corresponding to the 3 abnormal clusters) and the global classifier, and the ensemble classifier for K_{32} is composed of the previous 3 local classifiers for the 3 abnormal

Table 6.11: Model details of the 5 local classifiers and the single global classifier

	Training scheme (positive vs. negative)	**No. of one-hot encode features**	**No. of original features**
Global	all abnormal vs. all normal	429	210
Local 1	abnormal 1 vs. all normal	297	164
Local 2	abnormal 2 vs. all normal	308	189
Local 3	abnormal 3 vs. all normal	118	102
Local 4	all abnormal vs. normal 1	436	246
Local 5	all abnormal vs. normal 2	165	131
K_{31}	Local 1, 2, 3 & Global	743	310
K_{32}	Local 1, 2, 3, 4, 5	751	331

Table 6.12: Top 10 combinations with highest predictive performance

Combinations	$PAUC_{100}$ ($\times 10^{-5}$)	$PAUC_{300}$ ($\times 10^{-5}$)	AUC
K31_PS_SC_3	5.16	17.29	0.983
K32_PS_SC_3	5.15	17.29	0.983
K31_PS_SC_2	5.15	17.27	0.982
K31_PS_HS_3	5.14	17.32	0.983
K32_PS_SC_2	5.14	17.29	0.982
K31_PS_HS_2	5.13	17.29	0.983
K32_PS_HS_2	5.1	17.28	0.98
K32_PS_HS_3	5.1	17.27	0.981
K32_PS_AVG	5.09	17.26	0.981
K31_PS_MAX	5.08	17.15	0.979

clusters and the 2 local classifiers for the 2 normal clusters. The number of features used by the K_{31} and K_{32} is the number of features in the union of the contained classifiers, e.g. the features used by K_{31} are the union of features used by Local 1, 2, 3 and the global classifier.

We fix the classifier training scheme to be CCOvA and the base classifiers to be L1-SVM models, and we test different combinations of the other ensemble components, i.e. 2 possibilities for clustering choices (K_{31} and K_{32}), 3 possibilities for score pre-processing (#, SD, PS), and 8 possibilities for classifier combination methods (MIN, MAX, AVG, HC, HS_2, HS_3, SC_2 and SC_3), which result in $2 \times 3 \times 8 = 48$ combinations. For the score pre-processing, we aggregate all the transactions (around 280 abnormal and 28 million normal transactions) in the last 3 weeks of the training set, we apply the classifiers on this aggregated sample and calculate the mean value and standard deviation for the SD method, and we also fit the Platt's scaling (in essence, a logistic regression of the labels against the scores) on this sample. Then the calculated mean, standard deviation and logistic regression parameters are used for pre-processing the classifier scores when testing. For the classifier combination methods, we experiment with two fuzzy membership degrees (2 and 3), so there are two possibilities for each of the HS and SC. The testing results of all these 48 possibilities are shown in Table A.2 in Appendix. The top 10 combinations with the highest performance (in terms of $PAUC_{100}$) are shown in Table 6.12. As we can see in the table, the top 10 combinations are all using PS (Platt's scaling) as score pre-processing method, and the top 8 combinations are all based on distance information (compared with the last 2 combinations that use simple combination rules). Furthermore, the best combination is PS (Platt's scaling) and SC (soft combination) with fuzzy membership degree 3. In terms of the best combination, the performance results of K_{31} and K_{32} are similar to each other.

To compare different possibilities within each component, we fix each choice in each component and calculate the average performance ($PAUC_{100}$) by varying choices in other components, and

Table 6.13: Average performance for different choices in different components

Components	Choices	Average $PAUC_{100}$ ($\times 10^{-5}$)
Clustering choices	K31	4.38
	K32	3.93
Score pre-processing	#	3.89
	SD	3.91
	PS	4.67
Classifier Combinations	MIN	1.89
	MAX	4.02
	AVG	4.44
	HC	4.52
	HS	4.5 (with PS: 5.11)
	SC	4.68 (with PS: 5.15)
Fuzzy degree	2	4.55
	3	4.62

the average performances for different possibilities are shown in Table 6.13. From the table, we can see that the predictive performance ($PAUC_{100}$) of K_{31} is higher than K_{32} on average, which may due to the fact that the classifier training scheme of K_{31} includes the global classifier, and the K_{32} does not (it includes all five local classifiers). Furthermore, the clustering results of the 2 normal clusters may hinder the predictive performance since the cluster instances of the 2 normal clusters are a bit unbalanced (one normal cluster has around 6500 instances, and another has around 3500 instances). However, the best combinations of K_{31} and K_{32}, i.e. K31_PS_SC_3 and K32_PS_SC_3, have very similar performance. Furthermore, from the average performance table, we can also conclude that the simple standardization method (SD) is just a bit better than no score pre-processing (#), and the score pre-processing by PS (Platt's scaling) can significantly improve the ensemble performance. And we can also see that the classifier combination methods that are based on distance information have higher performance than the simple combination rules, and the SC (soft combination) method has the highest performance. Different fuzzy degrees can also affect the ensemble performance, higher fuzzy degrees can result in a crisper and less fuzzier combination. On this dataset, the ensemble with higher fuzzy membership degree 3 has higher performance.

The ensemble of K_{31} consists of 3 local classifiers and the global classifier (as shown in Table 6.11), so we also compare the performance of the best combination of K_{31} (K31_PS_SC_3) and each individual classifier that is contained in the ensemble, the PAUC graph of the comparison is shown in Figure 6.7. As we can see in the figure, the predictive performance of the ensemble model K31_PS_SC_3 is significantly higher than all the individual classifiers (3 locals and 1 global). We can also see that the performance of the local classifier 3 performs better than the global classifier on the testing set, which is quite surprising and needs further investigations as follows. As shown in the previous clustering experiment section, we perform K-means (K=3) within the 769 abnormal transactions in the *training set* to get 3 abnormal clusters, which results in 251, 334 and 184 instances in the 3 clusters respectively (32%, 44%, 24%). On the other hand, we can define that an abnormal transaction in the *testing set* belongs to one of the above 3 clusters if it has the smallest distance to that cluster center, then there are 67, 107 and 159 abnormal transactions in the testing set that belong to the above 3 clusters respectively (20%, 32% and 48%). The proportion of instances in the cluster 3 is much higher in the testing set than in the training set (24% in the training set and 48% in the testing set), which implies that the local classifier 3 benefits a lot from this proportion change and performs much better on the testing set, and the global classifier becomes worse since it was trained to concentrate on the majority clusters (cluster

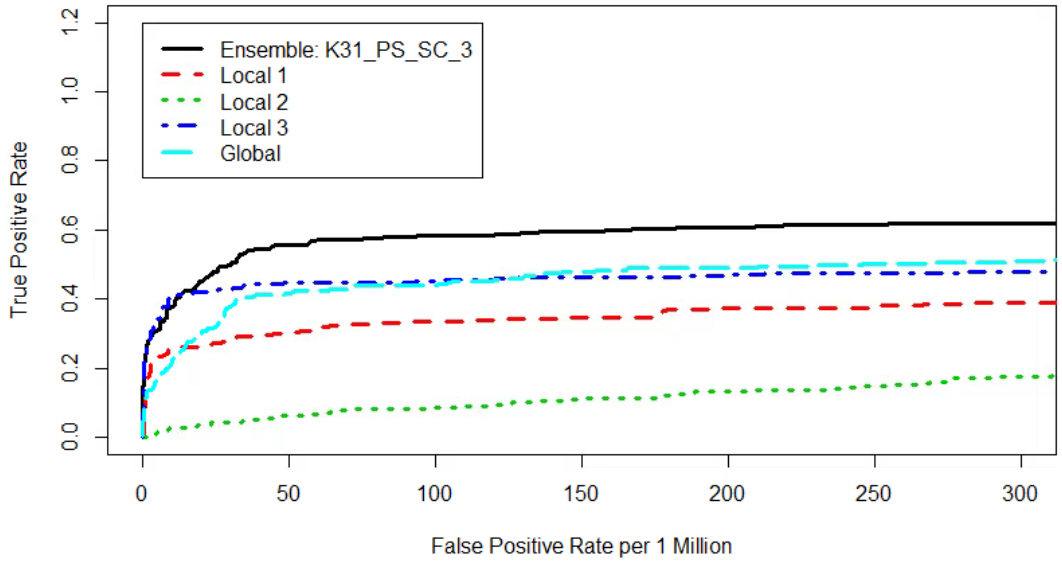


Figure 6.7: Performance comparison between the ensemble model K31_PS_SC_3 and each individual classifier

1 and 2) in the training set. This proportion change also reflects the fact that such distribution of abnormal sub-types can change over time and usually it is unknown in reality. Hence, our clustering-based ensemble approach can overcome this difficulty by discovering clusters, training local classifiers that are specialized in identifying certain obtained clusters, and combining the local classifiers flexibly based on the distance information.

In summary, we fix the classifier training scheme to be CCOvA and the base classifiers to be L1-SVM models, and we have compared the predictive performance of different choices in different components (e.g. different score pre-processing methods). Our experiment results show that:

1. For clustering choices, there is no significant difference between the K_{31} and K_{32}, so we only use K_{31} (only clustering the abnormal class into 3 clusters and keep the normal class as a whole) in the following experiments for simplicity.

2. For the score pre-processing methods, there is no significant difference between no score-processing (just keep the original scores) and simple standardization. On the other hand, the score pre-processing method PS (Platt's scaling) can improve the performance of the ensemble significantly, since the PS can not only convert the scores into same scales, but also into same meanings as probability estimates. Furthermore, Platt's scaling is also the most suitable for SVM models, which are known to have sigmoid shapes in terms of probability calibration.

3. For the classifier combination methods, the simple average rule can do fairly well, but the combination methods that are based on distances, especially the SC (soft combination), can further improve the ensemble performance.

4. For the fuzzy membership degrees, the fuzzy degree 3 has better performance on this transaction dataset (it prefers crisper and less fuzzier combinations).

In the following experiment sections, we take the best combination K31_PS_SC_3 as the representative of our ensemble approach for the CCOvA scheme, and we compare it with other classifier training schemes such as OvA and also the state-of-art classification techniques (such as Random Forest) in the following experiments.

6.3.3 Comparisons with other classifier training schemes

In the previous section, we focus on the comparisons of different component choices *within* the CCOvA classifier training scheme. In this section, we compare the performance of CCOvA scheme (K31_PS_SC_3 as the representative) with two common classifier training schemes OvO (One-vs-One) and OvA (One-vs-All) . We use L1-LR as base classification models in this section since the outputs of each local logistic regression model are probabilities and do not require pre-processing (e.g. probability calibration for L1-SVM in the above section). The training set and testing set are exactly the same (except the fact that some additional training data were used for pre-processing each local L1-SVM model in the above section), so that the performance comparisons are reasonable.

For the OvO classifier training scheme, we ignore the individual classifiers whose positive and negative examples are of the same main class since our final task is just binary classification of the two main classes. Furthermore, we only consider the clustering choice K_{32} since the OvO for K_{31} is almost the same as our proposed CCOvA (the only difference is that the CCOvA scheme has 1 more global classifier). We train 6 local L1-LR classifiers for K_{32} by making each abnormal cluster as positive and each normal cluster as negative. For tuning the parameters of each local L1-LR classifier, we use two different evaluation metrics $PAUC_{300}$ and AUC. For the classifier combination rules, we use three different simple rules, i.e. AVG, MIN and MAX, to get the final scores. Hence, we have $2 \times 3 = 6$ different combinations for OvO with K_{32}. The testing results of those different combinations in the OvO training scheme are shown in Table 6.14.

For the OvA classifier training scheme, we consider the two clustering choices (i.e. K_{31} and K_{32} that are obtained in the clustering section), so we train 4 local L1-LR classifiers for K_{31} and 5 local L1-LR classifiers for K_{32} respectively (each cluster acts as positive examples once and all the other clusters act as negative examples). To tune the parameters (C_p and C_n) for each local L1-LR classifier, the evaluation metrics $PAUC_{300}$ and AUC are not suitable anymore since the training examples of the same class (abnormal or normal) can be both positive and negative, which means that a FP (in the sense of false cluster prediction) of a local classifier in this OvA training scheme can be actually TP in the sense of the original main class prediction. In fact, the experiment results also show very bad predictive performance when using the PAUC or AUC for tuning the local classifiers separately in OvA scheme. Hence, we take the binary classification task into consideration when tuning each local classifier, i.e. when testing on the cross validation fold, we first scale the probabilities from each local classifier such that they sum up to 1 after divided by their sum ($\hat{p_i} = \frac{p_i}{\sum p_i}$), then we take the sum of the probabilities corresponding to the abnormal local classifiers (the ones that use abnormal clusters as positive examples) as the final scores, and we evaluate the final scores by the main class labels and get the $PAUC_{300}$ metric. For simplicity, we restrict the cost sensitive weights of all local classifiers corresponding to the abnormal clusters to be the same ($C_p = C_{A_1} = C_{A_2} = C_{A_3}$ and also the weights of all the normal clusters to be the same ($C_n = C_{N_1} = C_{N_2}$), then we tune the parameters C_p and C_n according to the above way. For consistency, the classifier combination rule is the same as above (scale the probabilities from the local classifiers such that they sum up to 1, then take the sum of the probabilities from the local classifiers that use each of the 3 abnormal clusters as positive examples). We denote this rule as "scale_sum" rule, and the testing results of the OvA scheme are shown in Table 6.14. The model details (such as number of used features) of the above OvO and OvA models are shown in Table 6.15.

Comparing the testing results of the models trained by OvO and OvA schemes with the models trained by our proposed CCOvA scheme (as shown in Section 6.3.2), we can conclude that:

1. The models trained by the CCOvA have much higher predictive performance than the models trained by the OvO and OvA schemes (when only using simple combination rules such as AVG). The main disadvantage of OvA scheme is that it differs from our final task, i.e. binary classification, since the positive and negative examples of this scheme contain instances of the same main class (e.g. A_2 and A_3 are negative when A_1 is positive). As for the OvO scheme, each local classifier is only concerned with part of the abnormal examples and part of the

Table 6.14: Testing results of the OvO scheme and OvA scheme

Classifier Training Scheme	Clustering Choice	Tuning Evaluation Metric	Classifier Combination Rules	$PAUC_{100}$ ($\times 10^{-5}$)	$PAUC_{300}$ ($\times 10^{-5}$)	PAUC
OvO	K_{32}	$PAUC_{300}$	MIN	2.15	9.55	0.881
OvO	K_{32}	$PAUC_{300}$	MAX	1.73	6.51	0.882
OvO	K_{32}	$PAUC_{300}$	AVG	2.18	8.78	0.946
OvO	K_{32}	AUC	MIN	0.407	2.45	0.912
OvO	K_{32}	AUC	MAX	1.01	4.65	0.911
OvO	K_{32}	AUC	AVG	1.20	5.93	0.963
OvA	K_{31}	Scale_sum $PAUC_{300}$	Scale_sum	2.60	10.0	0.981
OvA	K_{32}	Scale_sum $PAUC_{300}$	Scale_sum	1.29	7.06	0.979

Table 6.15: Model details of the OvO scheme and the OvA scheme

Models	No. of used one-hot encoded features	No. of used original features
OvO_K_{32}_$PAUC_{300}$	2505	464
OvO_K_{32}_AUC	1003	413
OvA_K_{31}_Scale_sum	1816	677
OvA_K_{32}_Scale_sum	3922	478

normal examples, the scores from the local classifiers are not guaranteed to be in the same scale although they have the same meanings as probabilities when using logistic regression as base models, thus the simple combination rules such as AVG may not work well. The common classifier combination rule Majority Voting can solve this problem by making each local classifier only output a vote, but we do not consider Majority Voting in our case since we use PAUC as performance evaluation metric and the majority voting cannot produce continuous output when the number of classifiers is small.

2. The models trained by the CCOvA are much easier to interpret than the models trained by the OvO and OvA schemes. As we can see in Table 6.11 and Table 6.15, the models trained by the CCOvA scheme use less number of original features (300+ compared with 400+) and one-hot features (700+ compared with at least 1000). Furthermore, the OvO scheme needs to train relatively large number of local classifiers ($K_A \times K_N$), and each local classifier of the OvO scheme only involve part of the abnormal transactions and part of the normal transactions, which is harder to interpret compared with the CCOvA scheme. As for the OvA, it is even harder to interpret compared with the CCOvA scheme since the positive and negative examples of each local classifier can be in the same main class (i.e. each local classifier is not just identifying the abnormal from the normal transactions, but also the subtypes of the abnormal transactions). In contrast, the CCOvA scheme can guarantee that each local classifier can be interpreted as specialized in identifying certain obtained clusters from the other whole main class, although the interpretability of the final ensemble model still highly depends on the interpretability of the obtained clusters.

In summary, we have shown that the models trained by the CCOvA scheme (using distance-based combination rules) have higher predictive performance and better interpretability than the models trained by the common OvO and OvA schemes (using simple combination rules).

6.3.4 Validation of the clustering-based ensemble idea

In this section, we validate the idea of this clustering-based ensemble approach by examining the effects of clustering and classification respectively. More specifically, since this ensemble approach is a combination of both clustering and classification (training local classifiers based on the clustering results), one may argue that the final ensemble model has good performance merely because it is an ensemble of several classifiers (but not due to the improvement by clustering), one may also argue that the final ensemble model has good performance merely due to a good clustering (but not due to the local classifiers trained on the clusters). Hence, firstly we examine the effects of clustering by removing the clustering results (so the only difference is that we replace the obtained clusters with random clusters) and comparing the final classification performance. We also examine the limitations of such clustering-based ensemble approach (i.e. its high dependency on the clustering results). Then we examine the effects of classification by removing all the classifiers (so we assign the labels of new instances to their closest cluster centers) and comparing the final classification performance.

Effects of clustering in this ensemble approach

To examine the effects of clustering, we train two groups of ensemble models and evaluate their performance (PAUC_{100}) on the testing set:

- For the first group of ensemble models (with the effects of clustering), we choose "clustering inside each main class separately" as clustering scheme and only perform clustering within the abnormal class (keep the normal class as a whole), we choose K-means (with different choices of K) as clustering algorithm, CCOvA as classifier training scheme, L1-SVM as base classification model, PS as score pre-processing method, and SC (soft combination) as classifier combination scheme.
- For the second group of ensemble models (without the effects of clustering), the only difference compared with the first group is that we put abnormal instances *randomly* (without replacement) in the clusters that are generated in the first group, so the size of each random cluster in the second group is the same as the cluster size in the first group. For example, in the first group if we choose K = 3 and generate 3 abnormal clusters with 50, 30 and 60 instances respectively, then in the second group we generate 3 random clusters that also have 50, 30 and 60 abnormal instances respectively. And all the other steps (such as calculating the cluster centers, combining the classifiers based on distances) are exactly the same.

We denote the models of the first group (with the effects of clustering) as K_{x1} where x is the choice of K (number of clusters) when performing K-means within the abnormal class and 1 means that we keep the normal class as a whole. And we denote the models of the second group (without the effects of clustering) as R_{x1} where R means random and x is the number of random clusters. The classification performance of these two groups of models (with different values of x) are shown in Figure 6.8.

As we can see in the figure, the predictive performance of the clustering group is significantly higher than the random group, which validates that the clustering indeed significantly improves the performance of the final ensemble model and it is necessary in the clustering-based ensemble approach. We can also see that the predictive performance of the random group (ensemble models with random clusters) is just a bit higher than the single global linear L1-SVM model, which is as expected because each individual classifier is trained on a random abnormal cluster and the final ensemble model is similar to a Bagging ensemble. We also compare our approach with a simple Bagging ensemble of linear classifiers in the next section (Figure 6.9).

Besides, we can also see that the predictive performance of the clustering group varies quite a lot with different x values (number of K-means clusters in the abnormal class), although all choices of x are consistently higher than the random group and the single global classifier. Hence, the main limitation of such clustering-based ensemble approach is that the predictive performance and interpretability of the final ensemble model highly depends on the "quality" of the clustering

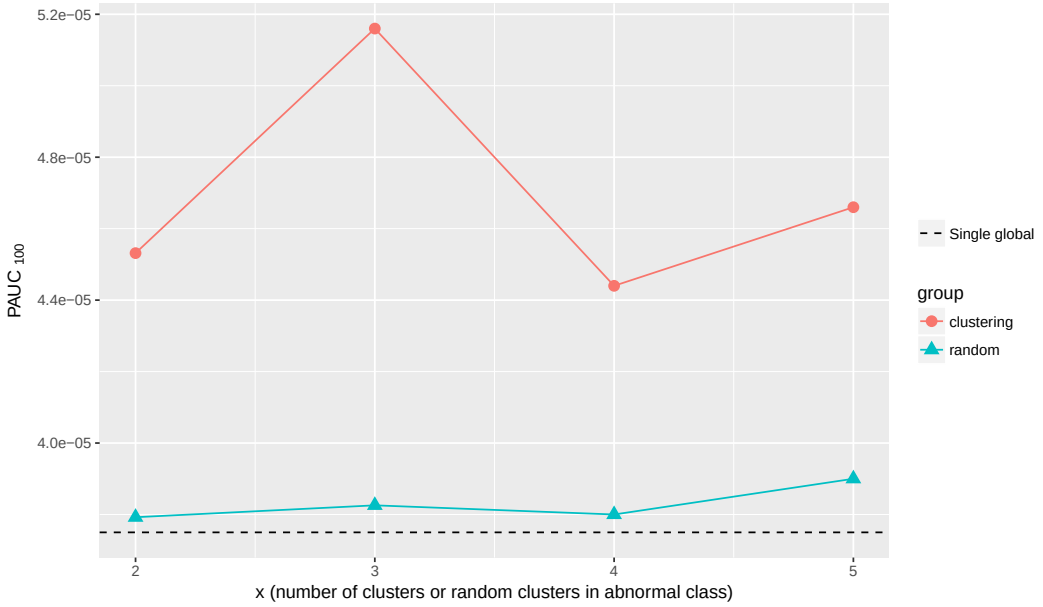


Figure 6.8: Performance comparison between ensemble models with clustering and without clustering (random clusters)

results, which is not easy to evaluate in practice. As discussed in Section 6.3.1, we evaluate and choose the number of abnormal clusters as 3 considering the following metrics: the average Silhouette index (as shown in Figure 6.6) at K=3 is higher than K = 4 and K = 5, the time distribution of the obtained clusters when K = 3 is widely and evenly spread, and the instance distribution of the obtained clusters when K = 3 is more balanced than K = 2. We can also see that the shape of the predictive performance of final ensemble model when $3 \leq K \leq 5$ (as shown in Figure 6.8) is similar to the shape of the average Silhouette index (as shown in Figure 6.6), which implies the final ensemble model's dependency on the clustering results, although we do not continue the experiments for larger K due to time limitations.

In practice, such limitation can be overcome by evaluating the above three metrics (internal index, time distribution and instance distribution) and the interpretability of obtained clusters. Furthermore, we can also try different clustering settings and choose the one whose final ensemble model has a good balance between predictive performance and interpretability. The above steps usually require human effort (e.g. domain experts to check if the clusters make sense) and they are not like some off-the-shelf algorithms (e.g. Random Forest) that can "click and run", but the clustering-based ensemble approach can offer the users a possibility to gain more knowledge about the data by clustering and improve the predictive performance by an ensemble of linear classifiers based on the obtained clustering results. If the obtained clusters have meaningful and practical interpretations (e.g. each cluster may correspond to one type of abnormal transactions), the local classifiers in the final ensemble model can also have good interpretability because each local classifier corresponds to one obtained cluster in the CCOvA training scheme.

Effects of classification in this ensemble approach

To examine the effects of classification, we simply examine the case of K31, where we have 3 abnormal clusters and keep the normal class as a whole. By removing the effects of classification, we mean that we do not build local classifiers on the obtained clusters, but instead use the clustering results directly for classification, i.e. we predict the label of a new instance directly based on its distances to the 3 abnormal cluster centers and the normal class center. We consider two prediction rules and examine their TPR and FPR:

- For a less strict prediction rule, we may predict a new instance to be abnormal if its *smallest* distance to all the 3 abnormal cluster centers is smaller than its distance to the normal class center (i.e. it is closer to one of the 3 abnormal clusters than the normal class). This classification rule (totally based on the cluster distance information) yields TPR = 90% and FPR = 20%, which severely violates the requirement of FPR $\leq$ 100 per 1 million (0.01%) defined by the bank.
- For a stricter prediction rule, we may predict a new instance to be abnormal if its *largest* distance to all the 3 abnormal cluster centers is smaller than its distance to the normal class center (i.e. it is closer to all of the 3 abnormal clusters than the normal class). This classification rule yields TPR = 6% and FPR = 160 per 1 million, which also severely violates the requirement of TPR > 50% and FPR < 100 per 1 million.

Hence, we conclude that only directly using the clustering results (distances to different cluster centers) for classification has bad predictive performance, and we should build local classifiers based on the clustering results for more fine-grained classification.

Summary of validation and limitation

In summary, we can conclude that:

1. Both clustering and classification (local classifiers) are necessary in our clustering-based ensemble approach, and removing any one of them significantly decreases the performance of the final ensemble model.
2. The main limitation of the clustering-based ensemble approach is that both predictive performance and interpretability of the final ensemble model have high dependency on the clustering results. Although we can overcome this limitation by evaluating the obtained clusters (as discussed in Section 6.3.1) and trying different clustering settings to choose the one whose final ensemble model has good performance and interpretability, it still requires much more human efforts and interventions than other off-the-shelf algorithms such as Random Forest.

6.3.5 Comparisons with state-of-art classification techniques

In this section, firstly we compare the predictive performance (PAUC_{100} values) of the models trained by our clustering-based ensemble approach (K31_PS_SC_3 as a representative) and the state-of-art classification techniques (Random Forest as a representative). For more reasonable comparisons, we also compare the above 2 models with a single global linear classifier and a Bagging ensemble of linear classifiers. Then we analyze the interpretability of the clustering-based ensemble approach, e.g. the number of used features, and the feature structure of each local classifier.

Predictive performance

For the predictive performance, the PAUC testing results of the Random Forest model, the single global L1-SVM model with feature selection, the Bagging of 20 L1-SVM classifiers (without feature selection), and the clustering-based ensemble model (K31_PS_SC_3) are shown in Figure 6.9. For the Bagging of 20 L1-SVM classifiers (without FS), we randomly create 20 bootstrapped samples of the same size as the original model training set (i.e. 10769 instances) and train L1-SVM models on those samples (all using the same parameters $C_p = 0.01$ and $C_n = 0.3$, which are tuned by CV on a single classifier), then we combine those 20 individual classifiers by simply taking the average of their scores. The PAUC values and number of used features of the above models are shown in Table 6.16.

As we can see in the figure and table, the clustering-based ensemble model (K31_PS_SC_3) achieves predictive performance ($\text{PAUC}_{100} = 5.16 \times 10^{-5}$) as good as the Random Forest model

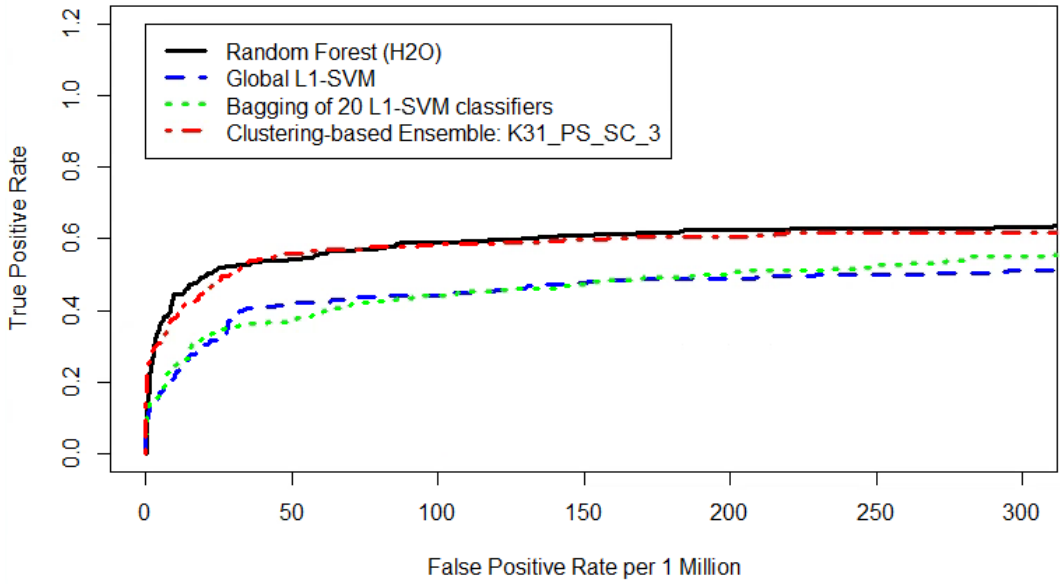


Figure 6.9: PAUC comparison between the Random Forest, single global L1-SVM, simple Bagging of 20 L1-SVM classifiers, and clustering-based ensemble approach

Table 6.16: PAUC values and model details of Random Forest, global L1-SVM, simple Bagging, and the clustering-based ensemble model

Models	$PAUC_{100}$ ($\times 10^{-5}$)	$PAUC_{300}$ ($\times 10^{-4}$)	AUC	No. of used one-hot encoded features	No. of used original features
Random Forest	5.22	1.76	0.981	#	542
Global L1-SVM	3.75	1.35	0.983	429	210
Simple Bagging	3.61	1.36	0.983	399	248
Clustering-based Ensemble	5.16	1.73	0.983	743	310

(5.22×10^{-5}) on this transaction dataset. Our ensemble model can be viewed as a significant improvement of the single global L1-SVM model, since the base model and training set of our ensemble model is exactly the same as the single global L1-SVM model. Besides, we can also see that the clustering-based ensemble model significantly outperforms the simple Bagging of 20 L1-SVM linear classifiers with average rule. The simple Bagging model has a bit lower performance compared with the single global model, which may be due to the facts that we fix the cost sensitive parameters for all individual classifiers but the imbalanced ratios for different samples are different.

We can see from the PAUC figure that the clustering-based ensemble approach theoretically can achieve TPR = 0.56 and FPR = 50 per 1 million. Since there are around 1 million normal transactions and 10 abnormal transactions every day, the clustering-based ensemble model can identify 5 out of the 10 abnormal transactions (56%) while only making 50 false positives every day on average. All the above performance results are based on the testing set (in a whole month), now we can select a classification threshold for Random Forest and the clustering-based ensemble model respectively and perform a daily testing (aggregate transactions on each day) as follows.

As discussed in Section 3.3.5, we aggregate all the transactions in the last 3 weeks of the training set, evaluate the two models on the aggregated sample, and calculate the top 5×10^{-5} (50 per 1 million) scores as the model thresholds. The calculated threshold for Random Forest is 0.06416, and the threshold for the clustering-based ensemble model (K31_PS_SC_3) is 0.01447. We remove the days that do not contain any abnormal transactions (otherwise the TPR is undefined), which ends up with 26 days of aggregated transactions in the testing set. Then we apply the

two models with the thresholds on the daily transactions and calculate the (FPR, TPR) pair for each day, and we show the daily testing results of both models in Figure 6.10. In the figure, the

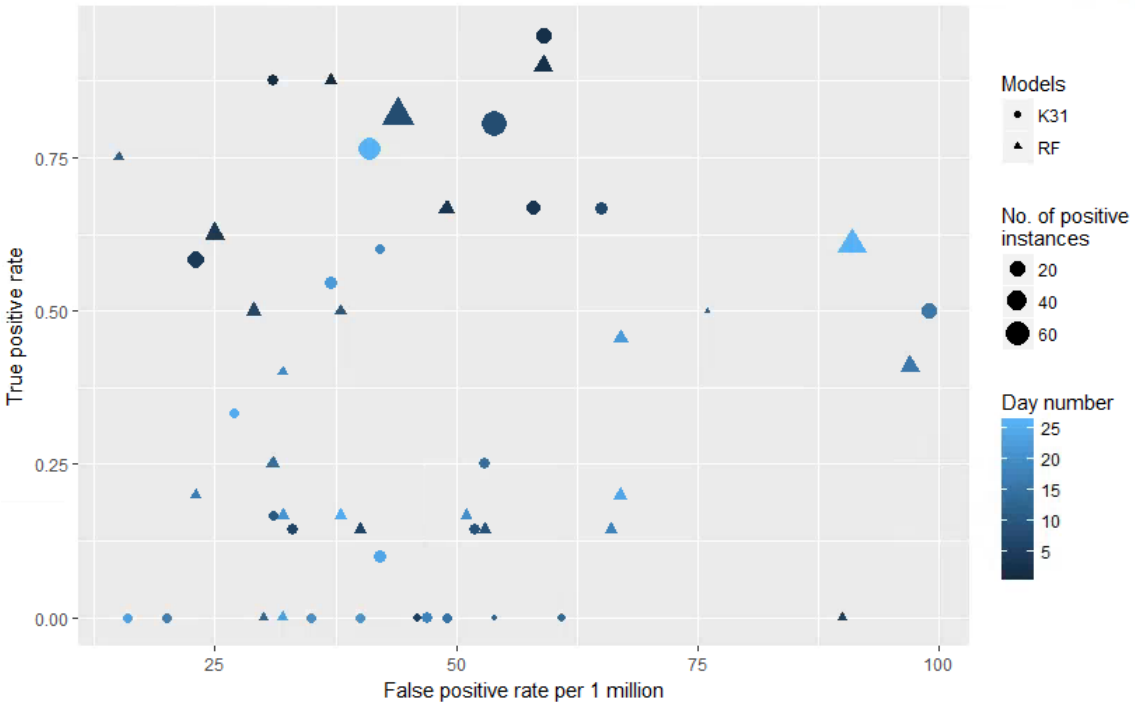


Figure 6.10: Daily testing results of the K31 ensemble model and Random Forest model

type of points represents the type of models (the points for Random Forest are triangles and the points for K31 ensemble model are circles), the size of points represents the number of abnormal transactions on that day (the results are more reliable if there are more abnormal transactions on one day), and the color of points represents the number of days (a darker point means that day is closer to the time span of the training set, and a lighter point means that day is further from the training set). As we can see in the figure, for both models the darker points locate in the upper part and the lighter points locate in the bottom part of the figure, which means that the performance of both models degrade gradually by time and reflects the reality that there are new and unseen types of frauds. We can also see that both models satisfy the low FPR requirement and the K31 ensemble model tends to produce even less FPs. We also calculate the (FPR, TPR) pair for both models using their thresholds on the whole month dataset, the Random Forest model has the performance (FPR = 50 per 1 million, TPR = 0.544), and the K31 ensemble model has the performance (FPR = 44 per 1 million, TPR = 0.550). The K31 ensemble model performs even better than the RF model when choosing their thresholds based on the requirement of FPR $\approx$ 50 per 1 million.

Interpretability

The above experiment results are concerned with the predictive performance, and now we analyze the interpretability of the clustering-based ensemble approach (we take the above K31 ensemble model as a representative). The first simple metric of interpretability that we can use is the number of used features. As we can see in Table 6.16, the Random Forest model uses 542 features while the K31 ensemble model (comprised of 3 local and 1 global linear classifiers) only uses 310 (original) features. Hence, the K31 ensemble model is easier to interpret in terms of the number of used features. However, the number of used features is just one single metric that cannot completely reflect the interpretability of models (it remains an open question to define and

measure interpretability).

We also consider the model structure of the ensemble method, which is hard to quantify. Both the Random Forest and the clustering-based ensemble are ensemble methods. The Random Forest is a tree-based bagging model, although a single decision tree is very easy to interpret, there are 500 trees in the above Random Forest model and each tree has depth 20 and 337 leaves on average. As discussed in Section 5.1, the model structure of Random Forest is not easy to interpret because of the randomness introduced into the model. Although we can calculate some feature importance indexes, such as the average Gini index reduction for each feature, the exact behavior and impact of each feature are not clear due to the complex interactions (lots of deep trees in the Random Forest model) between the features.

On the other hand, the clustering-based ensemble model only involves a relatively small number of sparse linear classifiers, for example, the above K31 ensemble model consists of only 4 individual linear classifiers. Firstly, each individual classifier contained in the clustering-based ensemble approach is easy to interpret especially when we use the one-hot encoding for the categorical features, i.e. the feature weights of each feature bin value (category) are the influences on the classification scores if that feature bin value is present. Positive feature weights have positive influence on the scores (making the transaction more likely to be abnormal) and the negative feature weights have negative influence (making the transaction less likely to be abnormal). The absolute values of the feature weights can be viewed as the magnitude of the effects. Furthermore, we use sparse linear classifiers by L1-regularization as the base models for the clustering-based ensemble approach, i.e. the number of features being used in the sparse linear classifier is relatively small compared with the number of input features. Hence, each individual linear classifier contained in the clustering-based ensemble approach is easy to interpret since the influence of each feature (among a relatively small number of used features) is explicit and fixed. Secondly, although the final ensemble model (i.e. the K31 ensemble model) is not linear anymore since it involves a weighted sum based on the instance's Euclidean distances to different cluster centers in the feature space, it is still easy to interpret considering the following reasons:

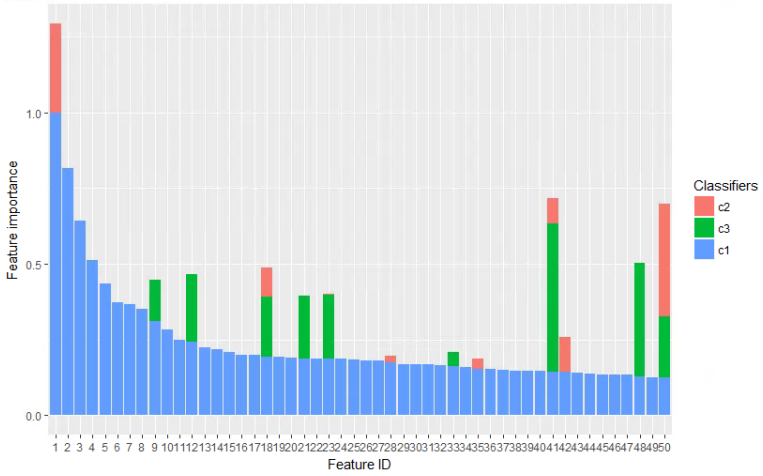
1. As we can see in Table 6.11, each comprised local classifier (Local 1, 2, and 3) in the K31 ensemble model is sparse (using less than 200 original features).

2. As we can see in the following Figure 6.11, the top features (we measure the importance of features in linear classifiers by the absolute values of the feature weights) of the three local linear classifiers are almost non-overlapping. Hence, each local classifier is doing their own job, i.e. each of them is specialized in identifying a certain obtained abnormal cluster.

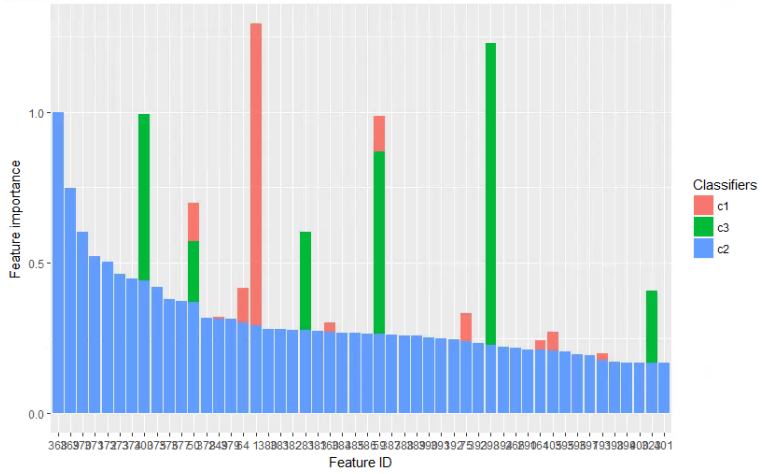
3. The interpretation of each local linear classifier can also be linked to the interpretation of the corresponding obtained cluster, since each classifier is trained on each cluster and the other whole main class according to the CCOvA scheme. As discussed in Section 6.3.1, one possible interpretation of the obtained 3 abnormal clusters by the Rabobank domain experts is that each abnormal cluster corresponds to one type of transaction sources with their typical fraud patterns.

4. The combination (weighted sum) of local classifiers based on the instance's distances to different cluster centers can be interpreted as giving different levels of trust to the local classifiers based on distance information.

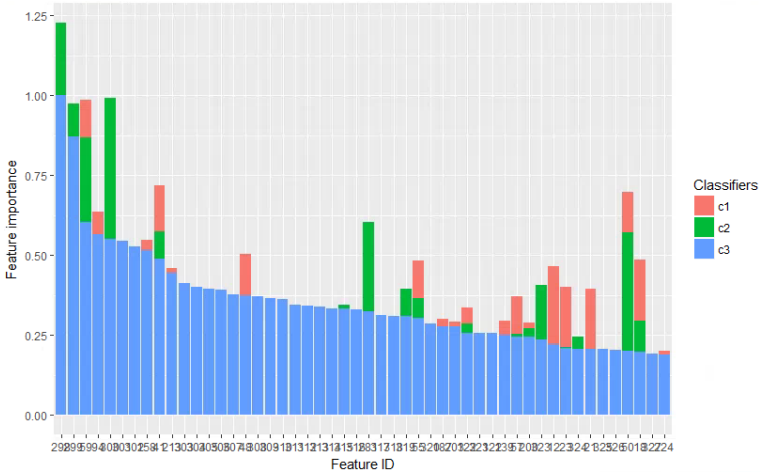
Now as we can see in Figure 6.11, we take the top 50 features with largest absolute weights from each of the 3 local classifiers, and we present them in 3 different bar charts. There are 50 features (bars) in each figure, the bars are sorted by the feature importance of the corresponding classifier, and the length of the bars of specific colors represent the importance of that feature in the corresponding classifier. We can see that most of the top features of each local classifier are almost non-overlapping with each other, even if there are some top features overlapping, the overlapping features that are important in one local classifier are less important in other local classifiers. Hence, each local classifier can be seen as doing their own jobs, and we can interpret their specializations by the top features of each local classifier. Furthermore, according to the



(a) Local classifier 1



(b) Local classifier 2



(c) Local classifier 3

Figure 6.11: Top 50 features of the 3 local classifiers

CCOvA scheme, the practical meanings or specializations of those local classifiers can also be linked to the abnormal clusters obtained in the clustering section. Hence, we conclude that the final ensemble model of the clustering-based ensemble approach is still relatively easy to interpret even if it is not linear anymore in the feature space.

In summary, we conclude that:

- For predictive performance: we compare our proposed clustering-based ensemble model (K31) with the state-of-art Random Forest, the single global L1-SVM model, and the simple Bagging ensemble of 20 L1-SVM classifiers on this transaction dataset. The clustering-based ensemble model has predictive performance as good as the Random Forest on this transaction dataset and both of them satisfy the TPR and FPR requirement defined in the research problem. The clustering-based ensemble model also significantly outperforms the single global L1-SVM model and the simple Bagging ensemble of linear classifiers.
- Besides testing on the one-month testing set, we also test the clustering-based ensemble model and the Random Forest model on daily basis by choosing classification thresholds. The daily testing results show that both models gradually degrade by time.
- For interpretability: we compare the number of used features and model structure of the clustering-based ensemble model (K31) and the Random Forest model. We show that each local classifier of the K31 ensemble model is linear and sparse, and the top features of the obtained 3 local classifiers almost do not overlap with each other. According to the CCOvA scheme, the interpretation of the local classifiers can be linked to the obtained clusters, and one possible interpretation of the 3 obtained abnormal clusters by the Rabobank domain experts is that each abnormal cluster corresponds to one type of transaction sources with their typical fraud patterns. Furthermore, the "Soft Combination" (weighted sum) of local classifiers based on the instance's distances to different cluster centers can be interpreted as giving different levels of trust to the local experts. Hence, we conclude that the final ensemble model (K31) is still relatively easy to interpret.

6.4 Conclusion of Experiments

From the experiments of the state-of-art classification techniques, we can conclude that:

1. PAUC is more reasonable than the whole AUC as the model evaluation metric considering the realistic requirement of low FPR.
2. L1-SVM performs better than L1-LR on this transaction dataset.
3. The stable feature selection technique based on sampling and averaging the feature vectors can improve the predictive performance of linear classifiers.
4. The two tree-based non-linear ensemble models Random Forest and XGBoost achieve great performance on this transaction dataset, although they are not easy to interpret (compared with a single decision tree).
5. The single linear classifier L1-SVM is easy to interpret, but its predictive performance is not good enough compared with the Random Forest model.

From the experiments of the clustering-based ensemble approach, we can conclude that:

1. Clustering experiments:
    - We evaluate the K-means algorithm with Euclidean distance and K-medoids algorithm with Manhattan distance on mixed dataset using the external index ARI, and the clustering results show that the K-means works better than K-medoids on this transaction dataset.

- We perform K-means clustering within the abnormal class and normal class separately. We choose K=3 for abnormal class and K=2 for normal class considering the internal index Silhouette index, cluster instance distribution and time distribution.
- The clustering results of K-modes with Hamming distance are almost the same as K-means with Euclidean distance, hence we use K-means in the following experiments.
- One possible interpretation of the obtained 3 abnormal clusters by the Rabobank domain experts is that each abnormal cluster corresponds to one type of transaction sources with their typical fraud patterns.

2. Comparisons within the CCOvA scheme:
   - There is no significant difference between the two clustering choices K31 and K32.
   - The Platt's scaling as the score pre-processing method can significantly improve the performance of the ensemble model when the base models are L1-SVM classifiers.
   - The classifier combination method SC (weighted sum) based on the instance's distances to different cluster centers has the highest performance in this ensemble approach.
3. The CCOvA classifier training scheme outperforms the other two common schemes (OvO and OvA) in terms of predictive performance and interpretability when only simple rules are used for the two common schemes.
4. Validations and limitations:
   - We validate that both clustering and classification (local classifiers) are necessary in our clustering-based ensemble approach, and removing any one of them will significantly decrease the performance of the final ensemble model.
   - The main limitation of the clustering-based ensemble approach is that both predictive performance and interpretability of the final ensemble model have high dependency on the clustering results. Although we can overcome this limitation by evaluating the obtained clusters (as discussed in Section 6.3.1) and trying different clustering settings to choose the one whose final ensemble model has good performance and interpretability, it still requires much more human efforts and interventions than other off-the-shelf algorithms such as Random Forest.
5. Comparisons with the state-of-art classification techniques:
   - The clustering-based ensemble model (K31) achieves similar predictive performance ($\text{PAUC}_{100} = 5.16 \times 10^{-5}$) as the Random Forest ($\text{PAUC}_{100} = 5.22 \times 10^{-5}$) on this transaction dataset.
   - By choosing the ensemble classification threshold 0.01447, the clustering-based ensemble model (K31) gets FPR = 44 per 1 million and TPR = 0.55 on the one-month testing set, which has satisfied the TPR and FPR requirement in the research problem. In contrast, the Random Forest model gets FPR = 50 per 1 million and TPR = 0.544.
   - The daily testing results of the K31 ensemble model and Random Forest model show that both models gradually degrade by time, hence require periodic updates to keep up with the unseen types of abnormal transactions.
   - The clustering-based ensemble model (K31) uses less features (310 original features) than the Random Forest (542 features). Each local classifier in the K31 ensemble is linear and sparse, and most of their top features do not overlap with each other, which means that they are doing their own jobs. Furthermore, according to the CCOvA scheme, the interpretation of the local classifiers can be linked to the 3 obtained clusters (see the above clustering conclusions). And the "Soft Combination" scheme can be interpreted as giving different levels of trust to the local experts based on distance information. Hence, we conclude that the clustering-based ensemble model (K31) is still relatively easy to interpret.

Chapter 7

Conclusion

In this thesis, we aim to develop and apply classification techniques for fraud detection in bank transactions. The research problem is formulated as a supervised binary classification problem with specific challenges (i.e. large volumes of data, high dimensional, highly imbalanced, and sub-classes may exist in the two main classes) and model requirements (i.e. output continuous confidence scores, low FPR, and easy to interpret).

In the following two sections, firstly we conclude our contributions in terms of academic and business values. Then we discuss the limitations and future work.

7.1 Contributions

In this section, firstly we conclude our academic contributions. Then we conclude our contributions in terms of business values.

7.1.1 Academic contributions

In this thesis, we proposed a clustering-based ensemble of sparse linear classifiers such that the final ensemble model has higher predictive performance than single linear classifiers for binary classification problems where the two main classes may have well-separated sub-classes, and the final ensemble model is also possibly easy to interpret, depending on the interpretability of the obtained clusters.

We decomposed such idea into six components (i.e. clustering schemes, clustering algorithms, base classification models, classifier training schemes, score pre-processing, and classifier combination schemes). The main contributions lie in the classifier training schemes and classifier combination schemes, where we proposed the CCOvA (Class Crossing One-vs-All) scheme for training local linear classifiers on the obtained clusters and the Soft Combination (weighted sum) scheme for combining the classification scores from local classifiers based on the instance's distances to different cluster centers. We showed that the CCOvA scheme has higher predictive performance and interpretability than the other two common schemes (i.e. OvA and OvO) when combined with the Soft Combination scheme.

According to the CCOvA (Class Crossing One-vs-All) scheme, the interpretations of the local classifiers can be linked to the obtained clusters (i.e. each local classifier can be interpreted as specialized in identifying a certain obtained cluster). Hence, if the obtained clusters have meaningful and practical interpretations (e.g. different sub-classes in the main class), the final ensemble model is also easy to interpret since the Soft Combination scheme can be simply interpreted as giving different levels of trust to the local specialized classifiers.

We also validated the effects of clustering and classification (local classifiers) in the clustering-based ensemble approach. Furthermore, we examined the limitations of this approach and we concluded that the main limitation is that the predictive performance and interpretability of the

final ensemble model have high dependency on the clustering results. Although we also proposed several ways to overcome this limitation (e.g. trying different clustering settings, evaluating the obtained clusters by several metrics), such approach still requires more human efforts and interventions than other off-the-shelf classification algorithms such as Random Forest.

We compared the above clustering-based ensemble approach with several state-of-art classification techniques (e.g. Random Forest, simple Bagging ensemble of linear classifiers) on a real transaction dataset provided by Rabobank. We showed that it significantly outperforms the simple Bagging ensemble of linear classifiers, and it has similar predictive performance compared with Random Forest on this transaction dataset.

However, we did not evaluate the proposed approach on other datasets with different characteristics due to time limitations. Hence, some future work (e.g. evaluate the approach on a dataset where the two main classes are compact and do not have well-separated sub-classes) need to be done to further investigate the performance and behaviour of such approach in different situations.

7.1.2 Business values

We made the following contributions in terms of business values for Rabobank who provided the real and anonymized transaction dataset:

- We identified and removed several artifact features that were later confirmed by the domain experts in Rabobank. We also illustrated and validated that the PAUC as a model evaluation metric is more suitable than the whole AUC when the model is required to have low FPR in our case.

- We explored different state-of-art classification models with specific techniques to address the above challenges and model requirements, e.g. down-sampling when growing each tree in the Random Forest model to address the highly imbalanced challenge. Furthermore, the features and weights of the single L1-SVM model (L1-SVM-FS-0.06-5 as shown in Section 6.2.2) were considered to be reasonable and useful by domain experts in Rabobank, i.e. the signs and magnitudes of the feature weights correctly reflect their effects and importance in fraud detection.

- We explored and applied different unsupervised clustering algorithms such as K-means and K-modes on the transaction dataset. The generated 3 clusters of the abnormal data by K-modes were easy to interpret (since each cluster in the K-modes algorithm is represented by a center that consists of the most frequent categories of each feature). One possible interpretation by the Rabobank domain experts of the above 3 clusters is that each cluster corresponds to one type of transaction sources (e.g. Internet, mobile payment) with their typical fraud patterns.

- We trained and tested the clustering-based ensemble model (K31 ensemble model as shown in Section 6.3.5) on the real transaction dataset, it achieved $\text{PAUC}_{100} = 5.16 \times 10^{-5}$ on the one-month testing set. Using the chosen threshold 0.01447 (it was calculated by aiming at FPR = 50 per 1 million), the ensemble model achieved FPR = 44 per 1 million and TPR = 0.550 on the one-month testing set, which satisfied the TPR and FPR requirements (TPR > 0.5 and FPR < 100 per 1 million) defined by Rabobank.

- Besides testing on the one-month scale, we also performed model testing on daily basis (as shown in Figure 6.10). The daily testing results showed that the FPR requirement was satisfied not just on the monthly basis, but also on daily basis (i.e. the FPR on each day was smaller than 100 per 1 million with no exception). Besides, the daily testing results provided a rough estimation of how fast the model degrade by time, hence it also gave an idea to choose a suitable model update timing.

- Besides predictive performance, we also showed that the K31 ensemble model was relatively easy to interpret since each local classifier was linear and sparse, and the top features of each

local classifier almost did not overlap with each other. According to the CCOvA scheme, the interpretations of the local classifiers can be linked to the 3 obtained abnormal clusters (shown in the above clustering results). Furthermore, the "Soft Combination" (weighted sum) scheme can be interpreted as giving different levels of trust to the local classifiers based on distance information.

7.2 Limitations & Future Work

We conclude the limitations and future work of this thesis as follows:

- We only evaluated the proposed clustering-based ensemble approach on the real transaction dataset provided by Rabobank, so the performance comparison with other state-of-art classification techniques was only applicable on this transaction dataset. For future work, we can evaluate the proposed approach on public datasets to have more reliable comparisons. And some future work (e.g. evaluate the approach on a dataset where the two main classes are compact and do not have well-separated sub-classes) need to be done to further investigate the performance and behaviour of such ensemble approach in different situations.
- We only evaluated the models in a static training and testing manner. Although our dataset splitting manner was based on time span (training on previous 4 months and testing on the last month) to make the testing results more realistic and reliable, there are better evaluation schemes such as prequential evaluation (test and retrain the new instances), which is more realistic considering that we need to update the model periodically in reality. For future work, we can evaluate the models using the prequential evaluation.
- We only provided a rough explanation of interpretability, since interpretability is rather subjective, and it remains an open question to define and measure interpretability of machine learning models. For future work, we can provide a running example to illustrate why the obtained clustering-based ensemble model is easy to interpret.
- We only compared the CCOvA (Class Crossing One-vs-All) scheme with the other two common schemes (i.e./ OvO and OvA) when the simple rules (such as average rules) were used for combining classifiers. However, we did not compare with those schemes when the most common classifier combination rule (i.e. voting) was used, since we used PAUC as evaluation metric and our model was required to output (nearly) continuous scores (the outputs of voting are discrete especially when the number of classifiers is small). For future work, we can still compare with those two common schemes when voting is used, we can do this by adjusting the voting threshold of each involved local classifier, and we should have a set of (TPR, FPR) as outputs, then we can compare this sets of points with the partial ROC curve of the model trained by the CCOvA scheme.

Bibliography

[1] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, 1(Dec):113–141, 2000. 34

[2] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984. 38

[3] Adrien Bibal and Benoît Frénay. Interpretability of machine learning models and representations: an introduction. In *Proc. ESANN*, pages 77–82, 2016. 27, 44

[4] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 19

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 20

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009. 2

[7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 42

[8] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 9

[9] Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. *University of California, Berkeley*, 110, 2004. 20

[10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016. 21

[11] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1995. 34

[12] Dmitriy Fradkin. Clustering inside classes improves performance of linear classifiers. In *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on*, volume 2, pages 439–442. IEEE, 2008. 42

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 21

[14] H2O.ai. *R Interface for H2O*, July 2017. R package version 3.10.5.3. 50

[15] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. 22

[16] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304, 1998. 31

[17] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985. 77

[18] Nathalie Japkowicz. Supervised learning with unsupervised output separation. In *International conference on artificial intelligence and soft computing*, volume 3, pages 321–325, 2002. 42

[19] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1–6, 1998. 32

[20] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002. 20

[21] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002. 9

[22] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632. ACM, 2005. 37

[23] Didrik Nielsen. Tree boosting with xgboost-why does xgboost win" every" machine learning competition? Master's thesis, NTNU, 2016. 21

[24] Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010. 8

[25] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999. 37

[26] Ashfaqur Rahman and Brijesh Verma. Novel layered clustering-based approach for generating ensemble of classifiers. *IEEE Transactions on Neural Networks*, 22(5):781–792, 2011. 42

[27] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971. 77

[28] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987. 77

[29] Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, and Zhuang Wang. Log-based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1867–1876. ACM, 2014. 25

[30] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. vii, 22, 23

[31] Brijesh Verma and Ashfaqur Rahman. Cluster-oriented ensemble classifier: Impact of multicluster characterization on ensemble classifier learning. *IEEE Transactions on knowledge and data engineering*, 24(4):605–618, 2012. 42

[32] Ricardo Vilalta, M-K Achari, and Christoph F Eick. Class decomposition via clustering: a new framework for low-variance classifiers. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 673–676. IEEE, 2003. 42

[33] Gary M Weiss. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19, 2004. 9

[34] Junjie Wu, Hui Xiong, Peng Wu, and Jian Chen. Local decomposition for rare class analysis. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 814–823. ACM, 2007. 42

[35] Hongshan Xiao, Zhi Xiao, and Yu Wang. Ensemble classification based on supervised clustering for credit scoring. *Applied Soft Computing*, 43:73–86, 2016. 42

[36] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005. 31

[37] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699. ACM, 2002. 37

Appendix A

Appendix

A.1 Definitions of ARI

The ARI (adjusted Rand index), proposed by [17], is an adjusted version of the originally proposed Rand Index by [27]. Let $I = \{i_1, i_2, \ldots, i_n\}$ be the whole set of n data instances, $X = \{X_1, X_2, \ldots, X_l\}$ and $Y = \{Y_1, Y_2, \ldots, Y_r\}$ be two partitions of the I, so there are l clusters in X and r clusters in Y. Let a be the number of *pairs of instances* that are in the same subset in X and in the same subset in Y, let b be the number of *pairs of instances* that are in different subsets in X and in different subsets in Y. The RI (Rand index) is defined as $RI = \frac{a+b}{\binom{n}{2}}$, which means that it measures the agreement between the two partitions. Now if the partition X is the ground-truth label set, then this RI index can be used to measure how close the cluster Y is compared with the ground-truth label set X. However, the expected value of the RI of two random partitions does not equal to 0. Hence the ARI is defined as $\frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$ such that the expected value of ARI of two random partitions are 0. We refer to [17] for more detailed definitions of ARI (it involves the contingency table).

A.2 Definitions of average Silhouette Index

The Silhouette Index was proposed by [28]. Given a dissimilarity measure $d(x, y)$ (e.g. Euclidean distance), a data partition of c clusters $P = \{P_1, P_2, \ldots, P_c\}$, we define the average dissimilarity of an instance i to a given cluster P_j as $d(i, P_j) = \frac{\sum_{j \in P_j} d(i,j)}{|P_j|}$. Now we assume i is in cluster P_k, and we define the average dissimilarity of instance i with all the other instances in the same cluster P_k as $a(i) = d(i, P_k)$. And we define the smallest average dissimilarity of i to any other cluster (not P_k) as $b(i) = min_{j \neq k} d(i, P_j)$. The Silhouette Index for instance i is then defined as $s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$. The $a(i)$ measures the dissimilarity of i to its own cluster, and $b(i)$ measures the dissimilarity of i to its second best cluster, so a larger $s(i)$ implies that such clustering is good for the instance i. Hence, the average Silhouette Index defined as $average(s(i))$ measures how tightly grouped all the data in each cluster are.

A.3 Figures and tables

1	C_p	C_n	Num. Original features	Num. Feature bins	PAUC_300	PAUC_whole
2	0.1	40	370	998	0.000182210338681	0.979052752313057
3	0.06	80	409	1149	0.000180840336134	0.97390813173755
4	0.15	50	384	1140	0.000179283167813	0.98141107079196
5	0.08	40	370	933	0.000176932773109	0.97902054367202
6	0.04	20	318	631	0.000173427552839	0.975492027417036
7	0.2	50	384	1218	0.000170732111026	0.982964425770314
8	0.08	30	345	832	0.000170527119939	0.979825403191585
9	0.1	10	285	656	0.000170323402088	0.983232452678045
10	0.1	90	417	1338	0.000169937611408	0.974365338256524
11	0.08	15	304	678	0.000169921059333	0.98196756217639
12	0.2	70	408	1374	0.000169349376114	0.980630833545546
13	0.25	80	414	1480	0.00016837916985	0.981962049062049
14	0.1	80	415	1303	0.000168360071301	0.975582147101272
15	0.04	50	375	867	0.000168357524828	0.970275651472716
16	0.15	7	272	672	0.000167797300738	0.985295516085222
17	0.1	100	424	1392	0.000167377132671	0.972504256854265
18	0.15	40	372	1073	0.000167187420423	0.98122875816994
19	0.15	1.5	230	544	0.000167007894067	0.985799545878957
20	0.1	7	267	594	0.000166615737204	0.984454280196928
21	0.06	30	344	773	0.000166605551311	0.977790777523137
22	0.2	40	369	1122	0.000166054239878	0.981712859689334
23	0.2	30	348	1017	0.000165674815381	0.98412398989899
24	0.1	15	306	717	0.000165451998981	0.983555033528564
25	0.15	30	354	986	0.000165446906035	0.982398680078094
26	0.1	50	384	1076	0.000165249554367	0.979354067990837
27	0.15	0.8	221	505	0.000165247007894	0.986689260249553
28	0.2	80	414	1443	0.000165066208302	0.980235410831006
29	0.15	1	225	520	0.000164856124268	0.98632109752992
30	0.1	1	216	454	0.000164668958492	0.985654053136406
31	0.08	10	285	615	0.000164649859944	0.982206387403449
32	0.2	5	264	670	0.000164475426534	0.98560581444699
33	0.15	80	411	1360	0.000164470333588	0.980391673032859
34	0.15	2	237	556	0.000164463967405	0.985996621678974
35	0.08	60	396	1103	0.000164455054749	0.977536779560319
36	0.1	20	325	788	0.000164283167813	0.982179583651648
37	0.15	70	402	1272	0.000164283167813	0.979279795857745
38	0.15	10	286	711	0.000164274255157	0.984394571768101
39	0.4	150	445	1934	0.000164094728801	0.980550358628308
40	0.15	5	261	635	0.000164087089381	0.985369696969696

Figure A.1: An example of parameter tuning details for the L1-SVM model, sorted by $PAUC_{300}$

Table A.1: Instance distribution and time distribution of K-means (K=2) within normal data

		No. of instances	Min_time	Max_time	Mean_time	SD_time
K = 2	**Cluster 1**	6266	20161001	20170131	20161204	34 days
	Cluster 2	3734	20161001	20170131	20161203	34 days

Table A.2: Testing results for different combinations of ensemble components

Classifier Schemes	Clustering Choices	Score Preprocess	Classifier Combination	$PAUC_{100}$ ($\times 10^{-5}$)	$PAUC_{300}$ ($\times 10^{-5}$)	AUC
CCOvA	K_{31}	#	MIN	2.39	9.55	0.929
		#	MAX	4.62	16.53	0.981
		#	AVG	4.22	15.38	0.976
		#	HC	4.51	16.35	0.982
		#	HS_2	4.42	16.51	0.985
		#	HS_3	4.52	16.64	0.985
		#	SC_2	4.48	16.54	0.983
		#	SC_3	4.61	16.78	0.984
		SD	MIN	2.47	9.83	0.932
		SD	MAX	4.49	16.41	0.978
		SD	AVG	4.32	15.31	0.978
		SD	HC	4.29	16.04	0.982
		SD	HS_2	4.29	16.11	0.985
		SD	HS_3	4.34	16.27	0.985
		SD	SC_2	4.52	16.48	0.982
		SD	SC_3	4.59	16.79	0.983
		PS	MIN	2.41	10.01	0.941
		PS	MAX	5.08	17.15	0.979
		PS	AVG	5.06	17.21	0.98
		PS	HC	4.95	16.81	0.982
		PS	HS_2	5.13	17.29	0.983
		PS	HS_3	5.14	17.32	0.983
		PS	SC_2	5.15	17.27	0.982
		PS	SC_3	5.16	17.29	0.983
	K_{32}	#	MIN	2.4	9.56	0.929
		#	MAX	0.47	2.55	0.904
		#	AVG	3.63	13.5	0.984
		#	HC	4.11	15.52	0.981
		#	HS_2	4.8	17.01	0.984
		#	HS_3	4.78	16.98	0.984
		#	SC_2	4.1	15.18	0.986
		#	SC_3	4.29	15.96	0.986
		SD	MIN	0.74	3.44	0.949
		SD	MAX	4.41	16.28	0.974
		SD	AVG	4.36	15.5	0.98
		SD	HC	4.3	16.05	0.977
		SD	HS_2	3.01	12.85	0.985
		SD	HS_3	3.3	13.78	0.985
		SD	SC_2	4.48	16.45	0.984
		SD	SC_3	4.49	16.63	0.985
		PS	MIN	0.96	4.6	0.973
		PS	MAX	5.06	17.12	0.979
		PS	AVG	5.09	17.26	0.981
		PS	HC	4.97	16.84	0.977
		PS	HS_2	5.1	17.28	0.98
		PS	HS_3	5.1	17.27	0.981
		PS	SC_2	5.14	17.29	0.982
		PS	SC_3	5.15	17.29	0.983