On some studies of Fraud Detection Pipeline and related issues from the scope of Ensemble Learning and Graph-based Learning

Tuan TRAN

Supervisor: Dr. An MAI

Quantitative and Computational Finance
John von Neumann Institute

This dissertation is submitted for the degree of
Master of Science

JVN Institute October 2018

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Tuan TRAN
October 2018

Acknowledgements

I would like to thank Dr. An Mai for his supervision and interesting discussions on several research topics. A great thank you also goes to Loc Tran who helps me complete this thesis.

This work has been partially funded by Vietnam National University - Ho Chi Minh City (VNU-HCM) under grant number C2018-42-02.

Abstract

The UK anti-fraud charity Fraud Advisory Panel (FAP) in their review of 2016 estimates business costs of fraud at £144 billion, and its individual counterpart at £9.7 billion. Banking, insurance, manufacturing, and government are the most common industries affected by fraud activities. Designing an efficient fraud detection system could avoid losing the money; however, building this system is challenging due to many difficult problems, e.g.imbalanced data, computing costs, etc. Over the last three decades, there are various research relates to fraud detection but no agreement on what is the best approach to build the fraud detection system. In this thesis, we aim to answer some questions such as i) how to build a simplified and effective Fraud Detection System that not only easy to implement but also providing reliable results and our proposed Fraud Detection Pipeline is a potential backbone of the system and is easy to be extended or upgraded, ii) when to update models in our system (and keep the accuracy stable) in order to reduce the cost of updating process, iii) how to deal with an extreme imbalance in big data classification problem, e.g. fraud detection, since this is the gap between two difficult problems, iv) further, how to apply graph-based semi-supervised learning to detect fraudulent transactions.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Credit Card Fraud

1.1.1 Definition and impact of Credit Card Fraud

A Criminal Code [101] which is a law that codifies most criminal offenses and procedures in Canada, section 380 provides the general definition for fraud in Canada:

1. Every one who, by deceit, falsehood or other fraudulent means, whether or not it is a false pretense within the meaning of this Act, defrauds the public or any person, whether ascertained or not, of any property, money or valuable security or any service,

   (a) is guilty of an indictable offense and liable to a term of imprisonment not exceeding fourteen years, where the subject-matter of the offense is a testamentary instrument or the value of the subject-matter of the offense exceeds five thousand dollars; or

   (b) is guilty

      i. of an indictable offense and is liable to imprisonment for a term not exceeding two years, or

      ii. of an offense punishable on summary conviction, where the value of the subject-matter of the offense does not exceed five thousand dollars.

In recent years, e-commerce has gained a lot in popularity mainly due to the ease of cross-border purchases and online credit card transactions. While e-commerce is already a mature business with many players, security for online payment lags behind. With an extensive use of credit cards, fraud seems like a major issue in the credit card business. It

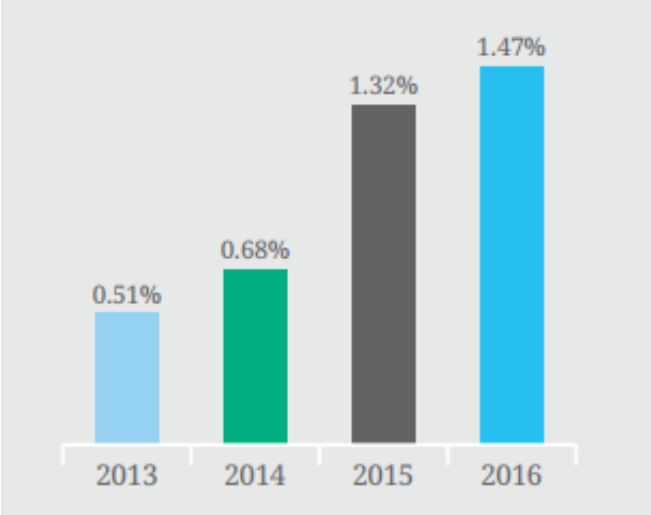


Fig. 1.1 Cost of Fraud as a % of Revenues Keeps Going Up [106]

is not easy to see the impact of fraud since companies and banks have a good motivation not to disclose a full report of losses due to frauds. However, as stated in the Lexis Nexis's study [105], in 2014 fraudulent card transactions worldwide have reached around $11 billion a year. A latest study of Lexis Nexis [106] estimated that a cost of fraud as a percentage of revenues keeps going up, from 0.51% in 2013 increasing to 1.47% in 2016 (Figure 1.1), and a total costs of fraud losses for each dollar is up to 2.4 times (Figure 1.2). In the United Kingdom, total losses through credit card fraud have been growing rapidly from £122 million in 1997 to £440.3 million in 2010 according to an estimation of the Association for Payment Clearing Services (APACS) [31].

There are many types of fraud, but in this thesis we focus on credit card fraud. The credit card frauds might occur in several ways, for instance:

- Card-holder-not-present fraud is a payment card transaction made where the card-holder doesn't or can't physically present the card for a merchant's visual examination at the time that an order is given and payment effected, for example a transaction is ordered via mail, telephone or Internet,
- Stolen card fraud is the most common type of fraud where the stolen card may be used for illegal purchases as much as possible and as quickly as possible until the card holder notifies and blocks the account.

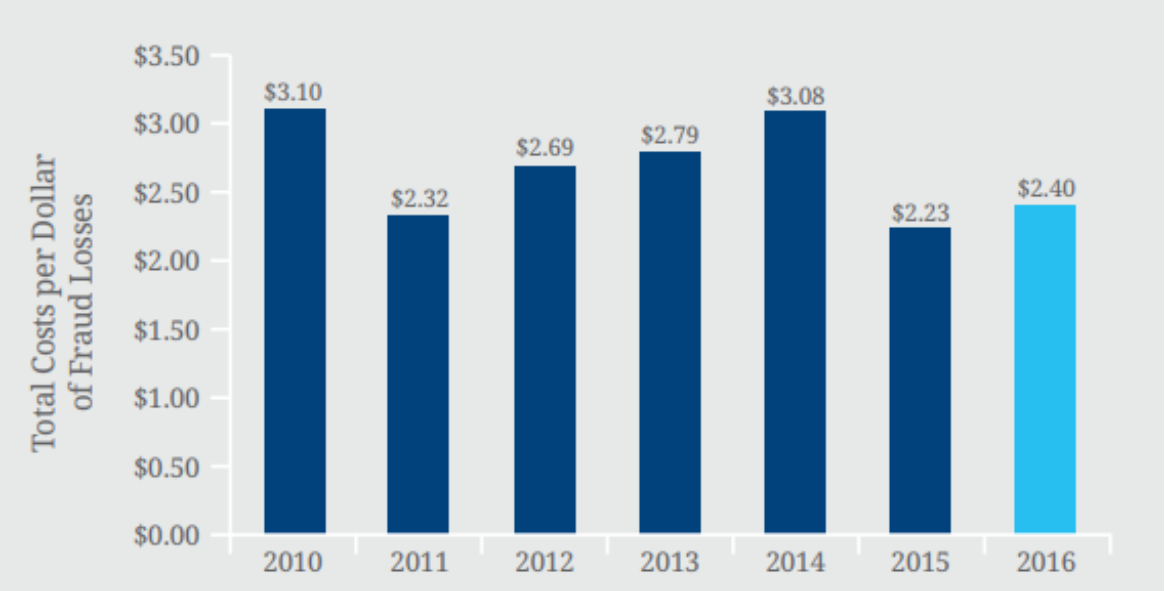


Fig. 1.2 Total Costs per Dollar of Fraud Losses [106]

1.1.2 Credit Card Fraud Detection

Detecting credit card fraud attracts a lot of attention from industry (bank, insurance, . . .) and from the research community for the last three decades. The fraud detection process is based on the analysis of recorded transactions, which are a set of attributes (for example identifier, transaction timestamp, amount of the transaction, . . .). Traditional methods of data analysis have long been used to detect fraud, it requires large human resource, time-consuming inquiry, wide range of knowledge to analyze a huge number of transactions. However, the human resource is limited and thus automatic fraud detection system is really necessary because it is not easy for a human analyst to detect fraudulent patterns in transaction datasets with a huge number of samples and many attributes. With a rising of Machine Learning field, which is able to learn patterns from data automatically and have been applied in several applications, this is a suitable approach for fraud detection problem. However, a design for the Fraud Detection System (FDS) is particularly challenging for several reasons:

- The number of credit card frauds is really small - a tiny fraction of all the daily transactions. Many algorithms in Machine Learning have not been designed to deal with this problem,
- Many fraud transactions have an insignificant difference to normal transactions, which makes these transactions hard to detect by Machine Learning algorithms,

- Over the time, new attack strategies are created and changes in customer behavior make the distribution of data change over time and it does not meet the basic assumption in Machine Learning (as well as Statistics),
- ....

1.2 Contributions

Focusing on the credit card fraud detection problem and some other related issues, in this thesis we present a series of independent contributions through this thesis, including:

Fraud Detection Pipeline

Despite rich literature on fraud detection, there is no agreement on what is the best way to solve this problem and also there is no public fraud detection system. This means that in the industry, when a company wants to build their own a fraud detection system, they cannot decide which is a good way to do so, especially for those companies in Vietnam. The first main contribution of this thesis is a survey on the fraud detection problem, we will show readers several challenges in building the Fraud Detection System. Based on that, we propose one system that not only good but also simple that companies could implement by themselves easily, which we call Fraud Detection Pipeline. Furthermore, we give readers many possible ways to extend the system.

Update Point Estimation for the case of Concept-Drift in fraud detection problem

Fraudsters always try to create new fraud strategies and normal customers usually change their behavior, these changes make non-fraud/fraud patterns in our dataset change over time. In this case, the system has to update its models frequently to keep its accuracy, either on a daily or a less frequent period. With the credit card fraud detection problem, a common choice is daily when we receive a daily full dataset and then update the models at night. However, updating models daily is worthless and actually it is only necessary when we do not have to adapt to new fraudulent patterns. In other words, we should update models if and only if it makes an improvement to our system. The second contribution is a mechanism that helps us make a decision on when we should update our models.

K-Segments Under Bagging approach: An experimental Study on Extremely Imbalanced Data Classification

Imbalanced dataset could be found in many real-world domains, e.g. fraud detection problem, and there are several methods have been proposed to handle the imbalance problem, but there is no guarantee those methods will work with an extreme imbalance case. Big Data is a problem concerning the volume and the complexity of data, whereas the big data in extremely imbalance data is a different problem from those original issues that attracts much attention recently. In this study, as the third contribution, we show that a simplified combination of under-sampling and ensemble learning is very effective in the case mentioned.

Solve fraud detection problem by using graph based learning methods

To detect the credit cards' fraud transactions, data scientists normally employ the unsupervised learning techniques and supervised learning technique. Among several algorithms that have been proposed for this problem, a graph-based semi-supervised learning have not been applied to the credit cards' fraudulent transactions detection before. We propose to apply an un-normalized graph p-Laplacian based semi-supervised learning technique combined with an undersampling technique to find a relationship of those frauds in the data.

1.3 Outline

This thesis is organized as follows:

- Chapter 2 prepares the necessary knowledge which would be helpful for the next chapters,
- Chapter 3 reviews several challenges of the fraud detection problem and their possible solutions, based on that we select the most suitable approaches and combine them into one best strategy as in our proposed Fraud Detection Pipeline,
- Chapter 4 uses the Fraud Detection Pipeline and presents a mechanism to detects which model need to update to keep the accuracy of our system and also reduces the update cost,
- Chapter 5 explains more details of our selected techniques in the pipeline how they are useful to us in the case of the extremely imbalanced big dataset,
- And finally, chapter 7 summarizes our results and proposes future research directions.

Chapter 2

Preliminaries

2.1 Machine Learning

Machine learning is a field in computer science that we try teaching the computers to do tasks without explicitly programmed [121, 80]. Machine learning is strongly associated with computational statistics, which explores data and builds algorithms that could find patterns in data and make predictions. More formal definition of the algorithms used in the machine learning is provided by Mitchell "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [102].

Typically, tasks in machine learning can be grouped into three wide categories [118]:

- Supervised learning: the given data contains input values and their desired output values, the aim of algorithms in supervised learning is to find a general rule that maps inputs to outputs. In this thesis we are focus on the supervised learning approach for our fraud detection problem,
- Unsupervised learning: there are no desired outputs in the dataset, it means that we give the learning algorithms freely to explore a hidden structure of the data,
- Reinforcement learning: this kind of learning is slightly different with the above, reinforcement learning builds a dynamic environment then let a computer interacts with it to perform a certain goal, after each action a feedback is provided to the computer as a reward or a punishment.

Supervised learning is a most common task in the world and can be seen as a function from labeled training data [103]. Given a set of N data points of a form $(x_1, y_1), \ldots, (x_N, y_N)$

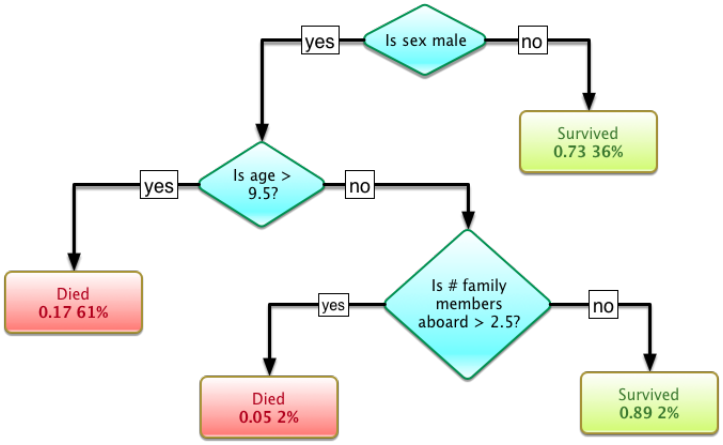


Fig. 2.1 A tree showing survival of passengers on the Titanic

such that x_i is a attribute vector of the i-th data point and y_i is its desired output, a supervised learning algorithm is a function $g : X \to Y$, where X is an input space (matrix of attribute vectors) and Y is a output space (matrix of desired outputs). Typically the function g is in a function space G, or we could represent g as a scoring function $f : X \times Y \to \mathbb{R}$ such that function f will return the output value y which has the highest score $g(x) = argmax_y f(x,y)$. For example, g could be a conditional probability model $g(x) = P(y \mid x)$, e.g. logistic regression [140, 25], or f could takes a form of a join t probability model $f(x,y) = P(x,y)$, e.g. naive Bayes [118].

In order to measure a performance of learning function, we could define a loss function $L : Y \times Y \to \mathbb{R}$ and a risk $R(g)$ of learning function g is defined as an expected loss of function g which could be estimated by:

$$\widehat{R}(g) = \frac{1}{N} \sum_i L(y_i, g(x_i)) \tag{2.1}$$

Decision tree learning

A decision tree learning is one common approach in data mining [117], it aims to create a model that predicts values like making a decision from a tree-based analysis. An example is shown in the figure 2.1 [149].

A tree learner could classify the data points by splitting the input dataset into subsets based on a given criterion. The splitting process at each node is repeated in a recursive procedure until all data points are classified. There are two kinds of decision trees:

- Regression tree: create a tree to predict the desired output as a real number,
- Classification tree: an analysis aims to predict a categorical output.

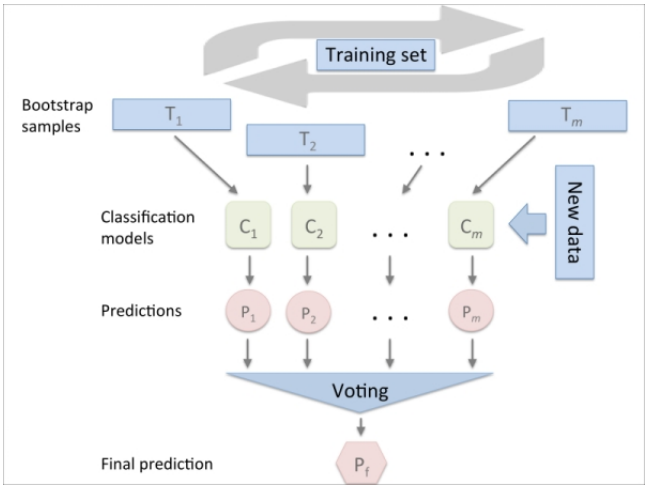


Fig. 2.2 Bagging ensemble [115]

A term Classification and Regression Tree (CART) analysis is used to refer to both the above analysis [15]. There are many decision tree algorithms, including:

- ID3 (Iterative Dichotomiser 3) [112],
- C4.5: an extension of ID3 algorithm [113].

Algorithms for constructing decision trees use different metrics for measuring which is the best feature to split the dataset. These measures compute the homogeneity of the output values within the subsets, some common measures in decision tree learning is Entropy, Information Gain, and Gini.

Ensemble learning

In machine learning, ensemble learning is a method combines multiple learning algorithms to obtain better accuracy than from any individual algorithms [109, 116]. The term ensemble is usually reserved for methods that generate multiple hypotheses using the same base learner. There are three common types of ensemble learning, such as:

Bootstrap aggregating (bagging) An ensemble technique is bootstrap aggregating which is usually called bagging (see figure 2.2), it combines multiple learnings via their equal weight votes. In order to support the model variance, bagging trains each model from a randomly drawn subset. And a famous example of bagging is the Random Forest algorithm which uses random decision trees as base learners.

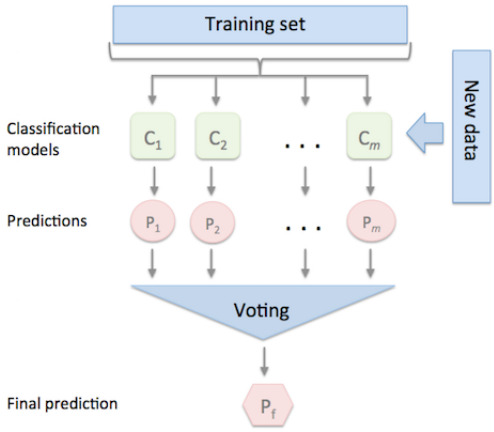


Fig. 2.3 Boosting ensemble [115]

Boosting Similar with the bagging ensemble, boosting is an incremental algorithm that will train each new model to underline the data points that previous models mis-classified (see figure 2.3). Boosting could have a better accuracy than bagging in some cases, however it also tends to overfit to the training data. A most common boosting algorithm is Adaboost [46].

Stacking Slightly different with these above techniques, stacking is a supervised learning algorithm that uses the predictions of several other learning algorithms as an input dataset to makes the final prediction (see figure 2.4). All of the other algorithms are trained using the given dataset independently, then a combiner algorithm is trained on that which can be any algorithm. Therefore, stacking can theoretically represent any of the ensemble techniques, although in practice we often use the Logistic Regression or Linear Regression as the combiner.

Random Forest

Random Forest [14] or random decision forests [66, 67] are an ensemble learning method for classification or regression that perform by constructing many trees, e.g. decision trees, then combine these results to makes final output values as a mode of the classes for classification tasks or a mean prediction for regression tasks. The Random Forest model based on two elements:

1. Base learner: each tree in the forest is the base learner, typically it is a weak learner with high variance,

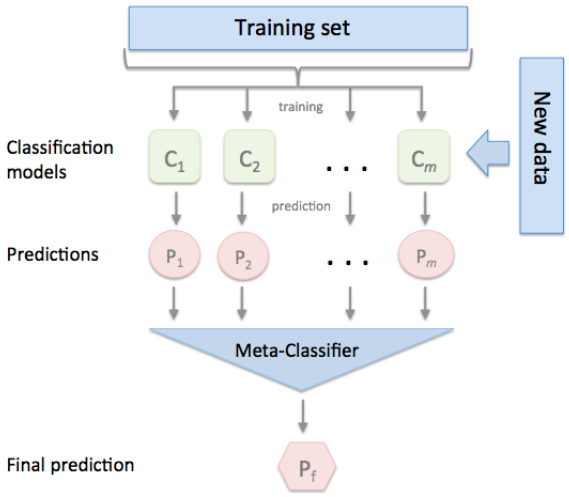


Fig. 2.4 Stacking ensemble [115]

2. Ensemble learning: combine the results of several weak learners to make a final prediction.

The Decision Tree above that are grown very deep tend to overfit their training sets, i.e. have a low bias, but very high variance. We use the tree-based method as the base learner by its strength, and to handle its weakness we combine all of the predictions via ensemble learning. This method is called Random Forest, it can formulate the algorithm as the following statement: given a dataset $X = x_1, \ldots, x_n$ with n data points and respectively desired output values $Y = y_1, \ldots, y_n$, bagging repeatedly B times selects a random sample from the dataset X then training trees on these samples.

Algorithm 1 Tree bagging

function BAGGING(X, Y, B)
  for $b = 1$ to B **do**
    (X_b, Y_b) is samples from (X, Y) which have n data points
    Train a classification or regression tree f_b on X_b, Y_b
  end for
end function

The authors of Random Forest have proposed using an average function to makes the final prediction in regression tasks or takes the majority vote in classification tasks [14]. However, there is no standard function as a combiner, thus we are free to choose the combination function. This bootstrapping procedure could decrease the variance of the model without increasing the bias. In order to estimate the uncertainty of the prediction, it could be computed as a standard deviation of the predictions from all the individual trees:

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x) - \hat{f})}{B-1}} \tag{2.2}$$

Random Forest uses the above bagging algorithm with only one difference, it uses a modified tree that selects a random subset of the features to split a current subset data. Typically, the dataset has p features then for a classification problem $\sqrt{p}$ features will be used in each split and for regression problems the inventor recommends that we should use $p/3$ as the number of features [48].

2.2 Measurement

The fraud detection problem is the classification task and in order to measure how well of the classifiers, there are many metrics could be used to evaluate the algorithms. In this section, we will review some common metrics which are typically used in classification tasks.

Accuracy

Accuracy is a measure of statistical variablity which represent a percentage of correct predictions on the total number of cases examined. In the fields of science and engineering, the accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's true value [9]. Consider 2 sets with n data points: the true labels from the given dataset $Y = y_1, \ldots, y_n$ and our predicted values $\hat{Y} = \hat{y_1}, \ldots, \hat{y_n}$, the accuracy is:

$$accuracy = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{y_i = \hat{y_i}} \tag{2.3}$$

Precision and recall

In pattern recognition, information retrieval and binary classification, precision and recall are two common metrics. Precision (also called positive predictive value) is the fraction of relevant data points among the given data points, and recall (also known as sensitivity) is the fraction of relevant data points that have been retrieved over the total amount of relevant data points (see figure 2.5).

In the information retrieval contexts, precision and recall were defined by Perry, Kent & Berry (1955) [110] as a set of retrieved elements and a set of relevant elements. Precision is the fraction of retrieved documents that are relevant to the query:

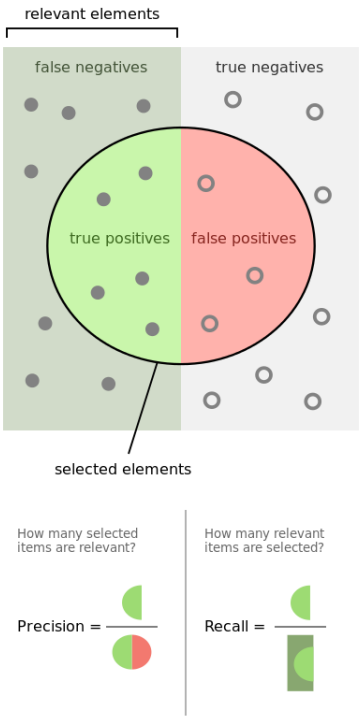


Fig. 2.5 Precision and recall [150]

$$\text{precision} = \frac{|\,\text{relevant elements} \cap \text{retrieved elements}\,|}{|\,\text{retrieved elements}\,|} \tag{2.4}$$

And the recall is the fraction of the relevant elements that are successfully retrieved:

$$\text{precision} = \frac{|\,\text{relevant elements} \cap \text{retrieved elements}\,|}{|\,\text{relevant elements}\,|} \tag{2.5}$$

In the classification tasks, considers four terms true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Precision and recall are defined as [108]:

$$\text{precision} = \frac{TP}{TP+FP} \tag{2.6}$$

$$\text{recall} = \frac{TP}{TP+FN} \tag{2.7}$$

Receiver operating characteristic (ROC)

A receiver operating characteristic curve is a graphical plot which is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

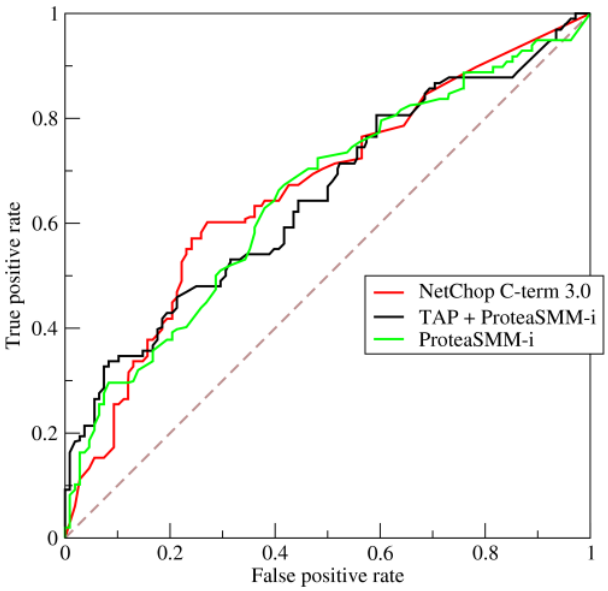


Fig. 2.6 ROC curve of three predictors of peptide cleaving in the proteasome [151]

ROC analysis provides tools to select a possible optimal threshold for models, an example ROC curve plot of three predictors of peptide cleaving in the proteasome is shown in figure 2.6.

Area under the ROC curve (AUC)

We cannot compare classifiers base on the ROC because the ROC curve is just a graphical plot. In order to represent ROC performance as a single scala, a common method is to calculate the area under the ROC curve (AUC for short). Because the AUC is a size of the area of the unit square, therefore the AUC value will always be between 0 and 1. However, random guessing produces an area of 0.5, no realistic classifier should have an AUC less than 0.5 then it makes the baseline performance for all classifiers is 0.5. In summary, the AUC metric is a measure of how much the ROC curve is close to the point of perfect classification.

The AUC metric usually used in machine learning community for model comparison [59]. However, in practice researchers recently noticed that AUC was quite noisy as a classification measure [57] and some other studies have other significant problems in model comparison [94, 58].

F-score

F_1 score (also F-score or F-measure) is a measure that considers both precision and recall by taking a harmonic average of these metrics.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.8}$$

A general formula of F-score for any positive real β is:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{2.9}$$

Van Rijsbergen et al. [135] interpret F_β that "measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision". In practice, the F-score metric is often used in machine learning when the dataset is imbalanced because it isn't affected by the imbalance problem.

Chapter 3

Fraud Detection Pipeline

Standing on the shoulders of giants

— Bernard of Chartres

There are various research and studies on fraud detection problem and related issues, including what algorithms should be used [89, 11, 95], how to prepare features [5, 148, 41], ... In this section, we will review challenges in building the Fraud Detection System and its relevant questions. Based on that, we propose a Fraud Detection Pipeline which is not only simple, reliable but also easy to implement. Furthermore, we provide the readers many potential improvements for the pipeline.

3.1 Imbalanced dataset

One of the main problems of fraud detection is dataset imbalanced, i.e. the number of fraudulent transactions are a small fraction of the dataset [75]. The imbalanced problem is of significant concern in the data mining and machine learning community because imbalanced datasets are common in many real-world domains. For instance, in the detection of fraudulent cases in telephone calls [42] and credit card transactions [17], the number of legitimate transactions heavily outnumbers the number of fraudulent transactions. Learning from imbalanced data sets is an important issue in supervised learning, with the credit card fraud detection problem, the imbalance is a big problem for the learning process. Let consider an example which consists only 1% fraudulent transactions (minority class) and the remaining belongs to legitimate category (majority class), a lazy classification that predicts all of the transactions are legitimate transactions will has 99% accuracy. This is a very high accuracy but we can not use this because it cannot detect any fraudulent transactions at all.

There are several methods that have been proposed to deal with the imbalanced problem and these could be separated into two levels: methods that operate at the data level and at the algorithmic level [21]. Algorithms at the algorithmic level are designed to deal with the minority class detection and at the data level, the rebalanced strategies are used as a pre-processing step to rebalance the dataset before any algorithm is applied.

Algorithmic level methods

At algorithmic level, these algorithms are modified or extended of existing classification algorithms for imbalanced tasks. Based on their styles we can separate these algorithms into two styles: imbalanced learning and cost-sensitive learning. In the first learning, the algorithms try to improve the accuracy of the minority class prediction, while the second learning tries to minimize the cost of wrong predictions.

Imbalance learning

Decision tree, e.g. C4.5 [113], use Information Gain as splitting criteria to maximize the number of predicted instances in each node, it makes the tree bias towards the majority class. Cieslak and Chawla [24] suggest splitting with Hellinger Distance (HD) which they show that HD is skew-insensitive and their proposed Hellinger Distance Decision Tree is of better performance compared to the standard C4.5. Other studies have also reported the negative effect of skewed class distributions not only in decision tree [64, 71], but also in Neural Network [71, 139], k-Nearest Neighbor (kNN) [84, 97, 7] and SVM [154, 153].

In Machine Learning, with a great success of ensemble learning on several applications, many ensemble strategies have been proposed for imbalanced learning. Bagging [13] and Boosting [47] are the most popular strategies which combine the imbalanced strategy with a classifier to aggregate classifiers [92, 144, 138, 76, 93, 142, 18, 74, 98].

Cost-sensitive learning

In classification applications dealing with imbalanced datasets, the correct prediction of minority class is more important than the correct prediction of majority class, which causes several classifiers to fail when predicting minority class because they assume the cost of these classes are the same. In the credit card fraud problem, the cost of our true prediction is zero, but if we cannot detect a fraud transaction then its amount of money is our lost and if we predict a non-fraud is a fraud then the cost is an investigation fee need to correct this transaction (see table 3.1 for a simple cost matrix).

Table 3.1 A simple cost matrix for one transaction.

	Non-fraud	Fraud
Predict non-fraud	0	its amount of money
Predict fraud	investigation fee	0

Classifiers in cost-sensitive learning use different costs for prediction of each class, these cost-based classifiers could handle wrong-prediction costs without sampling or modifying the dataset. For example in tree-based classifiers, cost-based splitting criteria are used to minimize costs [91]; or used in tree pruning[12]. An extension of cost-sensitive learning, Domingos proposed Metacost [32] framework that transforms non-cost-sensitive algorithm into a cost-sensitive algorithm. However, the cost for minority class is usually not available or difficult to compute, which makes the cost-sensitive algorithms are not popular [96].

Data level methods

Methods at data level are techniques that modifying an imbalanced dataset before any classifiers could be applied. In this section, we will introduce some popular techniques usually used to re-balance the distribution of the classes.

Sampling

Several studies [146, 86, 36] have shown that balanced training set will give a better performance when using normal algorithms, and sampling methods are the most common

techniques in data science to create a balanced training set. There are three popular sampling techniques, including undersampling, oversampling and SMOTE (see figure 3.1); and some extension sampling techniques are based on these three techniques, e.g. Borderline-SMOTE [56], ADASYN [63], ...

Undersampling [33] is a method to decrease the size of the majority class by removing instances at random. The idea is that many instances of the majority class are redundant and the removal of these instances, thereby, makes its distribution change not too much. However, the risk of dropping redundant instances still exists since this process is unsupervised and we cannot control what instances will be dropped. Furthermore, a perfectly balanced dataset, which means the size of the minority class equals the size of the majority class, is not a good choice for undersampling [27]. In practice, this technique is often used because it is simple and speeds up our process.

Oversampling [33] is an opposite method of undersampling that tries to increase the size of minority class at random. While duplicating the minority class, oversampling increases a risk of overfitting by biasing the classifier toward the minority class [33]. Furthermore, this method does not add any new information for minority instances and it also slows down our learning phase.

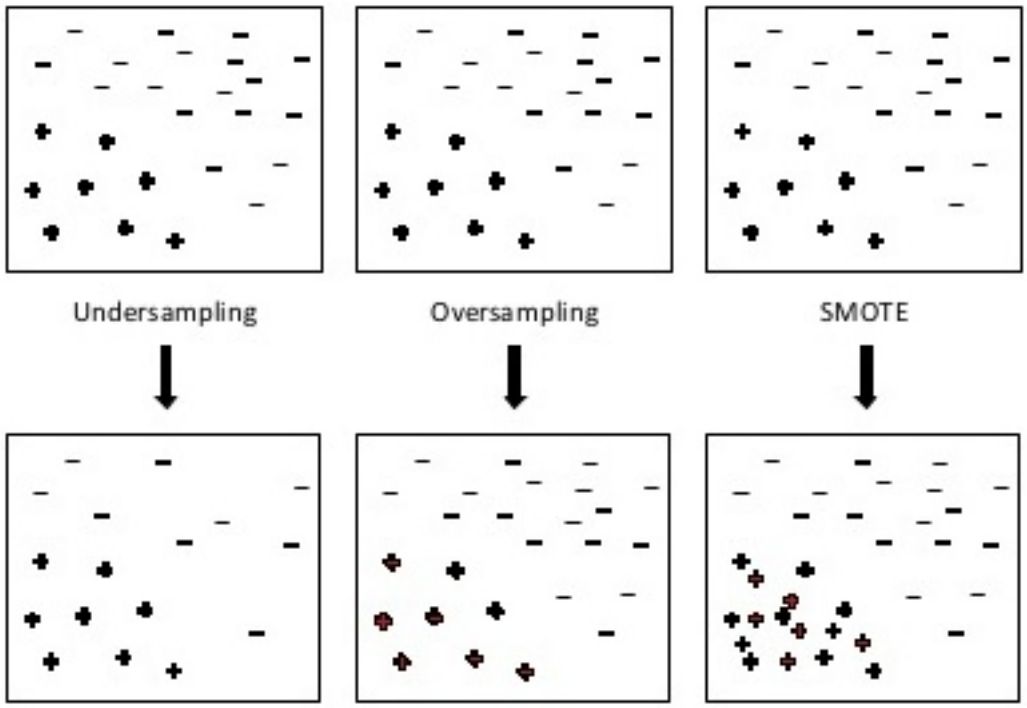
SMOTE [20] is a small extended version that combines both sampling techniques above, abnormal oversampling and normal undersampling. While oversamples the minority class by creating similar instances in a neighborhood area and also do undersampling the majority instances randomly, it could create clusters around each minority instance and helps classifiers build larger decision regions. SMOTE has shown to increase the accuracy of classifiers [20], a change of learning time depends on sampling ratio but it still has some drawbacks, e.g. new minority instances are created without considering its neighborhood so it increases the overlapping area between the classes. [141].

Cost-based

Cost-based methods is a kind of sampling technique that considers the misclassification cost which is assigned to each instance a different value. The above sampling techniques are random, with cost-based sampling methods, the weight of minority class is usually higher than the weight of majority class and then other sampling techniques could be used to rebalance the datasets.

Costing [159] is a cost-based undersampling that draws instances with an acceptable probability greater than a pre-defined threshold. Klement et al. [79] proposed a more complicated and effective approach that combines cost-based random under-sampling and ensemble learning. The cost-based methods are promising when dealing with imbalanced

Fig. 3.1 Three common sampling methods [29]



datasets, however, the cost of misclassification for each class may be not easy to see in the real life as we have mentioned in the section above.

Distance-based

Distance-based methods are slightly different than cost-based methods, i.e. instead of considering the cost of misclassification. These methods consider a distance among instances in the imbalanced dataset to undersample or to remove a noise and borderline instances of each class. They need to compute the distance between instances so these methods are more costly than the above methods.

Tomek [129] proposed a method that removes instances from the majority class that is close to the minority region in order to have a better separation between two classes. His method is useful in noisy datasets or datasets with the overlapping problem, i.e. removing those instances can make the classifier toward misclassification [125]. There are several distance-based sampling studies, e.g. Condensed Nearest Neighbor [61], One Sided Selection [84], Edited Nearest Neighbor [152] or Neighborhood Cleaning Rule [86].

3.2 Algorithms

In the fraud detection problem, almost Machine Learning algorithms, including supervised learning [17, 84, 8], unsupervised learning [128, 11], ..., have been proposed to detect fraudulent transactions. In this section, we will see which algorithms are used to solve the fraud detection problem.

Expert Systems

In the very early 90s, the decision which decides a transaction is a fraud or not follows from rules and the rules are generated from the knowledge of human expert. Expert's rules are just simple IF-THEN rules and extremely manually since all of the rules based on the expert's knowledge and ability. By applying expert system, suspicious activity or transaction can be detected from deviations from normal spending patterns [90]. Kevin at el. proposed expert system model to detect fraud for alert financial institutions [89]. Nik presented a FUZZY system to detect credit card frauds in various payment channels, their model fuzzy expert system gives the abnormal degree which determines how the new transaction is fraudulent in comparison with user behavioral [60].

Association Rules

Association rules are an extended approach of Expert Systems in which we add a certainty factor for expert's rules. Usual measures for the certainty are proposed by Agrawal et al [1] for establishing an association rule's fitness and the interest are the confidence $Conf(A \Rightarrow C)$, the conditional probability $p(C \mid A)$, the support $Supp(A \Rightarrow C)$, and the joint probability $p(A \cup C), \ldots$.

There are few studies use association rules and its extension, e.g. fuzzy association rules [122]. However, the expert's rules are still generated manually and it is the biggest disadvantage of those methods. In the current computing age, a complexity of the data is increasing fastly and we need to create rules or make fraud prediction automatically.

Neural Networks

The neural networks are non-linear statistical data modeling tools that are inspired by the functionality of the human brain using a set of interconnected nodes [155, 52]. Because of its strengths, neural networks are widely applied in various applications such as automobile insurance and corporate fraud. The literature describes that neural networks can be used as

a financial fraud detection tool. The neural network fraud classification model employing endogenous financial data created from the learned behavior pattern can be applied to a test sample [54]. The neural networks can be used to predict the occurrence of corporate fraud at management level [16].

Researchers have explored the effectiveness of neural networks, decision trees and Bayesian belief networks in detecting fraudulent financial statements (FFS) and to identify factors associated with FFS [78]. The study in [40] revealed that input vector consisted of financial ratios and qualitative variables, was more effective when fraud detection model was developed using the neural network. The model was also compared with standard statistical methods like linear and quadratic discriminant analysis, as well as logistic regression methods [40].

The Bayesian belief network (BBN) is a graphical model that presents a probabilistic dependency using a directed acyclic graph (DAG), in which nodes represent random variables and missing edges encode conditional independencies between the variables [78]. The Bayesian belief network is used in developing models for the credit card, automobile insurance, and corporate fraud detection. Bayesian belief network outperformed neural network and decision tree methods and achieved outstanding classification accuracy [78].

Logistic Regression (LR)

Logistic Regression [140, 25] is an important machine learning algorithm. The goal of LR is to model the probability of a target belong to each class. Logistic Regression is a simple and efficient approach that is a widely used technique in such problems [70]. There are many studies tried to use logit models to estimate fraudulent probability in the case of insurance frauds [73, 3] and other related areas like food stamp programs, and so forth [10, 62, 111].

Decision Tree

A Decision Tree is a decision support tool that uses a tree-like graph to find a prediction rule from the labeled dataset. In our fraud detection case, this means that it could generate expert-rules IF-THEN automatically. The Decision Tree has many advantages, e.g. simple to understand and interpret, important insights can be generated automatically like experts, ... and it has found several applications in fraud detection [120, 119, 4, 88].

Random Forest

Random Forest [14] is an ensemble of Decision Trees, where each tree is trained on a different bootstrap sample of the original training set and uses a random subset of all the features available. The forest of Decision Trees that are very different from each other, this diversity is a key factor for variance reduction in order to overcome the disadvantage of Decision Tree [83].

Several studies have shown that it achieves the best results among different classifiers [28, 26, 136, 148, 8, 6] when using Random Forest. Pozzolo et al. [29] have shown that the Random Forest combined with undersampling or SMOTE are often the best choice. In our case, we refer undersampling over SMOTE since our data is usually huge. We propose using Random Forest with undersampling for many reasons, e.g. powerful, fast, simple to interpret, ...

3.3 Sample Selection Bias

A standard assumption, not only in Data Science but also in Statistics, is stationary, i.e. the training set and testing set have the same distribution, since these sets come from one generating source. In a non-stationary environment, there is a distributional mismatch between the training set and testing set and classifiers are fitted using the training set could not use the testing set to check its accuracy.

This situation could occur when we modify the original dataset, e.g. under-sampling, and a newly created training set doesn't represent well for the whole dataset. We called this a Sample Selection Bias (SSB), which also happens in the collecting data step when we do not or could not collect data that present the population distribution. For example, when predicting a new customer that could pay a loan (and its interest) at the end of this month and we give a try with a current data of a bank, however the bank only records old customers that repay and they directly refuse a new bad customer without recorded information.

Let s is a random variable which takes either 1 (sampled) or 0 (not sampled). Using Bayes formula we could re-write the $P(x,y)$ as:

$$P(x,y) = \frac{P(x,y|s=1)P(s=1)}{P(s=1|x,y)} = \frac{P(s=1)}{P(s=1|x,y)}P(x,y|s=1) \tag{3.1}$$

As initially proposed by Zadrozny [157] there were four types of conditional dependencies:

- No *sample selection bias* or $P(s=1|x,y) = P(s=1)$. In other words, the selection process is independent from both feature vector x and class label y,

- *Feature bias* or $P(s=1|x,y) = P(s=1|x)$. The selection process is conditionally independent from y given x . It is important to understand that feature bias does not imply $s=1$ is completely independent from y,
- *Class bias* or $P(s=1|x,y) = P(s=1|y)$. The selection process is conditionally independent from x given y. Similarly, it does not imply that $s=1$ is completely independent from x,
- *Complete bias* or $P(s=1|x,y)$. The selection process is dependent on both x and y.

There are many research studies relate to the SSB [35, 159, 158, 157, 39, 34]. One approach for this problem is *important sampling* that assign a weight to each data point [159, 157, 39]. The $P(x,y)$ re-writes in term of $P(x,y|s=1)$ then we could do sampling with the weight $\frac{P(s=1)}{P(s=1|x,y)}$, but it is not an easy task due to estimating $P(s=1|x,y)$ is not straightforward. Instead of trying to find what sample well represents the dataset, another simpler approach to handle SSB is using ensemble learning to combine results from multiple samples or from multiple classifiers [38].

In the previous section, we preferred to use Random Forest with the undersampling method and this approach could lead to the Sample Selection Bias problem. In order to avoid it, we propose using ensemble method that will combine multiple Random Forests classifiers. Splitting the dataset into multiple subsets by applying the undersampling technique, each subset will be used as the training set of Random Forest. A final prediction could be mean or mode of these classifiers and it makes the final result more stable because we use all of the available data. We will give more details of this method in the following chapter.

3.4 Feature Engineering

One of the most important steps in Machine learning is Feature Engineering which we will pre-process features in the dataset to improve the accuracy of a algorithm. The Feature Engineering usually consist: impute missing values, detect outliers, extract new useful features, ... A set of raw attributes in many credit card datasets is quite similar because the data collected from a transaction must comply with international financial reporting standards (American Institute of CPAs, 2011) [2]. Typical attributes in one credit card dataset are summarized in table 3.2, these attributes could be grouped into four levels:

- Transaction level: these features typically are the raw attributes of the transaction which are collected in real time,

Table 3.2 List of typical raw attributes in a transaction dataset

Attribute name	Description
Transaction ID	Transaction identification number
Time	Date and time of the transaction
Account number	Identification number of the customer
Card number	Identification of the credit card
Transaction type	ie. Internet, ATM, POS, ...
Entry mode	ie. Chip and pin, magnetic stripe, ...
Amount	Amount of the transaction in Euros
Merchant code	Identification of the merchant type
Merchant group	Merchant group identification
Country	Country of transaction
Country 2	Country of residence
Type of card	ie. Visa debit, Mastercard, American Express...
Gender	Gender of the card holder
Age	Card holder age
Bank Issuer	bank of the card

- Card level: spending behavior of a card could be computed by aggregating information from transactions made in last given hours of a card,
- User level: spending behavior of the customer, in some cases one user has only one card and these levels are the same,
- Network level: users in our system are not isolated, they are connected and share the behavior, we will discuss this level in the next section.

There are many studies use only raw transactional features, e.g. time, amount, ... and don't take into account the spending behavior of the customer, which is expected to help discover fraud patterns [87]. Standard feature augmentation consists of computing variables such as average spending amount of the cardholder in the last week or last month, number of transactions in the same shop, number of daily transactions, ... [28, 82, 148, 8, 72].

Whitrow et al. proposed a transaction aggregation strategy in order to aggregate customer behavior [148]. The computation of the aggregated features consists in grouping the transactions made during the last given number of hours by each categorical features, e.g. card id, user id, transaction type, merchant group, country or other, followed by calculating the average/total/... amount spent on those transactions. This methodology has been used by a number of studies [8, 6, 28, 72, 119, 127, 147]. Whitrow et al. [147] proposed a fixed time frame to be 24, 60 or 168h, Bahnsen et al. [5] extended time frames to: 1, 3, 6, 12, 18, 24, 72 and 168h.

The process of aggregating features consists in selecting those transactions that were made in the previous t_p hours, for each transaction i in the dataset S:

$$S_{agg} = T_{agg}(S,i,t_p) \quad = \{x_j^{amount} \mid (x_j^{feature} = x_i^{feature}) \wedge (H(x_i^{time}, x_j^{time}) < t_p)\} \tag{3.2}$$

where:

T: function creates a subset of S associated with a transaction i with respect to the time frame t_p,

x_i^{amount}: the amount of transaction i,

$x_i^{feature}$: the categorical feature of transaction i which is used to group the transactions,

x_i^{time}: the time of transaction i,

$H(t_1,t_2)$: number of hours between the times t_1 and t_2.

After that, we can compute a customer behavior in last given hours, for example, an average amount of this time frame:

$$x_i^{average\_amount} = \frac{1}{N} \sum_{x^{amount} \in S_{agg}} x^{amount} \tag{3.3}$$

with:

$x_i^{average\_amount}$: a new feature computed from subset S_{agg} by function average,

N: number of transactions in subset S_{agg}.

These new useful features above for capturing customer spending patterns, Bahnsen et al. [5] are also interested in analyzing the time of a transaction. The logic behind this is a customer is expected to make transactions at similar hours. However dealing with the time of the transaction is not the same as dealing with the above features since the time in a day is the circle, therefore it is easy to make the mistake of using the arithmetic mean of transactions' time. They have proposed to overcome this limitation by modeling the time of the transaction as a periodic variable, in particular using the von Mises distribution [44].

Recently Wedge et al. [145] tried applying automated feature engineering with some simple functions to reduce the time cost of this time-consuming feature engineering step, they claimed that a number of false positives dropped by 54% compared to their previous approach. In summary, the feature pre-processing is important and some above methods are simple, easy to compute and run in real time. At least, the fraud detection system should have some simple aggregations, e.g. average, for some periods, e.g. 24h.

3.5 Measurement

The most common measure for classification tasks is accuracy; however, in the imbalanced dataset it is a misleading assessment measure, as well as some other measures e.g. MME, BER, TPR and TNR, ... There are some measures could handle this problem, e.g. AUC, F-measure,... and a well-accepted measure for imbalanced classification is the Area Under the ROC Curve (AUC) [19]. AUC shows that how much the ROC curve is close to the perfect classification; however, Hand [58] considers the standard calculation of the AUC as inappropriate because it making an average of different misclassification costs for classes. F-measure is more accurate in the sense that it is a function of a classifier and its threshold setting, i.e. it considers both the precision and the recall of the test to compute the score. The traditional F-measure or balanced F_1 score is the harmonic mean of precision and recall.

In many Fraud Detection System [119, 95, 4], cost-based measures are defined to quantify the monetary loss due to fraud [6] by computes average cost-matrix which is similar to the confusion matrix. Elkan [35] states that it is safer to assess the cost-sensitive problem in terms of benefit (inverse of cost) because there is the risk of applying different baselines when using a cost-matrix to measure overall cost. Dealing with this issue, normalized cost or savings [4] are used to judge the performance w.r.t. the maximum loss.

In the cost matrix, the cost of a missed fraud is often assumed to be equal to the transaction amount [35, 6], because it has to be refunded to a customer. Cost of false alert is considered to be equivalent to the cost of a phone call because our investigators have to make the phone call to the card-holder to verify a transaction whether it is a fraud or not. Furthermore, there are many intangible costs, e.g. reputation cost of a company, maintenance cost of the investigator's department, ... For all of these reasons, define a good cost measure is not easy in credit card fraud detection and there is no agreement on which is an appropriate way to measure the cost of frauds.

In a scenario with limited resources, e.g. there are only a few investigators, they can't check all alert transactions which are marked as fraudulent from the detection system. They have to put their effort into investigating transactions with those highest risk of fraud, in other words the detection system has to give each transaction its posterior fraud probability. With this requirement, the measure must contain the probability part in order to measures how good the system gives the high score to fraud transaction and reverse.

In summary, define a good measure to analyze the accuracy of the system is not easy, not only in our case but also in many other domains of Machine learning. Similar with the imbalanced problem, the cost-based measures seem very promising and we also need to know all of the cost for each instance. The $F-1$ score is better than all of the others, it is

easy to compute and suitable for the imbalance problem, and we thereby will use the $F-1$ measure to evaluate our fraud detection pipeline.

3.6 Fraud Network

In the very early age of the computer, experts have noticed that a fraudulent account is often connected to another fraudster. This is true in many domains, for example in the telecommunication domain, therefore analyzing the links in the data could discover fraud networks and improve the accuracy of the system [41].

The network not only could increase the true positives of our system, but also decrease the false positives. For example, consider the following legal scenario: a husband and his wife traveling to another country, the first transaction of the husband could create a fraud alert then the next transaction of his wife from the same city should be legal with or without confirmation of the first one.

Despite these advantages, aggregating fraud network-level features is not easy and recently had been taken with more consideration. Van Vlasselaer et al. have proposed APATE [136], a framework for credit card fraud detection that allows including network information as additional features describing a transaction. In the next year after APATE, they also proposed GOTCHA! [137], a novel approach which can improve the accuracy of traditional fraud detection tools in a social security context, this approach uses a time-weighted network and features extracted from a bipartite graph. They show that network-level features are able to improve significantly the performance of a standard supervised algorithm.

In summary, finding the network in the dataset is a complicated task and it's not the best choice for the first version of the fraud detection system.

3.7 Concept Drift

As mentioned in section 3.3, the non-stationary environment could occur when we modify the dataset. In case a data generating source changes itself over time in an unforeseen manner, it is known as Concept Drift [50] or Dataset Shift [114]. With Bayes rule, we have a joint distribution of a sample (x,y) is:

$$P(x,y) = P(y|x)P(x) = P(x|y)P(y) \tag{3.4}$$

In classification tasks, we usually want to estimate the probability $P(y|x)$, from the above formula, we have:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \tag{3.5}$$

From this Bayes formula, the change of data distribution at time t and $t+1$ could come from several sources[77]:

- $P(y|x)$: a change occurs with a class boundary $(P_t(y|x) \neq P_{t+1}(y|x))$, this makes any classifiers which are well designed will be biased,
- $P(x|y)$: this change $(P_t(x|y) \neq P_{t+1}(x|y))$ is a internal change within a class which is observed in an inside class but the class boundary isn't affected, also known as Covariate Shift [104],
- $P(y)$: change in prior probability $(P_t(y) \neq P_{t+1}(y))$, it makes a good design classifier become less reliable,
- or combination of these three parts.

There is no $P(x)$ in the list because the change in $P(x)$ does not affect y so we could ignore it [68]. In general, it is hard to say where the change comes from since we only see the generated data and we do not know or cannot control the data generating process. Learning in the non-stationary environment make us have to update models frequently to capture up-to-date patterns and also remove irrelevant patterns.

In fraud detection problem, we saw that fraudsters always try to create new fraudulent strategies to bypass our detection system, and new fraudsters also give a try with old strategies. Therefore, patterns in our system are learned from the past data may re-occur in the future then the removing process makes us losing the accuracy, this problem is known as the stability-plasticity dilemma [55].

In summary, dealing with Concept Drift problem, our fraud detection system has to update frequently to capture new fraudulent patterns as soon as possible, e.g. a bank has up-to-date labels at night then they could update their system daily.

3.8 Delayed True Labels

Most of the supervised learning algorithms are based on the assumption that labels of the dataset are correct; however, in the real world environment it could be not true. In our case, the fraud detection problem, after the system generates alerts then with a limited number of investigators, only a restricted quantity of alerts could be checked. It means a small set

of labeled transactions returned as *feedback* and makes the unrealistic assumption that all transactions are correctly labeled by a supervisor.

Furthermore, non-alerted transactions are a large set of unsupervised instances that could be either fraudulent or genuine but we only know the actual label of them after customers have possibly reported unauthorized transactions and maybe available several days later. And the customers have different habits, e.g. rarely check a transcript of credit card given by the bank, which makes our dataset non-accurate and hard to modeling the fraudulent patterns.

In fraud detection problem, a company, e.g. a bank, usually has all latest up-to-date labels of transactions at night, and other companies could have after a longer time. It means that the system has to be updated frequently, i.e. as soon as possible after receives accurate labels. And the system which is never updated will lose their accuracy over the time.

Pozzolo et al. [26] claimed that the accurate up-to-date labels are very important and they proposed a learning strategy to deal with the feedback. Their method used both feedbacks F_t and delayed supervised instances $D_{t-\delta}$:

- a sliding window classifier W_t: daily updated classifier over the supervised samples received in the last δ + M days, i.e. $\{F_t, \ldots, F_{t-(\delta-1)}, D_{t-\delta}, \ldots, D_{t-(\delta+M-1)}\}$,
- an ensemble of classifiers $\{\mathscr{M}_1, \mathscr{M}_2, \ldots, \mathscr{M}_M, \mathscr{F}\}$ where $\mathscr{M}_i$ is trained on $D_{t-(\delta+i-1)}$ and F_t is trained on all the feedbacks of the last δ days $\{F_t, \ldots, F_{t-(\delta-1)}\}$.

Their solutions used two basic approaches for handling concept drift that can be further improved by adopting dynamic sliding windows or adaptive ensemble sizes [37]. In summary, to keep the accuracy of the system, we need to update all of the classifiers as soon as the data available, and usually it is a daily update.

3.9 Performance

Particularly in fraud detection problem, we are dealing with a huge credit card transaction dataset. As we see in the sections above, many approaches are used to detect fraud transactions, many of them have shown that they have a good result with their approach, however, most of them could not be used in production because they did not consider their running time. For instance, in the age of Neural Network there are many studies try to use Neural Network models to detect fraud, but this approach runs the training process very slowly.

In the age of Big Data, there are some frameworks for distributed computing system and the most common one currently used in many biggest system around the world is Spark [160]. Spark is an open-source cluster-computing framework, originally developed at the

University of California, Berkeley's AMPLab. With Spark framework, we could distribute the computation for each transaction into one node in our cluster, and the cluster is easy to scale up.

Spark also supports Machine learning algorithms via its MLlib library [99]. However, with the current version of Spark (2.2.0 [45]), there are not many algorithms supported, which means that for those companies they have only a few options to build the detection system. Fortunately, some of the most common algorithms are supported, e.g. Linear Model, Decision Tree or Random Forest, ...

3.10 Fraud Detection Pipeline

We have discussed the challenges of building the Fraud Detection System and found out several possible ways to resolve them. Based on that, in this section, we analyze and propose a Fraud Detection Pipeline, which is not only easy to implement but also has a reliable result.

Imbalanced dataset and Algorithm

The imbalance problem and algorithms in the fraud detection problem could be grouped into two groups based on our point of view:

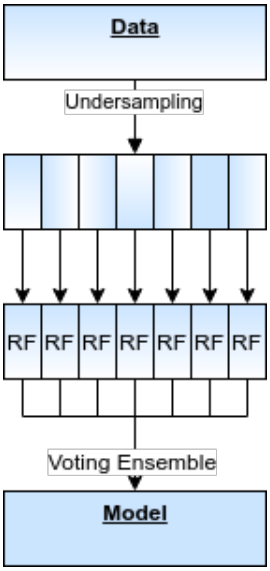
- Theoretical perspective: apply Data Science and Data Mining to obtain a *good* result like other applications,
- Financial perspective: this task is a problem in financial institutions and they want to minimize their losing money.

However the detail of losing money is usually not published since it is sensitive data, therefore those cost-based methods are out of the scope of this report. Among the rest approaches, Pozzolo et al. [29] have shown that the Random Forest combined with under-sampling or SMOTE often has the best result. In our case, we prefer undersampling over SMOTE because it could manage the huge dataset. Usage details of the undersampling and Random Forest algorithm are provided later.

Sample Selection Bias

A chosen sampling technique, undersampling, leads to the Sample Selection Bias problem and in order to avoid it, we propose to use an ensemble method that combines results from multiple samples [38]. The dataset will be split into multiple dis-joint samples, each of which

Fig. 3.2 Building single model by combining multiple Random Forest classifiers (RF)



is used to train a Random Forest classifier then the final prediction could be the mean or the mode value of these classifiers (see figure 3.2). This approach is also useful for the imbalance problem above and reduces training times of the Random Forest classifiers.

Feature Engineering and Fraud Network

Feature Engineering is one of the most important steps in Data Science and usually is difficult, time-consuming, and requires expert knowledge. We don't discuss in a detail this step, but in summary features after this step could be grouped into four levels:

- Transaction level: typically is the raw attributes of a transaction, which are available in real-time,
- Card level: this level describes a spending behavior of a credit card, which is computed by aggregating the features from its previous transactions,
- User level: usually same as card level if one user has only one card, it describes the spending behavior of a customer and a computing method is the same as card level,
- Network level: detect Fraud Network in our system could improve the accuracy significantly but it is a complicated task, therefore we skip this level in our pipeline.

We propose to aggregate features which describe customer's behavior (user level and/or card level) consisting grouping transactions made during a last given number of hours by

each categorical features, e.g. card id, user id, transaction type, ...; followed by calculating some simple functions, e.g. average, amount spent on those transactions. The time frames could be 1, 3, 6, 12, 18, 24, 72, 168 hours and more. The detailed formulas are in section 3.4.

Measurement

Some common measures could not be used to evaluate the Fraud Detection System due to the imbalance problem. From the theoretical perspective, the simplest and suitable measure to deal with this is F1-score. Or inside a financial company, with all cost information on each transaction, we could use cost-based measures for those cost-based methods. In this thesis, we only use F1-score for its simplicity.

Concept Drift

The data, which is fraudulent patterns and customer's behavior, always change over time. Dealing with this Concept Drift problem, all classifiers in our detection system have to update frequently to capture new patterns as soon as possible. We assume that we could update the system daily or at least every time frame, which is possible with a small dataset, and in the remaining of this chapter we will update all models at all time frames.

Performance

As we have described in section 3.9 that to bring this Fraud Detection Pipeline into production, we have to consider its performance. And fortunately for us, the distributed computing framework Spark could handle our requirements, e.g. distribute our computation or run Machine Learning algorithms parallel, ... Therefore we propose to use this framework in both the development and the production phase.

Fraud Detection Pipeline

With the Fraud Detection Pipeline architecture, we propose to build several models on different time frames for multiple purposes, such as:

- short-term model: using small latest data, e.g. one time frame data, to captures newest fraudulent strategies. In real life, it could be the last day or the last week data,
- intermediate-term model: using more data than the above, i.e. longer period than short-term, e.g. two time frame data, to capture not only newest but also older fraudulent strategies. In real life, it could be the last month or last year data,

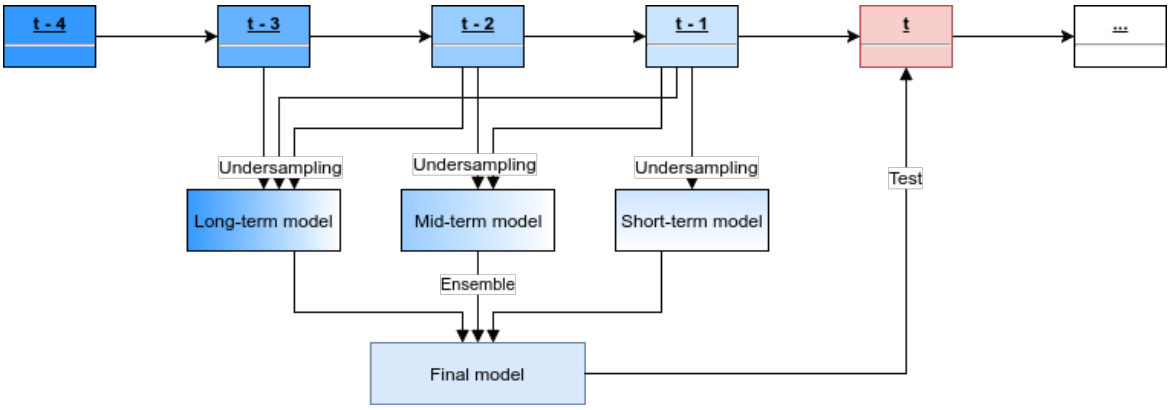


Fig. 3.3 Recommended Fraud Detection Pipeline

- long-term model: similar with the above models but using more or all of the available data, e.g. three time frame data, to keep all possible fraudulent strategies in our data.

The pipeline is summarized in figure 3.3 and the detail of each model have been described above (see figure 3.2). The size of the data and its training time increase from short-term to long-term model to accumulate fraudulent patterns. The long-term model needs a long time to prepare its new version and could be not available for the next day data, but with the fastest model, we could detect new fraudulent patterns while the new long-term model is training. This architecture could reduce workload for the slowest model and makes sure the system is always prepared for any possible situation.

3.11 Upgrade the Fraud Detection Pipeline

Since our proposed Fraud Detection Pipeline is a simple approach, there are several things could be done to improve the system and here we summarize some of the most potential improvements that will upgrade the current system easily:

- Imbalanced dataset: there are two main ideas to resolve this problem, one is the sampling methods and the other is cost-based methods. Our pipeline using the simpler way that does not need to know about the cost but it is very promising for the business view. We suggest trying to use the cost-based methods to have an overview of loss or saving money while using the system,

- Algorithm: the supervised Random Forest algorithm had chosen as the main classifiers in the pipeline. We suggest expanding the models in the pipeline that could use more data or use other Machine Learning algorithms, e.g. the unsupervised learning to detect

anomaly transactions or use stronger algorithms like Neural Networks, then finally combine all of classifiers to make the final prediction,

- Feature Engineering: there are some profiling methods to capture customer's behaviors as we have described in section 3.4. While the original attributes in the credit card dataset is limited, these aggregated features are very useful for the system and we strongly propose to apply all of these methods as the simplest way to upgrade the pipeline,
- Measurement: the F1 score is the most suitable choice for the imbalance problem, therefore we do not need to change the metric. However, if we have the cost of each transaction, with or without the cost-based methods for both imbalance problem and algorithms, we suggest having one cost-based measure to see how much the money the system could save for a company,
- Fraud Network: finding a network of fraudulent transactions in the data could improve the accuracy significantly. However, this task is not easy and requires complex researches so we suggest the readers only give a try with it after resolving all of the other things,
- Delayed True Labels: the delayed labels are very important since it is the up-to-date and accurate labels. We suggest extending the Algorithms section above with one model using this data and with wrong predictions of our models, it could not only predicts the true delayed labels but also explains why our prediction fails,
- Performance: the FDS usually need to process a very large number of transactions. While the distributed-computing Spark framework could handle it and also fits with other industry's requirements, therefore we do not have any reason to find an alternative solution and we suggest using the Spark framework in our system.

3.12 Experiments and results

Credit card transactions are very sensitive data, therefore there are only a few public datasets on the internet, some of them are very obsolete and do not have too many transactions. Fortunately there is one new and large transaction dataset which is published by Pozzolo et al. [27] in 2016, this dataset contains transactions made in September 2013 by European cardholders. It presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. This dataset is highly unbalanced, the positive class (fraudulent transactions) is 0.172% of all transactions, i.e. imbalance ratio is 577.

Table 3.3 F_1 results of the Fraud Detection Pipeline

Window length	Update mechanism		Training F_1		Testing F_1	
	Strategy	Time frame (on testing set)	Average	Std	Average	Std
1	Never update	0	0.4758	0.3480	0.3331	0.3242
	Daily update	24	0.6951	0.2312	**0.6024**	0.2808
2	Never update	0	0.7506	0.1632	0.72	0.2192
	Daily update	24	0.7325	0.2304	**0.722**	0.1893
3	Never update	0	0.7808	0.1511	**0.7351**	0.1665
	Daily update	24	0.7571	0.2267	0.7268	0.1871
Ensemble	Daily update	72	0.7554	0.2164	**0.7261**	0.1865

The dataset contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, they cannot provide the original features and more background information about the data. Features *V1*, *V2*,..., *V28* are the principal components obtained with PCA; the only features which have not been transformed with PCA are *Time* and *Amount*. The feature *Time* contains the seconds elapsed between each transaction and the first transaction in the dataset, the feature *Amount* is the transaction's amount. Column *Class* is the response variable and it takes value 1 in case of fraud and 0 otherwise.

In this thesis, we use this dataset to test the proposed Fraud Detection Pipeline and also for other studies below. Splitting this dataset into 48 time frames, we use the first 24 time frames for a training phase and the rest for the testing phase. Testing with three window lengths, short-term, intermediate-term and long-term periods are 1,2 and 3 time frames respectively. We run a stratified 5-fold cross-validation then compute the metric F_1 score as an average of five folds. This study runs on one single machine with 24 cores CPU and 128 gigabytes memory. The result is in figure 3.4 and a detail is in table 3.3.

In the result, the short-term model often fails with zero F_1 score if it does not re-train anymore and compare with daily update, the model will be improved significantly (81%). Only the long-term model with the daily update has a bit lower accuracy than the never update model (1%). As we forecast above, the more data we use in the longer term model the more accuracy we have, i.e. the long-term model with F_1 score is 22% higher than the short-term model. In general, the daily update mechanism usually better in both average and standard deviation of F_1 scores over 5-fold cross-validation.

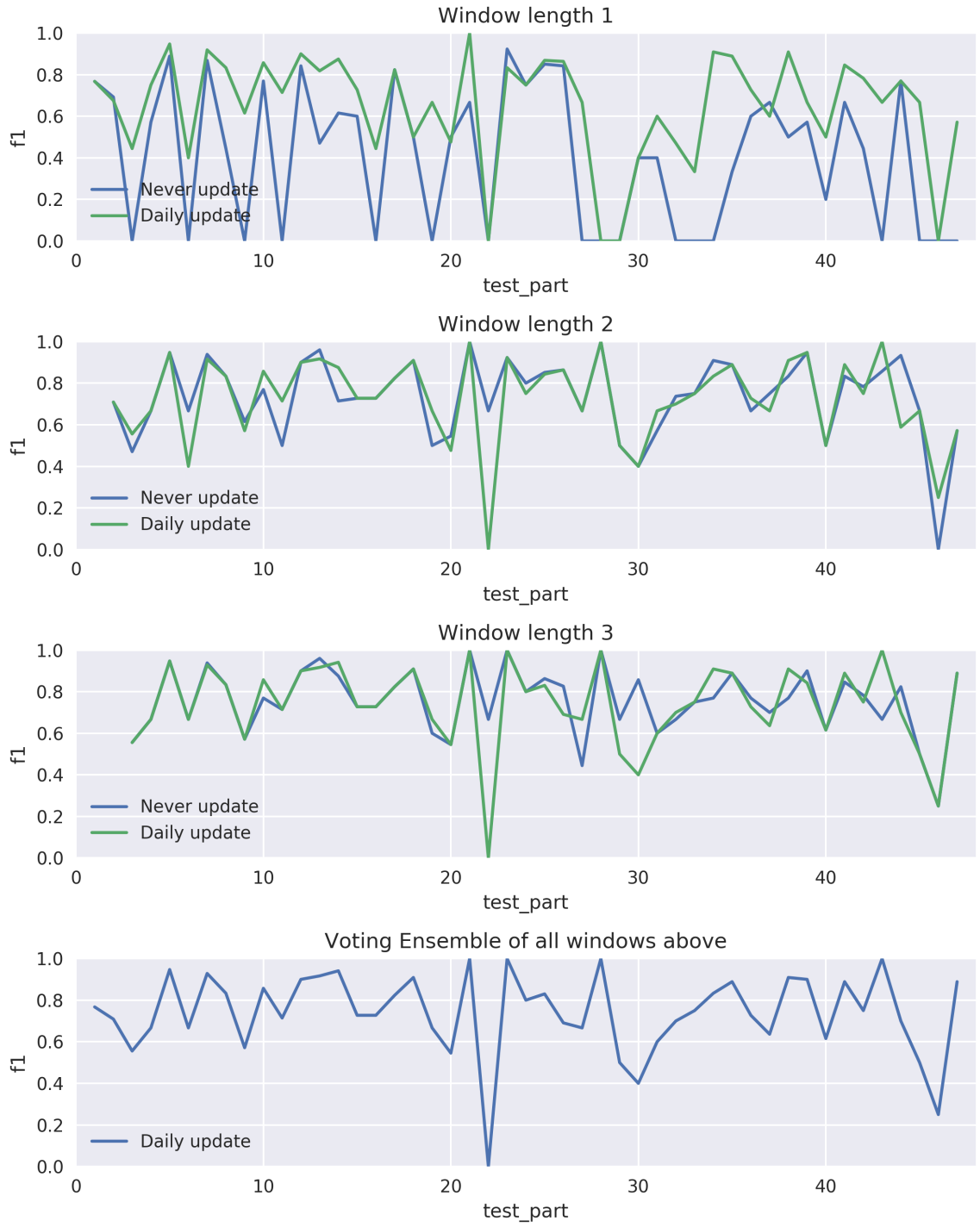


Fig. 3.4 Accuracy of our Fraud Detection Pipeline

3.13 Conclusion and other works

In this survey, we have discussed about several challenges in building an effective Fraud Detection System and then proposed one most suitable and also easy-to-use approach to build our Fraud Detection Pipeline. From the F_1 score in the results, it shows that our pipeline could handle the fraud detection problem effectively only with the simplified approach. We also suggest for the readers many things to do to upgrade our pipeline, and it leads to three independent works below:

- Concept Drift: we have assumed that we could update all of the models in our system every day; in case the data is very huge and the update process takes more than one day to complete then it is a costly task for our system. In chapter 4, we propose a method to decide which model needs to update with the most up-to-date data and we call it as Update Point Estimation algorithm,
- Imbalance problem: in our proposed pipeline, we used undersampling technique and ensemble learning to handle the imbalance problem (see figure 3.2) in the credit card dataset which is not only huge but also imbalanced extremely. In chapter 5, we show that why this combination is a most suitable in our case,
- Algorithms: we only use the Random Forest algorithm in our pipeline, in the rich of literature in fraud detection problem almost algorithms, e.g. supervised learning and unsupervised learning, are used. To the best of our knowledge, a graph-based semi-supervised learning techniques have not been applied to this problem. Therefore in chapter 6, we propose to use an un-normalized graph p-Laplacian based semi-supervised learning combined with the undersampling technique to the fraud detection problem.

Chapter 4

Update Point Estimation for the case of Concept-Drift in fraud detection problem

In the previous section, we proposed the Fraud Detection Pipeline and to avoid the Concept Drift problem, i.e. to keep its accuracy, it needs to update frequently, e.g. daily. In fact, it only needs to update to adapt to new patterns in the data. The Concept Drift is usually seen in streaming data and several methods have been proposed to deal with it, but in the fraud detection problem the data is often the batch data and these methods are not suitable for this case. We propose a method to detect when we should update our models and the result shows that our models not only need to update less frequently but also have the better performance.

4.1 Introduction

Imbalance is one common issue in many real domains, e.g. fraud detection, telecommunication. In those cases, we need to identify a small number of positive data points (minority class) stand among too many redundant data points. Consider a classification task of a dataset with imbalance ratio (IR) of 100, i.e. in every 101 samples there is only one positive sample that we need to detect. Most of algorithms in Machine Learning are not designed to handle this situation; if they maximize their accuracy then in the worst case they always have the accuracy of 99% by doing nothing. This lazy classifier marking all samples in a dataset as majority class has very high accuracy, but mis-classify all minority samples. Many studies have reported that they lose their performance with the imbalance datasets. [22, 23, 143, 69].

To handle the imbalance problem, there are many proposed methods and these could be grouped into two levels: algorithmic level and data level. At the algorithmic level, classifiers are designed or modified from existing algorithms to handle the imbalanced data by itself [12, 24]. At the data level, original imbalanced data will be modified by pre-processing step before applying to normal classification algorithms, e.g. under-sampling, over-sampling [33] or SMOTE [20]. Several studies [146, 86, 36] have shown that a balanced training set will give the better performance when using normal algorithms. Cost-sensitive approach [91] uses a similar idea with those above approaches to minimize misclassification costs, which are higher for data points in the minority class; however, it requires side information about the cost which is not always available.

In the age of computing world, the size of data quickly increases to a huge volume due to the advantages of computer technologies. Big dataset could be seen in many domains like genome biology, banking system. The size of the dataset could be million or billion records, which leads to the question that whether our solution effective with a massive amount of data. Particularly in an imbalanced big data classification [30, 134, 43], previous approaches could not provide solutions since they create a bigger dataset then cannot fit into our resources or has poor results. Especially, an extremely imbalanced data problem in a big dataset is another story that receives attention very recently [133, 81].

Several studies in imbalanced datasets have imbalance ratio less than 100, i.e. minority class greater than 1% of the data, and those approaches cannot guarantee the accuracy for the highly imbalanced tasks (IR $>$ 100). In some applications, the datasets are not only really huge but also highly imbalanced, e.g. fraud detection often has IR greater than 1000 [75]. From this scenario of two difficult problems, we need an effective way to tackle both of them at the same time and therefore, in this study we want to show a simplified combination of under-sampling technique and ensemble learning could have a good result in the extremely imbalanced big data classification.

The paper is organized as follows: section 2 mentions recent works relate to the extremely imbalanced data, imbalance big data classification and the gap between them. Section 3 presents our methodology, experimental design and comparative result on many datasets. Then we finally come to a conclusion and future works in section 4.

4.2 Recent works on extreme imbalance in big data classification

In the class imbalance problem, Mikel Galar at el. [49] gave a comprehensive review of several methodologies that have been proposed to deal with this problem. From their empirical comparison of the most significant published approaches, they have concluded that ensemble-based algorithms, e.g. using bagging or boosting, are worthwhile and under-sampling techniques, such as RUSBoost or UnderBagging, could have higher performances than many other complex approaches.

In the Big Data issue, techniques used to deal with it usually based on distributed computing on a system of computers. Similarly in order to handle the imbalance problem in the big dataset, Sara del Río at el. [30] implemented a distributed version of those common sampling techniques based on MapReduce framework [100] and they found that the under-sampling method could manage large datasets. Isaac Triguero at el. [133] faced with the extremely imbalanced big data in bioinformatics problem and their solution is a complex combination of Random Forest [14] and oversampling that balanced the distribution of classes. In a review of Alberto Fernández at el. [43] on the Big Data and Imbalanced classification, they considered different sampling ratio between classes and stated that a perfectly balanced sample was not the best method as traditional approaches that have been used [65, 51].

Recently, some researchers notice the extremely imbalance problem on datasets with imbalance ratio greater than 100 [126, 133]. This context easily leads to a classification task in the Big Data since the positive class is a very small fraction in the dataset, if the dataset is small then we cannot find any meaningful patterns on a few positive data points. To our best knowledge, there are a few studies on this gap of extremely imbalanced big data classification. For example, Sara del Río [30] tested their study on datasets with a largest size is over five million data points or a maximum IR is 74680. This new scenario raises a question about how effectiveness we could achieve with this kind of data. An ideal method need to use any given data points very effective, for both positive and negative data points.

In the traditional imbalance problem, many evaluation measures could not be used in this case independently since they are affected by the majority class. In order to evaluate the classifier, it requires a stronger metric that balance the classes and a commonly used metric for this case is F_1 score that can derive from confusion matrix (table 5.1). Based on two other metrics, Precision and Recall, the F_1 score is the harmonic average of them and could be computed as formula 5.3. In this study we propose to use the F_1 score as a main measure.

Table 4.1 Confusion matrix

	predicted positives	predicted negatives
real positives	True positive (TP)	False negative (FN)
real negatives	False positive (FP)	True negative (TN)

$$Precision = \frac{TP}{TP+FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.2}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.3}$$

4.3 Proposed analysis

Based on the question posed in the previous section, we propose another way to build samples that effectively use all of the data points in the dataset once time. Using the under-sampling method, all negative data points will be randomly divided into several dis-joint same-size subsets then each subset will be merged back with all positive data points then it is used to train a classifier. After that, we combine these classifiers by the voting ensemble as a final prediction. Suppose a dataset D has N data points with imbalance ratio is IR, let K be a number of subsets, our proposed approach is summarized by Algorithm 3, in which we define D^{major} and D^{minor} are the majority and minority classes in the dataset D respectively and $n = \left\lfloor \frac{\left|D^{major}\right|}{K} \right\rfloor$ is a size of a sampled subset. In the main part of Algorithm 3, in this study we mainly use Random Forest because of its robustness and good performance.

To illustrate the effectiveness of our proposed algorithm, we apply it on many datasets with various aspects, e.g. from small to large datasets, from low to very high imbalance ratio and moreover we try to do our test with several values of K, from 3, 5, 10 to 20. We run a stratified 5-fold cross-validation then compute the metric F_1 score as an average of five folds and the running time is also reported in seconds. This test is carried out in one single machine with 24 cores and 128 gigabytes in memory. All details of the final results are summarized in the table 5.2.

The results in bold indicate the best F_1 scores from the Under Bagging (UB) algorithm and our method K-Segments Under Bagging (K-SUB) with multiple K values. With datasets have low imbalance ratio (e.g. IR $<$ 50) the Under Bagging algorithm could handle the

Algorithm 2 K-Segments Under Bagging (K-SUB)

Inputs:
D^{major} is the majority class in the dataset D
D^{minor} is the minority class in the dataset D
K is the number of subsets
Procedure:

1: **for** $k = 1$ to K **do**:
2: Draw a subset K_k^{major} without replacement from the majority class with the size $n = \left\lceil \frac{|D^{major}|}{K} \right\rceil$
3: Let K_k is a merged subset of K_k^{major} with all data points in minority class D^{minor}
4: Train a classifier f^k by applying Random Forest algorithm to the subset K_k
5: **end for**
6: Combine all f_k into an aggregated model F
7: Return F

problem, and our method has also comparative results. However, from a high to extreme imbalance cases, i.e. IR $>= 50$, our method outperform the Under Bagging in all cases with any value of our chosen K values. Furthermore, the running time is dramatically slow for the case of Under Bagging algorithm, but with our method it only takes us few minutes in the same machine. Specially in the two biggest datasets with highest IR, the F_1 scores of the Under Bagging tend to become zero but our method still keeps the high F_1 scores. In the range of K values selected above, we observe the case $K = 20$ is not the best one, so it suggests that only a small enough value of K may be a good choice in our proposed analysis.

4.4 Conclusion

In this study, we have presented the K-Segments Under Bagging algorithm in the attempt to tackle the problem of extremely imbalanced data classification. The experimental results show that in the case of extremely imbalanced data, our method not only outperforms the previous method but also runs very fast. While in case of low imbalanced data, with the suitable K value, the K-SUB algorithm is still useful. Furthermore, it can be observed experimentally that we may obtain the good results for the small enough values of K, which suggests that in real applications we only have to tune the parameter K in a small range of value. With this new and challenging scenario, we have shown that the simplified combination of undersampling technique and ensemble learning is able to give better results instead of using other complicated methods addressed in the past. In the future work, we want to investigate deeper on this approach as well as other related issues.

Table 4.2 Summary of datasets and the experimental results

dataset	IR	sample	feature	UB		3-SUB		5-SUB		10-SUB		20-SUB	
				F1	time	F1	time	F1	time	F1	time	F1	time
ecoli	8	336	7	**0.5758**	1	0.5669	1	0.4850	2	0.4563	5	0.3953	9
spectrometer	10	531	93	0.7642	1	**0.8411**	1	0.8248	2	0.7933	4	0.6601	10
car_eval_34	11	1728	21	**0.7173**	2	0.4445	1	0.2418	2	0.2380	5	0.2081	9
isolet	11	7797	617	0.6084	5	**0.7702**	3	0.7326	4	0.5984	7	0.4710	13
libras_move	14	360	90	0.6797	2	**0.7389**	1	0.6473	2	0.4036	4	0.1717	9
thyroid_sick	15	3772	52	0.7551	2	0.8023	2	**0.8261**	3	0.8112	5	0.6708	10
oil	21	937	49	**0.3083**	3	0.0941	1	0.0829	2	0.0848	4	0.0841	9
car_eval_4	25	1728	21	**0.6078**	4	0.4413	1	0.2183	2	0.1623	5	0.1481	10
letter_img	26	20000	16	0.6133	5	**0.8681**	3	0.8381	4	0.8059	9	0.7018	15
webpage	34	344780	300	0.3783	14	0.1842	5	0.3130	6	**0.4496**	12	0.1616	18
mammography	42	11183	6	0.4033	5	0.6478	1	**0.6606**	1	0.6316	3	0.5221	5
protein_homo	111	145751	74	0.4877	24	0.7908	7	0.7953	7	**0.8125**	12	0.8113	15
10% kddcup R2L	443	494021	41	0.2396	643	0.8945	44	**0.9350**	46	0.8803	90	0.7176	101
fraud_detection	577	284807	29	0.2341	148	**0.8113**	30	0.6231	30	0.5567	55	0.3304	62
kdd SF PROBE	7988	703067	30	0.0126	3980	0.7895	80	0.8034	79	**0.8242**	156	0.7176	156
10% kdd U2R	9499	494021	41	0.0075	7008	**0.6034**	38	**0.6034**	39	0.5782	79	0.5291	90
kdd U2R	94199	4940219	41	0.0007	112285	0.3741	359	0.4571	343	**0.5617**	737	0.4916	780

Chapter 5

K-Segments Under Bagging approach: An experimental Study on Extremely Imbalanced Data Classification

Imbalanced dataset could be found in many real-world domains of applications, e.g. fraud detection problem, threat detection, etc.,. There are various methods that have been proposed to handle the imbalanced data classification problems, but there is no guarantee those methods will work well in the case of extremely imbalanced data. Practically, we can find the explosion of imbalance issue in the case of big dataset analysis, in which the imbalance ratio increases uncontrollably. We all know that Big Data is a terminology used to express the increasing level of both volume and complexity of the data, and the big data within extreme imbalance scenario is such a research question from recent work. In this study, we propose a simplified combination of under-sampling and ensemble learning which can adapt well with different scenarios of extreme imbalance. Experimentally, we carry out our test on 17 datasets, taken from the UCI repository and Kaggle, and can show that our proposed method is not only competitive with a common method but also very effective especially in the case of extremely imbalanced big data classification problems.

5.1 Introduction

Imbalance is one common issue in many real domains, e.g. fraud detection, telecommunication. In those cases, we need to identify a small number of positive data points (minority class) stand among too many redundant data points. Consider a classification task of a dataset with imbalance ratio (IR) of 100, i.e. in every 101 samples there is only one positive sample

that we need to detect. Most of algorithms in Machine Learning are not designed to handle this situation; if they maximize their accuracy then in the worst case they always have the accuracy of 99% by doing nothing. This lazy classifier marking all samples in a dataset as majority class has very high accuracy, but mis-classify all minority samples. Many studies have reported that they lose their performance with the imbalance datasets. [22, 23, 143, 69].

To handle the imbalance problem, there are many proposed methods and these could be grouped into two levels: algorithmic level and data level. At the algorithmic level, classifiers are designed or modified from existing algorithms to handle the imbalanced data by itself [12, 24]. At the data level, original imbalanced data will be modified by pre-processing step before applying to normal classification algorithms, e.g. under-sampling, over-sampling [33] or SMOTE [20]. Several studies [146, 86, 36] have shown that a balanced training set will give the better performance when using normal algorithms. Cost-sensitive approach [91] uses a similar idea with those above approaches to minimize misclassification costs, which are higher for data points in the minority class; however, it requires side information about the cost which is not always available.

In the age of computing world, the size of data quickly increases to a huge volume due to the advantages of computer technologies. Big dataset could be seen in many domains like genome biology, banking system. The size of the dataset could be million or billion records, which leads to the question that whether our solution effective with a massive amount of data. Particularly in an imbalanced big data classification [30, 134, 43], previous approaches could not provide solutions since they create a bigger dataset then cannot fit into our resources or has poor results. Especially, an extremely imbalanced data problem in a big dataset is another story that receives attention very recently [133, 81].

Several studies in imbalanced datasets have imbalance ratio less than 100, i.e. minority class greater than 1% of the data, and those approaches cannot guarantee the accuracy for the highly imbalanced tasks (IR $>$ 100). In some applications, the datasets are not only really huge but also highly imbalanced, e.g. fraud detection often has IR greater than 1000 [75]. From this scenario of two difficult problems, we need an effective way to tackle both of them at the same time and therefore, in this study we want to show a simplified combination of under-sampling technique and ensemble learning could have a good result in the extremely imbalanced big data classification.

The paper is organized as follows: section 2 mentions recent works relate to the extremely imbalanced data, imbalance big data classification and the gap between them. Section 3 presents our methodology, experimental design and comparative result on many datasets. Then we finally come to a conclusion and future works in section 4.

5.2 Recent works on extreme imbalance in big data classification

In the class imbalance problem, Mikel Galar at el. [49] gave a comprehensive review of several methodologies that have been proposed to deal with this problem. From their empirical comparison of the most significant published approaches, they have concluded that ensemble-based algorithms, e.g. using bagging or boosting, are worthwhile and under-sampling techniques, such as RUSBoost or UnderBagging, could have higher performances than many other complex approaches.

In the Big Data issue, techniques used to deal with it usually based on distributed computing on a system of computers. Similarly in order to handle the imbalance problem in the big dataset, Sara del Río at el. [30] implemented a distributed version of those common sampling techniques based on MapReduce framework [100] and they found that the under-sampling method could manage large datasets. Isaac Triguero at el. [133] faced with the extremely imbalanced big data in bioinformatics problem and their solution is a complex combination of Random Forest [14] and oversampling that balanced the distribution of classes. In a review of Alberto Fernández at el. [43] on the Big Data and Imbalanced classification, they considered different sampling ratio between classes and stated that a perfectly balanced sample was not the best method as traditional approaches that have been used [65, 51].

Recently, some researchers notice the extremely imbalance problem on datasets with imbalance ratio greater than 100 [126, 133]. This context easily leads to a classification task in the Big Data since the positive class is a very small fraction in the dataset, if the dataset is small then we cannot find any meaningful patterns on a few positive data points. To our best knowledge, there are a few studies on this gap of extremely imbalanced big data classification. For example, Sara del Río [30] tested their study on datasets with a largest size is over five million data points or a maximum IR is 74680. This new scenario raises a question about how effectiveness we could achieve with this kind of data. An ideal method need to use any given data points very effective, for both positive and negative data points.

In the traditional imbalance problem, many evaluation measures could not be used in this case independently since they are affected by the majority class. In order to evaluate the classifier, it requires a stronger metric that balance the classes and a commonly used metric for this case is F_1 score that can derive from confusion matrix (table 5.1). Based on two other metrics, Precision and Recall, the F_1 score is the harmonic average of them and could be computed as formula 5.3. In this study we propose to use the F_1 score as a main measure.

Table 5.1 Confusion matrix

	predicted positives	predicted negatives
real positives	True positive (TP)	False negative (FN)
real negatives	False positive (FP)	True negative (TN)

$$Precision = \frac{TP}{TP+FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP+FN} \tag{5.2}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.3}$$

5.3 Proposed analysis

Based on the question posed in the previous section, we propose another way to build samples that effectively use all of the data points in the dataset once time. Using the under-sampling method, all negative data points will be randomly divided into several dis-joint same-size subsets then each subset will be merged back with all positive data points then it is used to train a classifier. After that, we combine these classifiers by the voting ensemble as a final prediction. Suppose a dataset D has N data points with imbalance ratio is IR, let K be a number of subsets, our proposed approach is summarized by Algorithm 3, in which we define D^{major} and D^{minor} are the majority and minority classes in the dataset D respectively and $n = \left\lfloor \frac{\left|D^{major}\right|}{K} \right\rfloor$ is a size of a sampled subset. In the main part of Algorithm 3, in this study we mainly use Random Forest because of its robustness and good performance.

To illustrate the effectiveness of our proposed algorithm, we apply it on many datasets with various aspects, e.g. from small to large datasets, from low to very high imbalance ratio and moreover we try to do our test with several values of K, from 3, 5, 10 to 20. We run a stratified 5-fold cross-validation then compute the metric F_1 score as an average of five folds and the running time is also reported in seconds. This test is carried out in one single machine with 24 cores and 128 gigabytes in memory. All details of the final results are summarized in the table 5.2.

The results in bold indicate the best F_1 scores from the Under Bagging (UB) algorithm and our method K-Segments Under Bagging (K-SUB) with multiple K values. With datasets have low imbalance ratio (e.g. IR $<$ 50) the Under Bagging algorithm could handle the

Algorithm 3 K-Segments Under Bagging (K-SUB)

Inputs:
D^{major} is the majority class in the dataset D
D^{minor} is the minority class in the dataset D
K is the number of subsets
Procedure:

1: **for** $k = 1$ to K **do**:
2: Draw a subset K_k^{major} without replacement from the majority class with the size $n = \left\lceil \frac{|D^{major}|}{K} \right\rceil$
3: Let K_k is a merged subset of K_k^{major} with all data points in minority class D^{minor}
4: Train a classifier f^k by applying Random Forest algorithm to the subset K_k
5: **end for**
6: Combine all f_k into an aggregated model F
7: Return F

problem, and our method has also comparative results. However, from a high to extreme imbalance cases, i.e. IR $>= 50$, our method outperform the Under Bagging in all cases with any value of our chosen K values. Furthermore, the running time is dramatically slow for the case of Under Bagging algorithm, but with our method it only takes us few minutes in the same machine. Specially in the two biggest datasets with highest IR, the F_1 scores of the Under Bagging tend to become zero but our method still keeps the high F_1 scores. In the range of K values selected above, we observe the case $K = 20$ is not the best one, so it suggests that only a small enough value of K may be a good choice in our proposed analysis.

5.4 Conclusion

In this study, we have presented the K-Segments Under Bagging algorithm in the attempt to tackle the problem of extremely imbalanced data classification. The experimental results show that in the case of extremely imbalanced data, our method not only outperforms the previous method but also runs very fast. While in case of low imbalanced data, with the suitable K value, the K-SUB algorithm is still useful. Furthermore, it can be observed experimentally that we may obtain the good results for the small enough values of K, which suggests that in real applications we only have to tune the parameter K in a small range of value. With this new and challenging scenario, we have shown that the simplified combination of undersampling technique and ensemble learning is able to give better results instead of using other complicated methods addressed in the past. In the future work, we want to investigate deeper on this approach as well as other related issues.

Table 5.2 Summary of datasets and the experimental results

dataset	IR	sample	feature	UB		3-SUB		5-SUB		10-SUB		20-SUB	
				F1	time	F1	time	F1	time	F1	time	F1	time
ecoli	8	336	7	**0.5758**	1	0.5669	1	0.4850	2	0.4563	5	0.3953	9
spectrometer	10	531	93	0.7642	1	**0.8411**	1	0.8248	2	0.7933	4	0.6601	10
car_eval_34	11	1728	21	**0.7173**	2	0.4445	1	0.2418	2	0.2380	5	0.2081	9
isolet	11	7797	617	0.6084	5	**0.7702**	3	0.7326	4	0.5984	7	0.4710	13
libras_move	14	360	90	0.6797	2	**0.7389**	1	0.6473	2	0.4036	4	0.1717	9
thyroid_sick	15	3772	52	0.7551	2	0.8023	2	**0.8261**	3	0.8112	5	0.6708	10
oil	21	937	49	**0.3083**	3	0.0941	1	0.0829	2	0.0848	4	0.0841	9
car_eval_4	25	1728	21	**0.6078**	4	0.4413	1	0.2183	2	0.1623	5	0.1481	10
letter_img	26	20000	16	0.6133	5	**0.8681**	3	0.8381	4	0.8059	9	0.7018	15
webpage	34	344780	300	0.3783	14	0.1842	5	0.3130	6	**0.4496**	12	0.1616	18
mammography	42	11183	6	0.4033	5	0.6478	1	**0.6606**	1	0.6316	3	0.5221	5
protein_homo	111	145751	74	0.4877	24	0.7908	7	0.7953	7	**0.8125**	12	0.8113	15
10% kddcup R2L	443	494021	41	0.2396	643	0.8945	44	**0.9350**	46	0.8803	90	0.7176	101
fraud_detection	577	284807	29	0.2341	148	**0.8113**	30	0.6231	30	0.5567	55	0.3304	62
kdd SF PROBE	7988	703067	30	0.0126	3980	0.7895	80	0.8034	79	**0.8242**	156	0.7176	156
10% kdd U2R	9499	494021	41	0.0075	7008	**0.6034**	38	**0.6034**	39	0.5782	79	0.5291	90
kdd U2R	94199	4940219	41	0.0007	112285	0.3741	359	0.4571	343	**0.5617**	737	0.4916	780

Chapter 6

Solve fraud detection problem by using graph based learning methods

The credit cards' fraud transactions detection is the important problem in machine learning field. To detect the credit cards' fraud transactions help reduce the significant loss of the credit cards' holders and the banks. To detect the credit cards' fraud transactions, data scientists normally employ the un-supervised learning techniques and supervised learning technique. In this paper, we employ the graph p-Laplacian based semi-supervised learning methods combined with the under-sampling technique such as Cluster Centroids to solve the credit cards' fraud transactions detection problem. Experimental results show that that the graph p-Laplacian semi-supervised learning methods outperform the current state of art graph Laplacian based semi-supervised learning method ($p = 2$). In the scope of this thesis, some necessary knowledge will not be presented and the reader may find them in [130, 85].

6.1 Introduction

While purchasing online, the transactions can be done by using credit cards that are issued by the bank. In this case, if the cards or cards' details are stolen, the fraud transactions can be easily carried out. This will lead to the significant loss of the cardholder or the bank. In order to detect credit cards' fraud transactions, data scientists employ a lot of machine learning techniques. To the best of our knowledge, there are two classes of machine learning techniques used to detect credit cards' fraud transactions which are un-supervised learning techniques and supervised learning techniques. The un-supervised learning techniques used to detect credit cards' fraud transactions are k-means clustering technique [53], k-nearest neighbors technique [53], Local Outlier Factor technique [53], to name a few. The supervised

learning techniques used to detect credit cards' fraud transactions are Hidden Markov Model technique [124], neural network technique [107], Support Vector Machine technique [120], to name a few.

To the best of our knowledge, the graph based semi-supervised learning techniques [123] have not been applied to the credit cards' fraud transactions detection problem. In this paper, we will apply the un-normalized graph p-Laplacian based semi-supervised learning technique [132, 131] combined with the under-sampling technique to the credit cards' fraud transactions detection problem.

We will organize the paper as follows: Section 2 will introduce the preliminary notations and definitions used in this paper. Section 3 will introduce the definitions of the gradient and divergence operators of graphs. Section 4 will introduce the definition of Laplace operator of graphs and its properties. Section 5 will introduce the definition of the curvature operator of graphs and its properties. Section 6 will introduce the definition of the p-Laplace operator of graphs and its properties. Section 7 will show how to derive the algorithm of the un-normalized graph p-Laplacian based semi-supervised learning method from regularization framework. In section 8, we will compare the accuracy performance measures of the un-normalized graph Laplacian based semi-supervised learning algorithm (i.e. the current state of art graph based semi-supervised learning method) combined with the under-sampling technique such as Cluster Centroids technique [156] and the un-normalized graph p-Laplacian based semi-supervised learning algorithms combined with Cluster Centroids technique [156]. Section 9 will conclude this paper and the future direction of researches.

6.2 Preliminary notations and definitions

Given a graph $G = (V, E, W)$ where V is a set of vertices with $|V| = n$, $E \subseteq V * V$ is a set of edges and W is a $n * n$ similarity matrix with elements $w_{ij} \geq 0 (1 \leq i, j \leq n)$.

Also, please note that $w_{ij} = w_{ji}$.

The degree function $d : V \to R^+$ is

$$d_i = \sum_{j \sim i} w_{ij} \tag{6.1}$$

where $j \sim i$ is the set of vertices adjacent with i.

Define $D = diag(d_1, d_2, \ldots, d_n)$.

The inner product on the function space R^V is

$$< f, g >_V = \sum_{i \in V} f_i g_i \tag{6.2}$$

Also, define an inner product on the space of functions R^E on the edges

$$< F, G >_E = \sum_{(i,j)\in E} F_{ij} G_{ij} \tag{6.3}$$

Here let $H(V) = \left(R^V, < ., . >_V\right)$ and $H(E) = \left(R^E, < ., . >_E\right)$ be the Hilbert space real-valued functions defined on the vertices of the graph G and the Hilbert space of real-valued functions defined in the edges of G respectively.

6.3 Gradient and Divergence Operators

We define the gradient operator $d : H(V) \to H(E)$ to be

$$(df)_{ij} = \sqrt{w_{ij}} \left(f_j - f_i\right) \tag{6.4}$$

where $f : V \to R$ be a function of $H(V)$.

We define the divergence operator $div : H(E) \to H(V)$ to be

$$< df, F >_{H(E)} = < f, -divF >_{H(V)}, \tag{6.5}$$

where $f \in H(V), F \in H(E)$.

Thus, we have

$$(divF)_j = \sum_{i\sim j} \sqrt{w_{ij}} \left(F_{ji} - F_{ij}\right) \tag{6.6}$$

6.4 Laplace operator

We define the Laplace operator $\Delta : H(V) \to H(V)$ to be

$$\Delta f = -\frac{1}{2} div(df) \tag{6.7}$$

Thus, we have

$$(\Delta f)_j = d_j f_j - \sum_{i\sim j} w_{ij} f_i \tag{6.8}$$

The graph Laplacian is a linear operator. Furthermore, the graph Laplacian is self-adjoint and positive semi-definite.

Let $S_2(f) = < \Delta f, f >$, we have the following **theorem 1**

$$D_f S_2 = 2\Delta f \tag{6.9}$$

The proof of the above theorem can be found from [132, 131].

6.5 Curvature operator

We define the curvature operator $\kappa : H(V) \to H(V)$ to be

$$\kappa f = -\frac{1}{2} div\left(\frac{df}{||df||}\right) \tag{6.10}$$

Thus, we have

$$(\kappa f)_j = \frac{1}{2}\sum_{i\sim j} w_{ij}\left(\frac{1}{\|d_i f\|} + \frac{1}{\|d_j f\|}\right)(f_j - f_i) \tag{6.11}$$

From the above formula, we have

$$d_i f = \left((df)_{ij} : j \sim i\right)^T \tag{6.12}$$

The local variation of f at i is defined to be

$$\|d_i f\| = \sqrt{\sum_{j\sim i}(df)_{ij}^2} = \sqrt{\sum_{j\sim i} w_{ij}\left(f_j - f_i\right)^2} \tag{6.13}$$

To avoid the zero denominators in 6.11, the local variation of f at i is defined to be

$$\|d_i f\| = \sqrt{\sum_{j\sim i}(df)_{ij}^2 + \varepsilon} \tag{6.14}$$

where $\varepsilon = 10^{-10}$.
The graph curvature is a non-linear operator.
Let $S_1(f) = \sum_i \|d_i f\|$, we have the following **theorem 2**

$$D_f S_1 = \kappa f \tag{6.15}$$

The proof of the above theorem can be found from [132, 131].

6.6 p-Laplace operator

We define the p-Laplace operator $\Delta_p : H(V) \to H(V)$ to be

$$\Delta_p f = -\frac{1}{2} div\left(\|df\|^{p-2} df\right) \tag{6.16}$$

Thus, we have

$$(\Delta_p f)_j = \frac{1}{2}\sum_{i\sim j} w_{ij}\left(\|d_i f\|^{p-2} + \|d_j f\|^{p-2}\right)(f_j - f_i) \tag{6.17}$$

Let $S_p(f) = \frac{1}{p}\sum_i \|d_i f\|^p$, we have the following **theorem 3**

$$D_f S_p = p\Delta_p f \tag{6.18}$$

6.7 Discrete regularization on graphs and credit cards' fraud transactions detection problems

Given a transaction network *G=(V,E)*. *V* is the set of all transactions in the network and *E* is the set of all possible interactions between these transactions. Let *y* denote the initial function in *H(V)*. y_i can be defined as follows

$$y_i = \begin{cases} 1 \text{ if transaction } i \text{ is the fraud transaction} \\ -1 \text{ if transaction } i \text{ is the normal transaction} \\ 0 \text{ otherwise} \end{cases}$$

Our goal is to look for an estimated function f in *H(V)* such that f is not only smooth on G but also close enough to an initial function y. Then each transaction i is classified as $sign(f_i)$. This concept can be formulated as the following optimization problem

$$argmin_{f\in H(V)}\{S_p(f) + \frac{\mu}{2}\|f-y\|^2\} \tag{6.19}$$

The first term in 6.19 is the smoothness term. The second term is the fitting term. A positive parameter μ captures the trade-off between these two competing terms.

6.7.1 p-smoothness

For any number p, the optimization problem 6.19 is

$$argmin_{f\in H(V)}\{\frac{1}{p}\sum_i \|d_i f\|^p + \frac{\mu}{2}\|f-y\|^2\} \tag{6.20}$$

By theorem 3, we have

Theorem 4: The solution of 6.20 satisfies

$$\Delta_p f + \mu (f - y) = 0 \tag{6.21}$$

The *p-Laplace* operator is a non-linear operator; hence we do not have the closed form solution of equation 6.21. Thus, we have to construct iterative algorithm to obtain the solution. From 6.21, we have

$$\frac{1}{2}\sum_{i\sim j} w_{ij}\left(\|d_i f\|^{p-2} + \|d_j f\|^{p-2}\right)\left(f_j - f_i\right) + \mu\left(f_j - y_j\right) = 0 \tag{6.22}$$

Define the function $m : E \to R$ by

$$m_{ij} = \frac{1}{2} w_{ij}\left(\|d_i f\|^{p-2} + \|d_j f\|^{p-2}\right) \tag{6.23}$$

Then equation 6.22 which is

$$\sum_{i\sim j} m_{ij}\left(f_j - f_i\right) + \mu\left(f_j - y_j\right) = 0$$

can be transformed into

$$\left(\sum_{i\sim j} m_{ij} + \mu\right) f_j = \sum_{i\sim j} m_{ij} f_i + \mu y_j \tag{6.24}$$

Define the function $p : E \to R$ by

$$p_{ij} = \begin{cases} \frac{m_{ij}}{\sum_{i\sim j} m_{ij} + \mu} \text{ if } i \neq j \\ \frac{\mu}{\sum_{i\sim j} m_{ij} + \mu} \text{ if } i = j \end{cases} \tag{6.25}$$

Then

$$f_j = \sum_{i\sim j} p_{ij} f_i + p_{jj} y_j \tag{6.26}$$

Thus we can consider the iteration

$$f_j^{(t+1)} = \sum_{i\sim j} p_{ij}^{(t)} f_i^{(t)} + p_{jj}^{(t)} y_j \text{ for all } j \in V$$

to obtain the solution of 6.20.

6.8 Experiments and results

Datasets

In this paper, we use the transaction dataset available from [27]. This dataset contains 284,807 transactions. Each transaction has 30 features. In the other words, we are given transaction data matrix ($R^{284807*30}$) and the annotation (i.e. the label) matrix ($R^{284807*1}$). The ratio between the number of fraud transactions and the number of normal transactions is 0.00173. Hence we easily recognize that this is the imbalanced classification problem. In order to solve this imbalanced classification problem, we initially apply the under-sampling technique which is the Cluster Centroid technique [156] to this imbalanced dataset. Then we have that the ratio between the number of fraud transactions and the number of normal transactions is 0.4. In the other words, we are given the **new transaction data** matrix ($R^{1722*30}$) and the annotation (i.e. the label) matrix (R^{1722*1}).

Then we construct the similarity graph from the transaction data. The similarity graph used in this paper is the k-nearest neighbor graph: Transaction i is connected with transaction j if transaction i is among the k-nearest neighbor of transaction j or transaction j is among the k-nearest neighbor of transaction i.

In this paper, the similarity function is the Gaussian similarity function

$$s\big(T(i,:),T(j,:)\big) = exp\bigg(-\frac{d\big(T(i,:),T(j,:)\big)}{t}\bigg)$$

In this paper, t is set to 0.1 and the 5-nearest neighbor graph is used to construct the similarity graph from the **new transaction data**.

Experimental Results

In this section, we experiment with the above proposed un-normalized graph p-Laplacian methods with *p=1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9* and the current state of the art method (i.e. the un-normalized graph Laplacian based semi-supervised learning method *p=2*) in terms of classification accuracy performance measure. The accuracy performance measure Q is given as follows

$$Q = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

The **new transaction data** is divided into two subsets: the training set and the testing set. The training set contains 1,208 transactions. The testing set contains 514 transactions. The parameter μ is set to 1.

The accuracy performance measures of the above proposed methods and the current state of the art method is given in the following table 1

Table 6.1 The comparison of accuracies of proposed methods with different p-values

Accuracy Performance Measures (%)	
p=1	88.52
p=1.1	88.52
p=1.2	88.52
p=1.3	88.52
p=1.4	88.52
p=1.5	88.52
p=1.6	88.52
p=1.7	88.52
p=1.8	88.52
p=1.9	88.52
p=2	88.33

From the above table, we easily recognized that the un-normalized graph p-Laplacian semi-supervised learning methods outperform the current state of art method. The results from the above table show that the un-normalized graph p-Laplacian semi-supervised learning methods are at least as good as the current state of the art method (p=*2*) but often lead to better classification accuracy performance measures.

6.9 Conclusions

We have developed the detailed regularization frameworks for the un-normalized graph p-Laplacian semi-supervised learning methods applying to the credit cards' fraud transactions detection problem. Experiments show that the un-normalized graph p-Laplacian semi-supervised learning methods are at least as good as the current state of the art method (i.e. p=*2*) but often lead to significant better classification accuracy performance measures.

In the future, we will develop the detailed regularization frameworks for the un-normalized hypergraph p-Laplacian semi-supervised learning methods and will apply these methods to this credit cards' fraud transactions detection problem.

Chapter 7

Conclusion and Future works

Fraud detection is a challenging and complex problem in many real-world applications, particularly credit card fraud detection problem. This thesis investigates on how to use Machine Learning and Data Science to address some of the issues in this problem. This chapter summarizes the main results of this thesis, discusses open issues and presents our future work.

Chapter 3 is the survey on various challenges of the fraud detection problem, with each challenge we also presented some most suitable solutions. After they are analyzed, we have proposed the most simple and effective strategy to build the Fraud Detection Pipeline that could be used as the backbone of one fraud detection system. Based on our Fraud Detection Pipeline, we also suggest many ways to extend or upgrade it. With this comprehensive survey, we want to write more details and clearly to contribute to the community the most recent works in the fraud detection problem.

Using our Fraud Detection Pipeline, which requires to update all models in the system every time frame (e.g daily), in chapter 4 we presented the mechanism to predict the improvement if we update a model, then with the pre-defined threshold we could decide which model need to update. The result shows that our system reduces the number of update times significantly, i.e 80%. In future work, we want to replace the pre-defined threshold with a dynamic threshold that can make our system more stable and also not-sensitive with our parameter.

In the Fraud Detection Pipeline, we have proposed to use undersampling technique and ensemble learning for the credit card fraud detection dataset, which is not only huge but also very high imbalanced. In chapter 5, we presented the study of this combination on the case of extremely imbalanced big data classification and the result shows that our approach is very effective and promising if the dataset is bigger or more imbalance. In future work, we

want to investigate deeper on this new gap of two difficult problems, e.g feature selection on the extremely imbalanced big data classification.

Most of the Machine Learning algorithms are applied to the fraud detection problem, but the graph-based learning has not been considered. In chapter 6, we used the graph p-Laplacian based semi-supervised learning with undersampling on this problem and the result shows that it outperforms the current state of the art graph Laplacian based semi-supervised method. Finding the relationship of the fraud network is one of the hardest tasks in the fraud detection problem and the graph-based learning attracts more attention recently. In future work, we want to apply the graph-based learning to detect the fraud networks in our data.

References

[1] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM.

[2] AICPA (2017). American institute of cpas.

[3] Artís, M., Ayuso, M., and Guillén, M. (2002). Detection of automobile insurance fraud with discrete choice models and misclassified claims. *Journal of Risk and Insurance*, 69(3):325–340.

[4] Bahnsen, A. C., Aouada, D., and Ottersten, B. (2015). Example-dependent cost-sensitive decision trees. *Expert Systems with Applications*, 42(19):6609–6619.

[5] Bahnsen, A. C., Aouada, D., Stojanovic, A., and Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems With Applications*, 51:134–142.

[6] Bahnsen, A. C., Stojanovic, A., Aouada, D., and Ottersten, B. (2013). Cost sensitive credit card fraud detection using bayes minimum risk. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 1, pages 333–338. IEEE.

[7] Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1):20–29.

[8] Bhattacharyya, S., Jha, S., Tharakunnel, K., and Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602–613.

[9] BiPM, I., IFCC, I., ISO, I., and IUPAP, O. (2008). International vocabulary of metrology–basic and general concepts and associated terms, 2008. *JCGM*, 200:99–12.

[10] Bollinger, C. R. and David, M. H. (1997). Modeling discrete choice with response error: Food stamp participation. *Journal of the American Statistical Association*, 92(439):827–835.

[11] Bolton, R. J., Hand, D. J., et al. (2001). Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, pages 235–255.

[12] Bradford, J. P., Kunz, C., Kohavi, R., Brunk, C., and Brodley, C. E. (1998). Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, pages 131–136. Springer.

[13] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

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

[15] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

[16] Cerullo, M. J. and Cerullo, V. (1999). Using neural networks to predict financial reporting fraud: Part 1. *Computer Fraud & Security*, 1999(5):14–17.

[17] Chan, P. K., Fan, W., Prodromidis, A. L., and Stolfo, S. J. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications*, 14(6):67–74.

[18] Chawla, N., Lazarevic, A., Hall, L., and Bowyer, K. (2003). Smoteboost: Improving prediction of the minority class in boosting. *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119.

[19] Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer.

[20] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[21] Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6.

[22] Chen, C., Liaw, A., and Breiman, L. (2004). Using random forest to learn imbalanced data. *University of California, Berkeley*, 110:1–12.

[23] Chen, X.-w., Gerlach, B., and Casasent, D. (2005). Pruning support vectors for imbalanced data classification. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1883–1888. IEEE.

[24] Cieslak, D. A. and Chawla, N. V. (2008). Learning decision trees for unbalanced data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 241–256. Springer.

[25] Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242.

[26] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G. (2015a). Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE.

[27] Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G. (2015b). Calibrating probability with undersampling for unbalanced classification. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 159–166. IEEE.

[28] Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., and Bontempi, G. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10):4915–4928.

[29] Dal Pozzolo, A., Caelen, O., Waterschoot, S., and Bontempi, G. (2013). Racing for unbalanced methods selection. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 24–31. Springer.

[30] Del Río, S., López, V., Benítez, J. M., and Herrera, F. (2014). On the use of mapreduce for imbalanced big data using random forest. *Information Sciences*, 285:112–137.

[31] Delamaire, L., Abdou, H., and Pointon, J. (2009). Credit card fraud and detection techniques: a review. *Banks and Bank systems*, 4(2):57–68.

[32] Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM.

[33] Drummond, C., Holte, R. C., et al. (2003). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11. Citeseer Washington DC.

[34] Dudík, M., Phillips, S. J., and Schapire, R. E. (2006). Correcting sample selection bias in maximum entropy density estimation. In *Advances in neural information processing systems*, pages 323–330.

[35] Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd.

[36] Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.

[37] Fan, W. (2004). Systematic data selection to mine concept-drifting data streams. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 128–137. ACM.

[38] Fan, W. and Davidson, I. (2007). On sample selection bias and its efficient correction via model averaging and unlabeled examples. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 320–331. SIAM.

[39] Fan, W., Davidson, I., Zadrozny, B., and Yu, P. S. (2005). An improved categorization of classifier's sensitivity on sample selection bias. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE.

[40] Fanning, K. M. and Cogger, K. O. (1998). Neural network detection of management fraud using published financial data. *Intelligent Systems in Accounting, Finance & Management*, 7(1):21–41.

[41] Fast, A., Friedland, L., Maier, M., Taylor, B., Jensen, D., Goldberg, H. G., and Komoroske, J. (2007). Relational data pre-processing techniques for improved securities fraud detection. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 941–949. ACM.

[42] Fawcett, T. and Provost, F. (1997). Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316.

[43] Fernández, A., del Río, S., Chawla, N. V., and Herrera, F. (2017). An insight into imbalanced big data classification: outcomes and challenges. *Complex & Intelligent Systems*, 3(2):105–120.

[44] Fisher, N. I. (1995). *Statistical analysis of circular data*. Cambridge University Press.

[45] Foundation, T. A. S. (2017). Spark 2.2.0.

[46] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

[47] Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156.

[48] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

[49] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

[50] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44.

[51] García, S. and Herrera, F. (2009). Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary computation*, 17(3):275–306.

[52] Ghosh, S. and Reilly, D. L. (1994). Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621–630. IEEE.

[53] Goldstein, M. and Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173.

[54] Green, B. P. and Choi, J. H. (1997). Assessing the risk of management fraud through neural network technology. *Auditing*, 16(1):14.

[55] Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, 1(1):17–61.

[56] Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. *Advances in intelligent computing*, pages 878–887.

[57] Hanczar, B., Hua, J., Sima, C., Weinstein, J., Bittner, M., and Dougherty, E. R. (2010). Small-sample precision of roc-related estimates. *Bioinformatics*, 26(6):822–830.

[58] Hand, D. J. (2009). Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning*, 77(1):103–123.

[59] Hanley, J. A. and McNeil, B. J. (1983). A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843.

[60] HaratiNik, M. R., Akrami, M., Khadivi, S., and Shajari, M. (2012). Fuzzgy: A hybrid model for credit card fraud detection. In *Telecommunications (IST), 2012 Sixth International Symposium on*, pages 1088–1093. IEEE.

[61] Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516.

[62] Hausman, J. A., Abrevaya, J., and Scott-Morton, F. M. (1998). Misclassification of the dependent variable in a discrete-response setting. *Journal of econometrics*, 87(2):239–269.

[63] He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE.

[64] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

[65] Hido, S., Kashima, H., and Takahashi, Y. (2009). Roughly balanced bagging for imbalanced data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2(5-6):412–426.

[66] Ho, T. K. (1995). Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE.

[67] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844.

[68] Hoens, T. R., Polikar, R., and Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101.

[69] Hong, X., Chen, S., and Harris, C. J. (2007). A kernel-based two-class classifier for imbalanced data sets. *IEEE Transactions on neural networks*, 18(1):28–41.

[70] Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

[71] Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.

[72] Jha, S., Guillen, M., and Westland, J. C. (2012). Employing transaction aggregation strategy to detect credit card fraud. *Expert systems with applications*, 39(16):12650–12657.

[73] Jin, Y., Rejesus*, R. M., and Little, B. B. (2005). Binary choice models for rare events data: a crop insurance fraud application. *Applied Economics*, 37(7):841–848.

[74] Joshi, M. V., Kumar, V., and Agarwal, R. C. (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 257–264. IEEE.

[75] Juszczak, P., Adams, N. M., Hand, D. J., Whitrow, C., and Weston, D. J. (2008). Off-the-peg and bespoke classifiers for fraud detection. *Computational Statistics & Data Analysis*, 52(9):4521–4532.

[76] Kang, P. and Cho, S. (2006). Eus svms: Ensemble of under-sampled svms for data imbalance problems. In *Neural Information Processing*, pages 837–846. Springer.

[77] Kelly, M. G., Hand, D. J., and Adams, N. M. (1999). The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 367–371. ACM.

[78] Kirkos, E., Spathis, C., and Manolopoulos, Y. (2007). Data mining techniques for the detection of fraudulent financial statements. *Expert systems with applications*, 32(4):995–1003.

[79] Klement, W., Flach, P., Japkowicz, N., and Matwin, S. (2009). Cost-based sampling of individual instances. In *Canadian Conference on Artificial Intelligence*, pages 86–97. Springer.

[80] Koza, J. R., Bennett III, F. H., Andre, D., and Keane, M. A. (1996). Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In *Artificial Intelligence in Design'96*, pages 151–170. Springer.

[81] Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232.

[82] Krivko, M. (2010). A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications*, 37(8):6070–6076.

[83] Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pages 231–238.

[84] Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186. Nashville, USA.

[85] Latouche, P. and Rossi, F. (2015). Graphs in machine learning: an introduction. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Proceedings of the 23-th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2015)*, pages 207–218.

[86] Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. *Artificial Intelligence in Medicine*, pages 63–66.

[87] Lebbe, M. A., Agbinya, J. I., Chaczko, Z., and Braun, R. (2008). Artificial immune system inspired danger modelling in wireless mesh networks. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 984–988. IEEE.

[88] Lee, T.-S., Chiu, C.-C., Chou, Y.-C., and Lu, C.-J. (2006). Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. *Computational Statistics & Data Analysis*, 50(4):1113–1130.

[89] Leonard, K. J. (1993). Detecting credit card fraud using expert systems. *Computers & industrial engineering*, 25(1-4):103–106.

[90] Leonard, K. J. (1995). The development of a rule based expert system model for fraud alert in consumer credit. *European journal of operational research*, 80(2):350–356.

[91] Ling, C. X., Yang, Q., Wang, J., and Zhang, S. (2004). Decision trees with minimal costs. In *Proceedings of the twenty-first international conference on Machine learning*, page 69. ACM.

[92] Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550.

[93] Liu, Y., An, A., and Huang, X. (2006). Boosting prediction accuracy on imbalanced datasets with svm ensembles. In *PAKDD*, volume 6, pages 107–118. Springer.

[94] Lobo, J. M., Jiménez-Valverde, A., and Real, R. (2008). Auc: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography*, 17(2):145–151.

[95] Mahmoudi, N. and Duman, E. (2015). Detecting credit card fraud by modified fisher discriminant analysis. *Expert Systems with Applications*, 42(5):2510–2516.

[96] Maloof, M., Langley, P., Sage, S., and Binford, T. (1997). Learning to detect rooftops in aerial images. In *Proceedings of the Image Understanding Workshop*, pages 835–845.

[97] Mani, I. and Zhang, I. (2003). knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126.

[98] Mease, D., Wyner, A. J., and Buja, A. (2007). Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8(Mar):409–439.

[99] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

[100] Miner, D. and Shook, A. (2012). *MapReduce design patterns: building effective algorithms and analytics for Hadoop and other systems*. " O'Reilly Media, Inc.".

[101] Minister, T. and General, A. (1985). Criminal code (r.s.c., 1985, c. c-46).

[102] Mitchell, T. M. et al. (1997). Machine learning. wcb.

[103] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

[104] Moreno-Torres, J. G., Raeder, T., Alaiz-RodríGuez, R., Chawla, N. V., and Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530.

[105] Nexis, L. (2014). True cost of fraud 2014 study.

[106] Nexis, L. (2016). True cost of fraud 2016 study.

[107] Nune, G. K. and Sena, P. V. (2015). Novel artificial neural networks and logistic approach for detecting credit card deceit. *International Journal of Computer Science and Network Security (IJCSNS)*, 15(9):21.

[108] Olson, D. L. and Delen, D. (2008). *Advanced data mining techniques*. Springer Science & Business Media.

[109] Opitz, D. W. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *J. Artif. Intell. Res.(JAIR)*, 11:169–198.

[110] Perry, J. W., Kent, A., and Berry, M. M. (1955). Machine literature searching x. machine language; factors underlying its design and development. *Journal of the Association for Information Science and Technology*, 6(4):242–254.

[111] Poterba, J. M. and Summers, L. H. (1995). Unemployment benefits and labor market transitions: A multinomial logit model with errors in classification. *The Review of Economics and Statistics*, pages 207–216.

[112] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

[113] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

[114] Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press.

[115] Raschka, S. (2015). *Python machine learning*. Packt Publishing Ltd.

[116] Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

[117] Rokach, L. and Maimon, O. (2014). *Data mining with decision trees: theory and applications*. World scientific.

[118] Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27.

[119] Sahin, Y., Bulkan, S., and Duman, E. (2013). A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923.

[120] Şahin, Y. G. and Duman, E. (2011). Detecting credit card fraud by decision trees and support vector machines.

[121] Samuel, A. L. (2000). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 44(1.2):206–226.

[122] Sánchez, D., Vila, M., Cerda, L., and Serrano, J.-M. (2009). Association rules applied to credit card fraud detection. *Expert systems with applications*, 36(2):3630–3640.

[123] Shin, H., Lisewski, A. M., and Lichtarge, O. (2007). Graph sharpening plus graph integration: a synergy that improves protein functional classification. *Bioinformatics*, 23(23):3217–3224.

[124] Singh, A., Narayan, D., et al. (2012). A survey on hidden markov model for credit card fraud detection. *International Journal of Engineering and Advanced Technology (IJEAT)*, 1(3).

[125] Suman, S., Laddhad, K., and Deshmukh, U. (2005). Methods for handling highly skewed datasets. *Part I-October*, 3.

[126] Tang, Y., Zhang, Y.-Q., Chawla, N. V., and Krasser, S. (2009). Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288.

[127] Tasoulis, D., Adams, N., Weston, D., and Hand, D. (2008). Mining information from plastic card transaction streams. In *Proceedings in Computational Statistics: 18th Symposium (COMPSTAT 2008)*, volume 2, pages 315–322.

[128] Tasoulis, D. K., Adams, N. M., and Hand, D. J. (2006). Unsupervised clustering in streaming data. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 638–642. IEEE.

[129] Tomek, I. (1976). Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, 6:769–772.

[130] Tran, L. (2012). Application of three graph laplacian based semi-supervised learning methods to protein function prediction problem. *arXiv preprint arXiv:1211.4289*.

[131] Tran, L. and Tran, L. (2017). The un-normalized graph p-laplacian based semi-supervised learning method and speech recognition problem. *International Journal of Advances in Soft Computing & Its Applications*, 9(1).

[132] Tran, L. H. and Tran, L. H. (2015). Un-normalized graph p-laplacian semi-supervised learning method applied to cancer classification problem. *Journal of Automation and Control Engineering Vol*, 3(1).

[133] Triguero, I., del Río, S., López, V., Bacardit, J., Benítez, J. M., and Herrera, F. (2015a). Rosefw-rf: the winner algorithm for the ecbdl'14 big data competition: an extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems*, 87:69–79.

[134] Triguero, I., Galar, M., Vluymans, S., Cornelis, C., Bustince, H., Herrera, F., and Saeys, Y. (2015b). Evolutionary undersampling for imbalanced big data classification. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 715–722. IEEE.

[135] Van Rijsbergen, C. (1979). Information retrieval. dept. of computer science, university of glasgow. *URL: citeseer. ist. psu. edu/vanrijsbergen79information. html*, 14.

[136] Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., and Baesens, B. (2015). Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48.

[137] Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., and Baesens, B. (2016). Gotcha! network-based fraud detection for social security fraud. *Management Science*.

[138] Vilariño, F., Spyridonos, P., Vitrià, J., and Radeva, P. (2005). Experiments with svm and stratified sampling with an imbalanced problem: detection of intestinal contractions. *Pattern Recognition and Image Analysis*, pages 783–791.

[139] Visa, S. and Ralescu, A. (2005). Issues in mining imbalanced data sets-a review paper. In *Proceedings of the sixteen midwest artificial intelligence and cognitive science conference*, volume 2005, pages 67–73. sn.

[140] Walker, S. H. and Duncan, D. B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179.

[141] Wang, B. and Japkowicz, N. (2004). Imbalanced data set learning with synthetic samples. In *Proc. IRIS Machine Learning Workshop*, volume 19.

[142] Wang, B. X. and Japkowicz, N. (2010). Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, 25(1):1–20.

[143] Wang, J., Xu, M., Wang, H., and Zhang, J. (2006). Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *Signal Processing, 2006 8th International Conference on*, volume 3. IEEE.

[144] Wang, S., Tang, K., and Yao, X. (2009). Diversity exploration and negative correlation learning on imbalanced data sets. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 3259–3266. IEEE.

[145] Wedge, R., Kanter, J. M., Rubio, S. M., Perez, S. I., and Veeramachaneni, K. (2017). Solving the" false positives" problem in fraud prediction. *arXiv preprint arXiv:1710.07709*.

[146] Weiss, G. M. and Provost, F. (2001). The effect of class distribution on classifier learning: an empirical study. *Rutgers Univ*.

[147] Weston, D. J., Hand, D. J., Adams, N. M., Whitrow, C., and Juszczak, P. (2008). Plastic card fraud detection using peer group analysis. *Advances in Data Analysis and Classification*, 2(1):45–62.

[148] Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., and Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55.

[149] Wikipedia (2017). Decision tree learning.

[150] Wikipedia (2018a). Precision and recall.

[151] Wikipedia (2018b). Receiver operating characteristic.

[152] Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421.

[153] Wu, G. and Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, pages 49–56.

[154] Yan, R., Liu, Y., Jin, R., and Hauptmann, A. (2003). On predicting rare classes with svm ensembles in scene classification. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 3, pages III–21. IEEE.

[155] Yeh, I.-C. and Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480.

[156] Yen, S.-J. and Lee, Y.-S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3):5718–5727.

[157] Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM.

[158] Zadrozny, B. and Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 204–213. ACM.

[159] Zadrozny, B., Langford, J., and Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 435–442. IEEE.

[160] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95.