Fraud Detection Pipeline

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 August 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
August 2018

Acknowledgements

I would like to thanks Dr. An Mai for his supervision and fruitful discussions on several research topics. A great thanks also to Loc Tran who helps me to 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

Billions of dollars of loss are caused every year by fraudulent credit card transactions, according to a review by the UK anti-fraud charity Fraud Advisory Panel (FAP) business fraud accounted for £144 billion, while fraud against individuals was estimated at £9.7 billion. The industries most commonly affected are banking, insurance, manufacturing, and government. The design of efficient fraud detection system is key for reducing their losses, however building this system is really challenging due to many hard problems such as imbalance dataset, ... Over last three decades, there are several researches relate to fraud detection but have no agreement on what is a best fraud detection system. In this thesis, we aim to answer a question that how could we build the Fraud Detection System and some related issues.

Table of contents

List of figures **xiii**

List of tables **xv**

1 Introduction **1**

1.1 Credit Card Fraud 1

1.1.1 Definition and impact of Credit Card Fraud 1

1.1.2 Credit Card Fraud Detection 3

1.2 Contributions 4

1.3 Outline 5

2 Preliminaries **7**

2.1 Machine Learning 7

2.2 Measurement 17

3 Fraud Detection Pipeline **21**

3.1 Imbalanced dataset 21

3.2 Algorithms 25

3.3 Sample Selection Bias 31

3.4 Feature Engineering 32

3.5 Measurement 35

3.6 Fraud Network 36

3.7 Concept Drift 37

3.8 Delayed True Labels 38

3.9 Performance 38

3.10 Fraud Detection Pipeline 39

3.11 Upgrade the Fraud Detection Pipeline 42

3.12 Experiments and results 44

3.13 Conclusion and other works 44

4 Update Point Estimation for the case of Concept-Drift in Fraud Detection Problem 47
4.1 Introduction 47
4.2 Related works 48
4.3 Methodology 50
4.4 Experiment and Conclusion 52

5 An Experimental Study on Undersampling and Ensemble for Extremely Imbalanced Big Data Classification 55
5.1 Introduction 55
5.2 A new problem of extreme imbalance in big data classification 57
5.3 Experimental analysis 58
5.4 Conclusion 60

6 Solve fraud detection problem by using graph based learning methods 61
6.1 Introduction 61
6.2 Preliminary notations and definitions 62
6.3 Gradient and Divergence Operators 63
6.4 Laplace operator 63
6.5 Curvature operator 64
6.6 p-Laplace operator 64
6.7 Discrete regularization on graphs and credit cards' fraud transactions detection problems 65
6.7.1 p-smoothness 65
6.8 Experiments and results 67
6.9 Conclusions 68

7 Conclusion and Future works 69

References 71

Appendix A How to install LaTeX 85

Appendix B Installing the CUED class file 89

List of figures

1.1 Cost of Fraud as a % of Revenues Keeps Going Up [140] 2
1.2 Total Costs per Dollar of Fraud Losses [140] 3

2.1 A tree showing survival of passengers on the Titanic 12
2.2 Bagging ensemble. 15
2.3 Boosting ensemble. 16
2.4 Stacking ensemble. 17
2.5 Precision and recall 18
2.6 ROC curve of three predictors of peptide cleaving in the proteasome 20

3.1 Three common sampling methods 24
3.2 Building single model by combining multiple Random Forest classifiers (RF) 40
3.3 Recommended Fraud Detection Pipeline 42
3.4 Accuracy of our Fraud Detection Pipeline 45

4.1 Illustration of four types in Concept Drift 49
4.2 Result of concept drift detection 54

List of tables

3.1 A simple cost matrix for one transaction. 23
3.2 List of typical raw attributes in a transaction dataset 33
3.3 F_1 results of the Fraud Detection Pipeline 46

4.1 The experimental results of Update Point Estimation 52

5.1 Confusion matrix 58
5.2 Summary of datasets and the experimental results 59

6.1 The comparison of accuracies of proposed methods with different p-values . 68

Chapter 1

Introduction

1.1 Credit Card Fraud

1.1.1 Definition and impact of Credit Card Fraud

A Criminal Code [134] 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

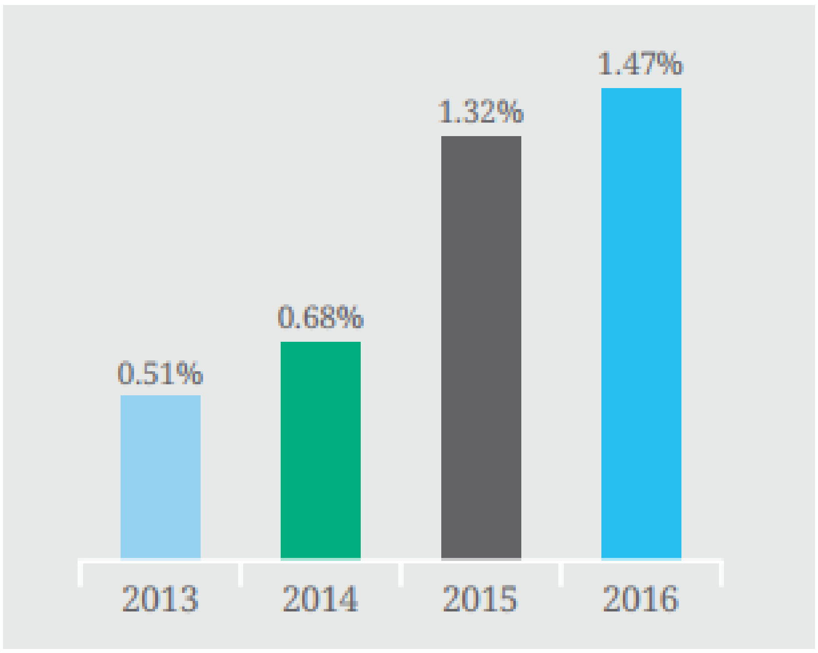


Fig. 1.1 Cost of Fraud as a % of Revenues Keeps Going Up [140]

is not easy to see the impact of fraud since companies and banks do not like to disclose a number of losses due to frauds. However, as stated in the Lexis Nexis's study [139], in 2014 fraudulent card transactions worldwide have reached around $11 billion a year. A latest study of Lexis Nexis [140] 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 a estimation of the Association for Payment Clearing Services (APACS) [41].

There are many types of fraud, but in this thesis we are focusing 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, such as for mail-order transactions by mail or fax, or over the 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 holder notifies the issuing bank and the bank puts a block on the account. Most banks have free 24-hour telephone numbers to encourage prompt reporting. Still,

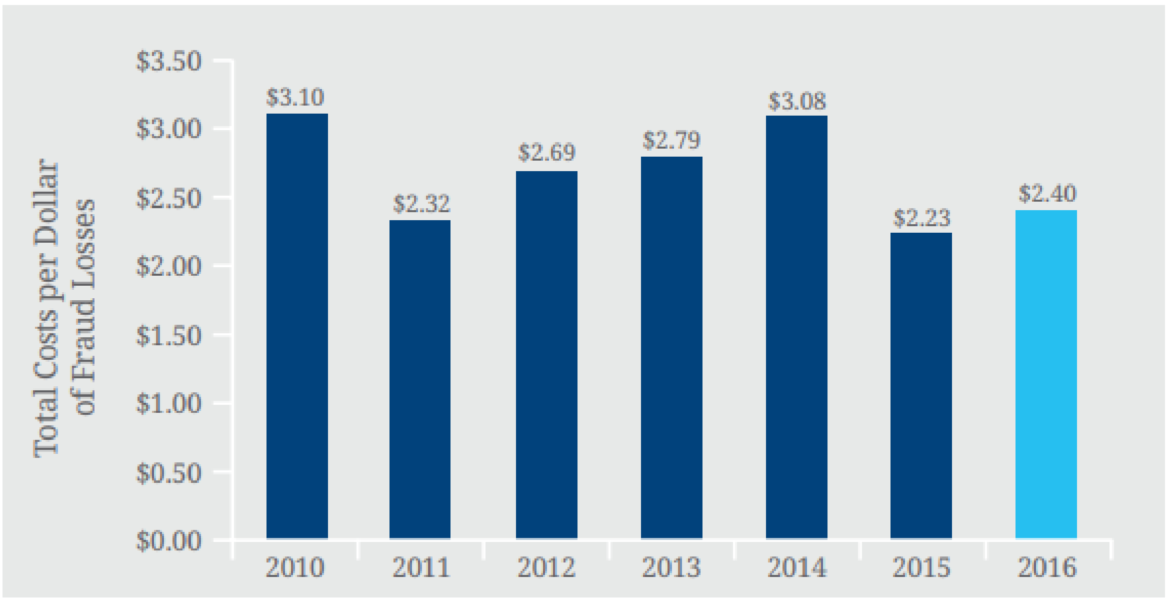


Fig. 1.2 Total Costs per Dollar of Fraud Losses [140]

it is possible for a thief to make unauthorized purchases on a card before the card is canceled. The detection of such a fraud typically relies on the discovery of an unexpected usage pattern of the credit card (generally unexpectedly important) with respect to the common practice.

1.1.2 Credit Card Fraud Detection

Detecting credit card fraud receives a lot of attention from industry (bank, insurance, . . .) and from research community since 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 require large human resource and time-consuming investigations to deal with different domains of knowledge like financial, economics, business practices and law. 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:

- Number of creadit card frauds is really small, it's just a small fraction of all the daily transactions. Many algorithms in Machine Learning have not been designed to deal with this problem,
- Many fraud transactions have a little different with normal transactions, it makes Machine Learning algorithms hard to detect a these transactions are fraud or not,
- Over time we see new attack strategies and changes in customer behavior, it makes distribution of data change over time and it doesn't meet a basic assumption in Machine Learning (and also Statistics),
- ....

1.2 Contributions

Focusing on the credit card fraud detection problem, in this report we present our works, including:

Fraud Detection Pipeline

Despite of rich literatures for fraud detection, there is no agreement on what is a best way to solve this problem and also there is no public fraud detection system. It means that in the industry, when a company want to build their own a fraud detection system, they can't know what is a good way to do, especially for those companies in Vietnam. A first main contribution of this report 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 themself easily, which we call it's a Fraud Detection Pipeline. Furthermore, we give readers many possible ways to extend the system.

Update Point Detection

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 have to update its models frequently to keep its accuracy, for example it could be daily or longer period. With credit card fraud detection problem, a common choice is daily when we receive whole data in a day then update the models at night. However, update models daily is worthless and actually it's only necessary if we don't adapt with new fraudulent patterns. In another words, we should update models if and only if it make improvement on our system, a second contribution is a mechanism that helps us make decision that when should we update our models.

An experimental study on undersampling and ensemble for extremely imbalanced big 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 another problem that concerns about volume and complexity of the data, the big data within extremely imbalance is a different question from those original issues and it has received attention very recently. In this study, we show that a simplified combination of under-sampling and ensemble learning is very effective in this case.

Solve fraud detection problem by using graph based learning methods

To detect the credit cards' fraud transactions, data scientists normally employ the un-supervised learning techniques and supervised learning technique. Among several algorithms have been proposed for this problem, a graph-based semi-supervised learning have not been applied to the credit cards' fraudulent transactions detection. We propose to apply a 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

The report is organized as follows: Chapter 2 provides the reader preliminary knowledge about Machine Learning, e.g a problem of classification, Machine Learning algorithms or common measurements, $\cdots$. Chapter 3 reviews several challenges of the Fraud Detection System and our proposed Fraud Detection Pipeline. Chapter 4 presents a mechanism to detect which model in our pipeline need to update. Chapter 5 explains more details of our chosen techniques in the proposed pipeline. Chapter 6 applies a graph-based learning into the fraud detection problem. And finally, Chapter 7 summarize 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 [159, 108]. Machine learning is strongly associated with computational statistics, which explores data and builds algorithms that could find patterns from 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“ [135].

Typically, tasks in machine learning can be grouped into three wide categories [156]:

- 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 is no desired outputs in the dataset, it means that we give the learning algorithms freely to explore a hidden struture 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

Supervised learning in the machine learning can be seen as a function from labeled training data [136]. Given a set of N data points of a form $(x_1, y_1), \ldots, (x_N, y_N)$ 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 [183, 35], or f could takes a form of a join t probability model $f(x,y) = P(x,y)$, e.g naive Bayes [156].

In order to measure a performance of learning function, we 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:

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

In order to choose the function f or g, there are two basic approaches:

- Empirical risk minimization: finds the learning function that best represent the output values from the given input data, it means that the function g could minimizes $R(g)$. Therefore, the supervised learning problem could be solved by applying an optimization algorithm to find the function g,

- Structural risk minimization: use a regularization penalty in the optimization to controls the bias/variance tradeoff. The supervised learning optimization problem becomes finding the function g that minimizes $J(g) = \widehat{R}(g) + \lambda C(g)$.

Linear Regression

Regression or regression analysis in supervised learning is processes for finding the relationships between input data points and their desired output values, it usually used for prediction and forecasting. A regression model involves following parameters and variables:

- Unknown parameters β: the parameter of the regression model which is usually represented as a vector,

- The independent variable X: the given data points which are represented as a matrix,

- The dependent variable Y: the desired output values from the above data points, it's usually represented as a matrix with only one column.

Then a regression model is a function f of variable X and parameter β relates to Y:

$$Y \approx f(X,\beta),$$

The function f represents the relationship of data and its output values, it means that f is built on characteristics of the given data so each kind of regression model consists some assumptions for its input data. There are some classical assumptions for regression analysis, include:

- The sample data is representative of the population,
- The error of the regression model is a random variable with a mean of zero and a variance is homologous across data points,
- Attributes in the data are linearly independent, it means that there is no attribute could be expressed as a linear combinations of the others.

A simplest regression model is Linear regression (LR) and its more generalize is generalized linear model (GLM) which has the dependent variable y_i is a linear combination of the parameters:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \varepsilon_i,$$

where:
$i = 1, \ldots, n$: i-th observation in the dataset,
k: number of attributes,
ε: the residual which is an error term between the true value and our estimation $\varepsilon_i = y_i - \widehat{y_i}$ with $\widehat{y_i} = \widehat{\beta_0} + \widehat{\beta_1} x_{i1} + \cdots + \widehat{\beta_k} x_{ik}$.
Using the matrix operators, we can rewrite the above equation as:

$$y_i = x_i^\top \beta + \varepsilon_i,$$

Stacking n data points together and written in vector form as:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon,$$

where:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix},$$

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}.$$

Therefore, we could estimate the output values $\widehat{\mathbf{y}}$ as:

$$\widehat{\mathbf{y}} = \mathbf{X}\widehat{\beta},$$

And the residual, $\varepsilon = \mathbf{y} - \widehat{\mathbf{y}}$, represents the different the true values $\mathbf{y}$ and the prediction of the model. One of methods measures the performance of model is a sum of squared residuals (SSE), sometimes also denoted RSS (Residual sum of squares):

$$SSE = \sum_{i=1}^{n} \varepsilon_i^2,$$

In order to estimate the parameter $\widehat{\beta}$, ordinary least squares (OLS) is a simplest and thus most common estimator which is used in this case. The OLS method will minimizes the SSE, and leads to a closed-form expression for the estimated value of the parameter β:

$$\widehat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top y.$$

Logistic Regression

Classification in machine learning and statistics is a problem of detecting to which of categories an instance belongs, an example would be identifying a given email is spam or not. Classification tasks could be grouped into two separate problems, binary classification and multiclass classification, there are only two classes in binary classification, whereas multiclass classification involves assigning an observation to one of several classes [77].

There are a large number of algorithms for classification using a linear function that assigns a score to each possible category, the predicted category will be the one with the highest score. This kind of scoring function is known as a linear predictor function which is similar with the regression formula above, it has a following general form:

$$score(X_i, k) = \beta_k \cdot X_i,$$

where:

X_i is the attribute vector of observation i,

β_k is the vector of weights respect to category k,

$score(X_i, k)$ is the score assigned to category k of observation i.

There are several examples of linear classifiers, e.g Logistic Regression [183, 35], Multinomial logistic regression [131] or Linear discriminant analysis [59]. Another classification algorithms include Support vector machines [34], Neural networks, ...

In the case of binary classification $k = 2$, a most well-known classifier is the Logistic Regression which is used to estimate a probability of a binary dependent variables, usually 0 and 1, by using a logistic function. The logistic function is the cummlative logistic distribution, it makes the Logistic Regression can be seen as a special case of the generalized linear model.

For a k-dimension of observation i-th $x_i \in X$ with the desired output value is $y_i \in 0, 1$, the Logistic Regression model is:

$$p_i \equiv p_i(\beta) \equiv \mathbb{P}(y_i = 1 | X = x_i) = \frac{e^{\beta_0 + \Sigma_{j=1}^{k} \beta_j x_{ij}}}{1 + e^{\beta_0 + \Sigma_{j=1}^{k} \beta_j x_{ij}}},$$

or equivalently:

$$logit(p_i) = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij},$$

where:

$$logit(p) = \log\left(\frac{p}{1-p}\right),$$

The value y_i is binary, therefore the data are Bernoulli:

$$y_i \mid x_i \sim Bernoulli(p_i),$$

It turns out that the conditional likelihood function is:

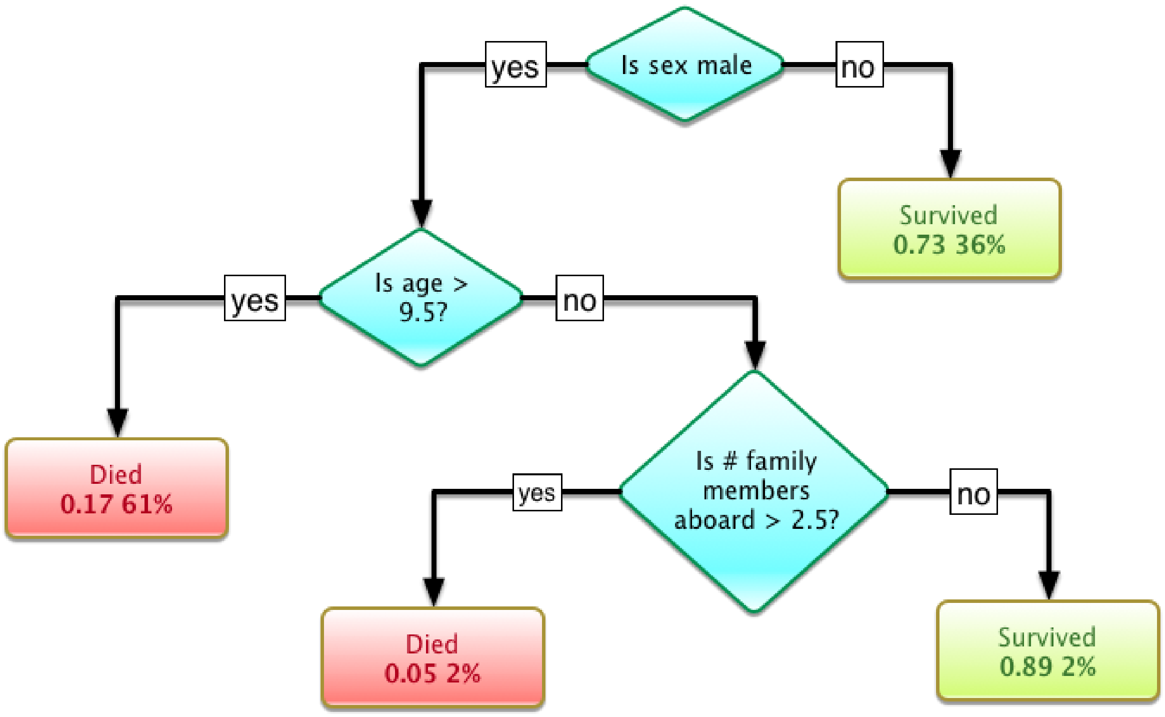


Fig. 2.1 A tree showing survival of passengers on the Titanic

$$\mathscr{L}(\beta) = \prod_{i=1}^{n} p_i(\beta)^{y_i} \left(1 - p_i(\beta)\right)^{1-y_i}.$$

We could use maximum likelihood estimation (MLE) to estimates $\widehat{\beta}$ by maximizing numerically the $\mathscr{L}(\beta)$.

Decision tree learning

Differently with the linear models above, a decision tree learning is another common approach in data mining [154], it aims to create a model that predicts values like making decision from a tree-based analysis. An example is shown in the figure 2.1 [195].

A tree learner could classifies the data points by splitting the input dataset into subsets based on an 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 desired output as a real number,
- Classification tree: an analysis aims to predict a categorical output.

A term Classification and Regression Tree (CART) analysis is used to refer to both the above analysis [20]. There are many decision tree algorithms, including:

- ID3 (Iterative Dichotomiser 3) [148],
- C4.5: an extension of ID3 algorithm [149].

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

Entropy The Entropy $H(S)$ is a measure of the amount of uncertainty in the dataset S, which has a following formula:

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x),$$

where:
S is the dataset or subset which is used to compute the entropy,
X is a set of classes in S,
$p(x)$ is a proportion of the number of data points in class x to the number of data points in S.

When $H(S) = 0$, the set S is perfectly classified, it means that all of data points in S are of the same class then we could stop splitting a current node.

Information Gain Information Gain $IG(A)$ is the measure of the amount of uncertainty in S was reduced after splitting set S on feature A, it is represented by a following formula:

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t),$$

where:
$H(S)$ is an entropy of set S,
T is the subsets which are created from splitting set S by feature A, hence $S = \bigcup_{t \in T} t$,
$p(t)$ is a proportion of the number of data points in class t to the number of data points in S,
$H(t)$ is an entropy of set t.

Gini impurity Gini impurity is a measure which is used by the CART algorithm, it measure how often a randomly chosen data point from the given set would be incorrectly classified if it was randomly classified according to the distribution of classes in the subset. Considering the dataset with J classes, with $i \in 1,2,\ldots,J$ and let p_i be the fraction of data points classified with class i in the set, the Gini impurity can be computed by a following formula:

$$I_G(p) = \sum_{i=1}^{J} p_i(1-p_i) = \sum_{i=1}^{J}(p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2 = \sum_{i \neq k} p_i p_k.$$

Random Forest

Random Forest [19] or random decision forests [84, 85] 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 bases on two elements:

1. Base learner: each tree in the forest is the base learner, typically it is a simple and weak learner,
2. Ensemble learning: combine results of several learners to make a final prediction.

The Decision Tree above that are grown very deep tend to overfit their training sets, i.e have 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 predictions via ensemble learning.

Ensemble learning In machine learning, ensemble learning is a method combines multiple learning algorithms to obtain better accuracy than from any individual algorithms [143, 153]. The term ensemble is usually reserved for methods that generate multiple hypotheses using the same base learner. There are many common types of ensemble learning, in this section we will describe some most common ensembles in data mining.

Bayes optimal classifier The Bayes Optimal Classifier not only is a classification technique but also is an ensemble of all the hypothesis in the hypothesis space. Each hypothesis has a vote proportional to the likelihood that the given dataset would be sampled from a data source if that hypothesis was true. The Bayes Optimal Classifier could be expresses as a following equation:

$$y = \underset{c_j \in C}{argmax} \sum_{h_i \in H} P(c_j \mid h_i) P(T \mid h_i) P(h_i),$$

where:

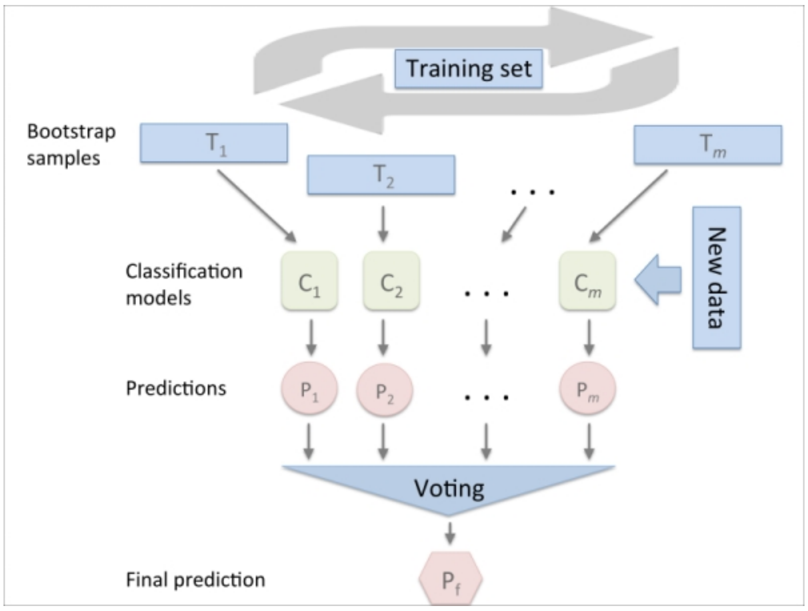


Fig. 2.2 Bagging ensemble.

y is the predicted class,

C is the set of all possible classes,

H is the hypothesis space,

P is the probability symbol,

T is the training dataset.

There are many issues with Bayes Optimal Classifier that makes it cannot be used in practical problems, such as:

- Most interesting hypothesis spaces are too large to iterate over, as required by the argmax,
- Estimating the prior probability for each hypothesis ($P(h_i)$ is rarely feasible.

Bootstrap aggregating (bagging) One of the common ensemble techniques 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 use random decision trees as base learners.

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 has a better accuracy than bagging in some cases, however it

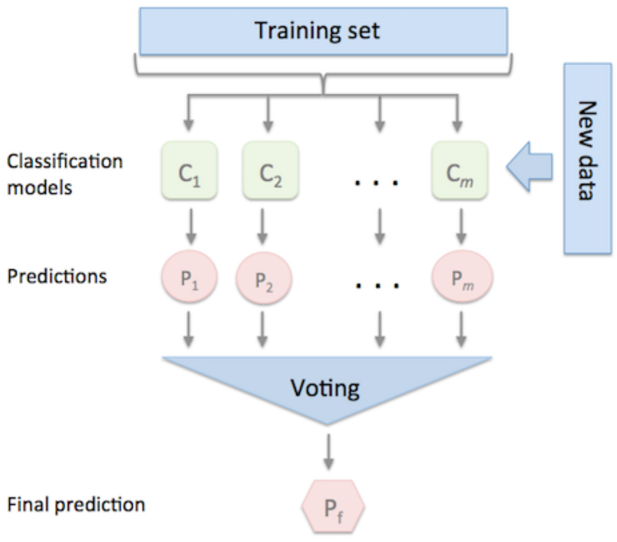


Fig. 2.3 Boosting ensemble.

also tends to overfit to the training data. A most common boosting algorithm is Adaboost [61].

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.

Back to the Random Forest algorithm, we can formulate the Random Forest as a following statement: given a dataset $X = x_1, \dots, x_n$ with n data points and respectively desired output values $Y = y_1, \dots, y_n$, 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
```

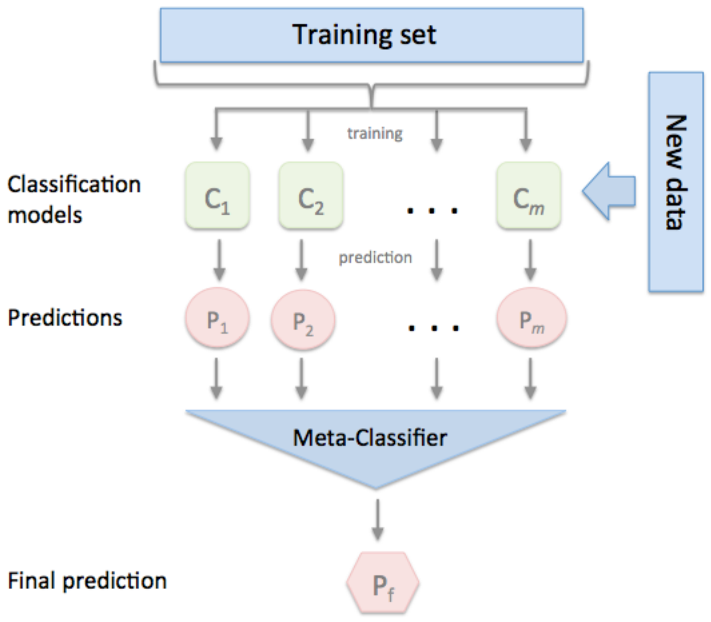


Fig. 2.4 Stacking ensemble.

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 [19]. However, there is no standard function as a combiner, thus we are free to choose the combination function. This boostrapping 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:

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

Random Forest use the above bagging algorithm with only one difference, it use 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 [63].

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 evalute the algorithms. In this section, we will review some common metrics which are typically used in classifcation tasks.

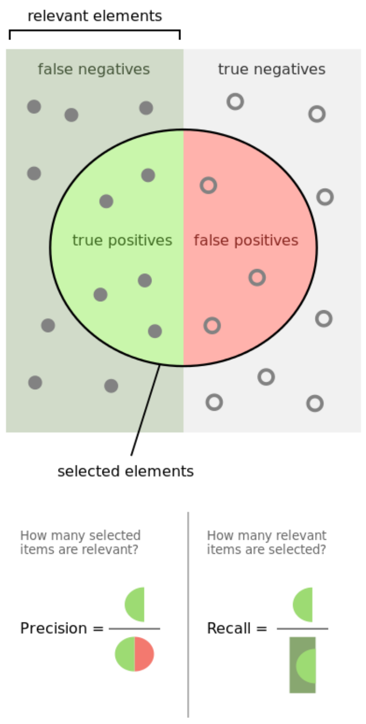


Fig. 2.5 Precision and recall

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 [14]. Consider 2 sets with n data points: the true labels from the given dataset $Y = y_1, \dots, y_n$ and our predicted values $\hat{Y} = \hat{y}_1, \dots, \hat{y}_n$, the accuracy is:

$$accuracy = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{y_i = \hat{y}_i}.$$

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) [145] 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:

$$\text{precision} = \frac{\mid \text{relevant elements} \cap \text{retrieved elements} \mid}{\mid \text{retrieved elements} \mid},$$

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

$$\text{precision} = \frac{\mid \text{relevant elements} \cap \text{retrieved elements} \mid}{\mid \text{relevant elements} \mid},$$

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 [142]:

$$\text{precision} = \frac{TP}{TP+FP},$$

$$\text{recall} = \frac{TP}{TP+FN},$$

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. ROC analysis provides tools to select 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). Since the AUC is a portion of the area of the unit square, its value will always be between 0 and 1. However, because random guessing produces the diagonal line between (0, 0) and (1, 1), which has 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 comparision [75]. However, in practice researchers recently noticed that AUC was quite noisy as a

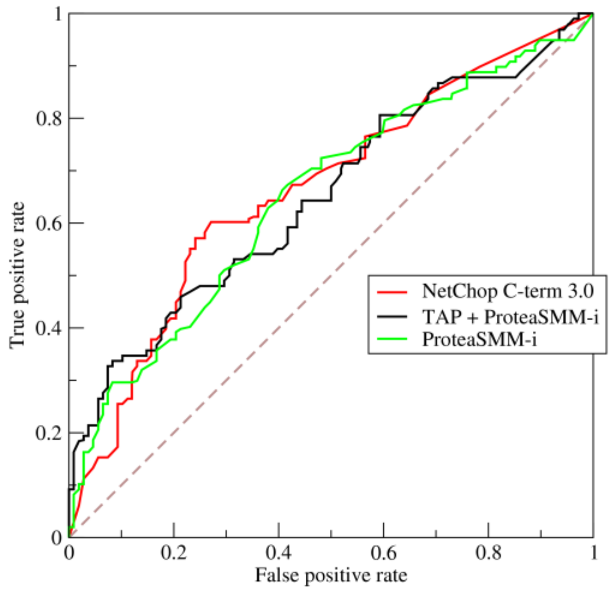


Fig. 2.6 ROC curve of three predictors of peptide cleaving in the proteasome

classification measure [73] and some other studies have other significant problems in model comparison [124, 74].

F-score

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

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

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

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

Van Rijsbergen et al. [178] 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 imbalanced problem.

Chapter 3

Fraud Detection Pipeline

There are many researches and studies for fraud detection problem and related issues, including what algorithms should be used [118, 16, 127], how to prepare features [6, 193, 55], ... 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 that is not only simple, reliable but also easy to implement. Furthermore, we provide the readers many potential things that could improve the pipeline.

3.1 Imbalanced dataset

One of main problems of fraud detection is our dataset is very imbalanced since a number of fraud is a small fraction of the dataset [95]. The imbalanced problem receives 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 [56] and credit card transactions [23], 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, imbalance is a big problem for the learning process. Let consider an example consists only 1% fraudulent transactions (minority class) and the remaining belong to legitimate category (majority class), a lazy classification that predict 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 can't detect any fraudulent transactions.

There are several methods 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 algorithmic level [28]. 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, algorithms try to improve an accuracy of the minority class prediction, while the second learning try to minimize the cost of wrong predictions.

Imbalance learning

Decision tree, e.g C4.5 [149], use Information Gain as splitting criteria to maximize number of predicted instances in each node, it makes the tree bias towards the majority class. Cieslak and Chawla [33] suggest splitting with Hellinger Distance (HD) which they show that HD is skew-insensitive and their proposed Hellinger Distance Decision Tree has better performance compared to standard C4.5. Other studies have also reported the negative effect of skewed class distributions not only on decision tree [82, 91], but also on Neural Network [91, 182], k-Nearest Neighbor (kNN) [112, 129, 9] and SVM [198, 197].

In Machine Learning, with a greate success of ensemble learning for several applications there are many ensemble strategies have been proposed for imbalanced learning. Bagging [18] and Boosting [62] are the most popular strategies which combine an unbalanced strategy with a classifier to aggregate classifiers [122, 189, 181, 96, 123, 185, 25, 94, 130].

Cost-sensitive learning

Classification applications dealing with imbalanced datasets, the correct prediction of minority class is more important than the correct prediction of majority class, it causes several classifiers fail when predicting minority class because they assume the cost of these class is the same. In credit card fraud problem, you can imagine that the cost of our true prediction is zero, and if we can't 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).

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 [121]; or used in tree pruning[17]. An extension of cost-sensitive learning,

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

Domingos proposed Metacost [43] framework that transforms non-cost-sensitive algorithm into a cost-sensitive algorithm. However, the cost for minority class usually not available or hard to compute, it makes the use of cost-sensitive algorithms are not popular [128].

Data level methods

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

Sampling

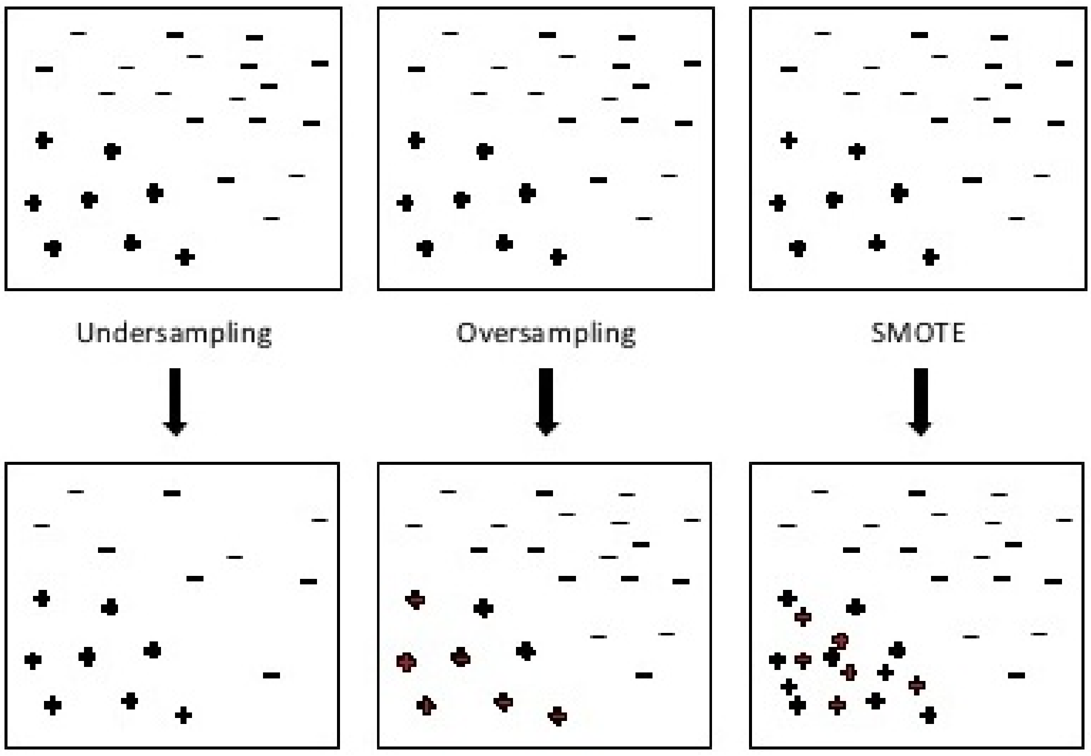
Several studies [191, 114, 49] have shown that balanced training set will give a better performance when using normal algorithms, and sampling methods are most common techniques in data science used to create a balanced training set. There are three popular sampling techniques, including undersampling, oversampling and SMOTE (see figure 3.1); and some extension sampling techniques based on these three techniques, e.g Borderline-SMOTE [71], ADASYN [81], ...

Undersampling [44] is a method tries to decrease the size of majority class by removing instances at random. Its idea is that many instances of the majority class are redundant therefore the removal of these instances make its distribution change not too much. However, the risk of dropping redundant instances still exists since this process is unsupervised and we can't control what instances will be dropped. Furthermore, a perfectly balanced dataset, which means the number of minority class equals the number of majority class, is not a good choice for undersampling [37]. In practice, this technique is often used because it's simple and speeds up our process.

Oversampling [44] 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 classifier toward the minority class [44]. Furthermore, this method doesn't add any new informative for minority instances and it also slow down our learning phase.

SMOTE [27] 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 [27], 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 increase an overlapping area between classes [184].

Fig. 3.1 Three common sampling methods



Cost-based

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

Costing [203] is a cost-based undersampling that drawns instances with an accepting probability greater than a pre-defined threshold. Klement et al. [100] proposed a more complicated and effective approach that combine cost-based random under-sampling and ensemble learning. The cost-based methods are promising when dealing with imbalanced datasets, however, the cost of misclassification for each class maybe 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, 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. These methods need to compute the distance between instances so these methods are more costly than the above methods.

Tomek [173] 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 overlapping problem since it removes those instances which can make a classifiers toward misclassification [168]. There are several distance-based sampling studies, e.g Condensed Nearest Neighbor [79], One Sided Selection [112], Edited Nearest Neighbor [196] or Neighborhood Cleaning Rule [114]

3.2 Algorithms

In the fraud detection problem, almost Machine Learning algorithms, including supervised learning [23, 112, 12], unsupervised learning [172, 16], ..., have been proposed to detect fraudulent transactions. In this section, we will see which algorithms are used to solve 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 the human expert. Expert's rules are just simple IF-THEN rules and extremely manually since all of the rules based on expert's knowledge and their ability. By applying expert system suspicious activity or transaction can be detected from deviations from normal spending patterns [119]. Kevin at el. proposed expert system model to detect fraud for alert financial institutions [118]. 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 [78].

Association Rules

Association rules are an extended approach of Expert Systems 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 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 [160]. However, the expert's rules are still generated manually and it is a 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 nonlinear statistical data modeling tools that are inspired by the functionality of the human brain using a set of interconnected nodes [200, 68]. Neural networks are widely applied in classification and clustering, and its advantages are as follows. First, it is adaptive; second, it can generate robust models; and third, the classification process can be modified if new training weights are set. Neural networks are chiefly applied to credit card, automobile insurance and corporate fraud.

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 [69]. The neural networks can be used to predict the occurrence of corporate fraud at the management level [22].

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 [99].

The study in [54] revealed that input vector consisted of financial ratios and qualitative variables, was more effective when fraud detection model was developed using neural network. The model was also compared with standard statistical methods like linear and quadratic discriminant analysis, as well as logistic regression methods [54].

The generalized adaptive neural network architectures and the adaptive logic network are well received for fraud detection. The hybrid techniques like fuzzy rule integrated with a neural network (neuro-fuzzy systems) are also proposed. The literature describes that the integrated fuzzy neural network outperformed traditional statistical models and neural networks models reported in prior studies [120].

The Bayesian belief network (BBN) represents a set of random variables and their conditional independencies using a directed acyclic graph (DAG), in which nodes represent random variables and missing edges encode conditional independencies between the variables [99]. The Bayesian belief network is used in developing models for credit card, automobile insurance, and corporate fraud detection. Bayesian belief network outperformed neural network and decision tree methods and achieved outstanding classification accuracy [99].

Support Vector Machine (SVM)

Support vector machine (SVM) [34] is a supervised statistical learning with a goal is to find an optimal hyperplane which can maximize the margin of the training data classes. SVMs are linear classifiers that work in a high-dimensional feature space, an advantage of working in a high-dimensional feature space is that a non-linear relationship between classes which are hard to model in the original input space, but it will become a linear relationship in a higher dimensional feature space. An important property of SVMs is *kernel representation*, a kernel function [162] can represent the dot product of projections of two data points in a high-dimensional feature space.

In credit card fraud detection, Ghosh and Reilly [68] have developed a model using SVMs or Chen et al. [31] proposed a binary support vector system (BSVS) in which support vectors were selected by means of the genetic algorithms (GA). There are also many studies used SVMs and its extension version for fraud detection problem [158, 30, 125]. Since its strength is kernel function, it's also a disadvantage because how to choose good kernel function is a hard question.

Bayesian Network

A Bayesian network is a graphical model that represents conditional dependencies among random variables. The underlying graphical model is in the form of directed acyclic graph. Bayesian networks are usefulfor finding unknown probabilities given known probabilitiesin the presence of uncertainty [137]. Bayesian networks can play an important and effective role in modeling situations where some basic information is already known but incoming data is uncertain or partially unavailable [144, 24].The goal of using Bayes rules is often the prediction of the class label associated to a given vector of features or attributes [147]. Bayesian networks have been successfully applied to various fields of interest for instance detect anomaly and frauds in credit card transactions or telecommunication networks [105, 21, 126].

In [147], two approaches are suggested for credit card fraud detection using Bayesian network.In the first, the fraudulent user behavior and in the second the legitimate (normal) user behavior are modeled by Bayesian network. The fraudulent behavior net is constructed fromexpert knowledge, while the legitimate net is set up in respect to available data from non fraudulent users. During operation, legitimatenet is adapted to a specific user based on emerging data.Classification of new transactions were simplyconducted by inserting it to both networksand then specify the type of behavior (legitimate/fraud) according to correspondingprobabilities. Applying Bayes rule, gives the probability of fraud for new transactions [42]. They claimed that lots of popular methods such as regression, K-nearest neighbor and neural networks takes too long time to be applicable in their data.

Fuzzy Logic Based System

Fuzzy logic based system is the system based on fuzzy rules. Fuzzy logic systems address the uncertainty of the input and output variables by defining fuzzy sets and numbers in order to express values in the form of linguistic variables (e.g. small, medium and large). Two important types of these systems are fuzzy neural network and fuzzy Darwinian system.

Fuzzy Neural Network (FNN) The aim of applying Fuzzy Neural Network (FNN) is to learn fromgreatnumber of uncertain and imprecise records of information, which is very common in real world applications [205]. Fuzzy neural networks proposedin [169] to accelerate rule induction for fraud detection in customer specific credit cards. In this research authors applied GNN (Granular Neural Network) method which implements fuzzy neural network based on knowledge discovery (FNNKD), for accelerating thetraining network and detecting fraudster in parallel.

Fuzzy Darwinian System Fuzzy Darwinian Detection [11] is a kind of Evolutionary-Fuzzy system that uses genetic programming in order to evolve fuzzy rules. Extracting the rules, the systemcan classify the transactions into fraudulent and normal. This system was composed of genetic programming (GP) unit in combination with fuzzy expert system. Results indicated that the proposed system has very high accuracy and low false positive rate in comparison with other techniques, but it is extremely expensive [50].

Genetic Algorithm (GA)

Inspired from natural evolution, Genetic algorithms (GA), were originally introduced by John Holland [87]. GA searches for optimum solution with a population of candidate solutions that are traditionally represented in the form of binary strings called chromosomes.

The basic idea is that the stronger members of the population have more chance to survive and reproduce. The strength of a solution is its capability to solve the underlying problem which is indicated by fitness. New generation is selected in proportion to fitness among previous population and newly created offspring. Normally, new offspring will be produced by applying genetic operators such as mutation and crossing over on some fitter members of current generation (parents). As generations progress, the solution are evolved and the average fitness of population increases.This process is repeated until some stopping criteria, (i.e. often passing a pre-specified number of generations) is satisfied.

Genetic Programming (GP) [76] is an extension of genetic algorithms that represent each individual by a tree rather than a bit string. Due to hierarchy nature of the tree, GP can produce various types of model such as mathematical functions, logical and arithmetic expressions, computer programs, networks structures, etc.

Genetic algorithms have been used in data mining tasks mainly for feature selection. It is also widely used in combination with other algorithms for parameter tuning and optimization. Due to availability of genetic algorithm code in different programming languages, it is a popular and strong algorithm in credit card fraud detection. However, GA is very expensive in consuming time and memory. Genetic programming has also various applications in data mining as classification tool.

EkremDuman et al. developed a method for credit card fraud detection [47]. They defined a costsensitive objective function that assigned different cost to different misclassification errors (e.g. false positive, false negative). In this case, the goal of a classifier will be the minimization of overall cost instead of the number of misclassified transactions. This is due the fact that the correct classification of some transactions was more important than others. The utilized classifier in this work was a novel combination of the genetic algorithms and the scatter search. For evaluating the proposed method, it was applied to real data and showed promising result in comparison to literature. Analyzing the influence of the features in detecting fraud indicated that statistics of the popular and unpopular regions for a credit card holder is the most important feature. Authors excluded some type of features such as the MCC and country statistics from their study that resulted in less generality for typical fraud detection problem.

K.RamaKalyaniet al. [151] presented a model of credit card fraud detection based on the principles of genetic algorithm. The goal of the approach was first developing a synthetizing algorithm for generating test data and then to detect fraudulent transaction with the proposed algorithm.

Bentley et al. [11] developed a genetic programming based fuzzy system to extract rules for classifying data tested on real home insurance claims and credit card transactions.

Logistic Regression (LR)

Logistic Regression [183, 35] is an important machine learning algorithm, a 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 [89]. There are many studies tried to use logit models to estimate a fraudulent probability in the case of insurance frauds [93, 3] and other related areas like food stamp programs, and so forth [15, 80, 146]

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, it means that it could generate expert-rules IF-THEN automatically. In data mining, decision trees can be described as the combination of mathematical and computational techniques to aid the description, categorization, and generalization of a given set of data.

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 [158, 157, 5, 117].

Hidden Markov Model (HMM)

A Hidden Markov Model is a double embedded stochastic process which is applied to model much more complicated stochastic processes as compared to a traditional Markov model. The underlying system is assumed to be a Markov process with unobserved states. In simpler Markov models like Markov chains, states are definite transition probabilities are only unknown parameters. In contrast, the states of a HMM are hidden, but state dependent outputs are visible.

In credit card fraud detection a HMM is trained for modeling the normal behavior encoded in user profiles [165]. According to this model, a new incoming transaction will be classified to fraud if it is not accepted by model with sufficiently high probability.Each user profile contains a set of information about last 10 transactions of that user liketime; category and amount of for each transaction [165, 164].

HMM can also be embedded in online fraud detection systems which receive transaction details and verify whether it is normal or fraudulent.If the system confirms the transaction to be malicious, an alarm is raised and related bank rejects that transaction. The responding cardholder may then be informed about possible card misuse.

Nearest Neighbour Method

Nearest neighbour method is a similarity based classification approach. Based on a combination of the classes of the most similar k record(s), every record is classified. Sometimes this method is also known as the k-nearest neighbour technique [72]. K-nearest neighbour method is used in automobile insurance claims fraud detection [72] and for identifying defaults of credit card clients [200].

Random Forest

Random Forest [19] 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 to solve the disadvantage of Decision Tree [111].

Several studies have shown that it achieves the best results among different classifiers [38, 36, 179, 193, 12, 7] when using RF. Pozzolo et al. [39] have shown that the Random Forest combined with undersampling or SMOTE are often a best choice, in our case we refer undersampling over SMOTE since our data is usually huge. Therefore we propose using Random Forest with undersampling for many reasons, e.g it's a powerful algorithm, could interpret the result to business man, a fast algorithm, ...

3.3 Sample Selection Bias

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

This situation could occurs 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 it's a Sample Selection Bias (SSB), it's also seen in the collecting data step when we do not or could not collect data that present a 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:

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

As initially proposed by Zadrozny [201] 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 [48, 203, 202, 201, 53, 46]. One approach for this problem is *important sampling* that assign a weight to each data point [203, 201, 53]. 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's 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 [52].

In previous section, we prefered to use Random Forest with undersampling method and this approach could leads 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 subset by applying undersampling, each subset will be used as 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 a below chapter.

3.4 Feature Engineering

One of the most important steps in Machine learning is Feature Engineering which we will pre-processing features in the dataset to improve the accuracy of a algorithm. The Feature

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

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 transaction which are collected in real time,
- 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 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 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 [116]. 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, ... [38, 110, 193, 12, 92].

Whitrow et al. proposed a transaction aggregation strategy in order to aggregate customer behavior [193]. 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 [12, 7, 38, 92, 157, 171, 192]. Whitrow et al. [192] proposed a fixed time frame to be 24, 60 or 168h, Bahnsen et al. [6] 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:

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

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:

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

,

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. [6] 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 [58].

Recently Wedge et al. [190] 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, and similarly with 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) [26]. AUC shows that how much the ROC curve is close to the perfect classification, however Hand [74] considers the standard calculation of the AUC as inappropriate because it making an average of different misclassification costs for classes. F-measure is more like accurate in the sense that it's a function of a classifier and its threshold setting, 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 [157, 127, 5], cost-based measures are defined to quantify the monetary loss due to fraud [7] by computes average cost-matrix which is similar with confusion matrix. Elkan [48] states that it is safer to assess the cost-sensitive problem in terms of benefit (inverse of cost) because there is the risk of using different baselines when using a cost-matrix to measure overall cost. Dealing with this issue, normalized cost or savings [5] 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 [48, 7], 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 is fraud or not. Furthermore, there are many hidden costs that we cannot see directly, e.g a reputation cost of company or a maintenance cost of 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 the right way to measure the cost of frauds.

In a scenario with limited resources, e.g there are only 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 in 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 more better than all of the others, it's easy to compute and could be used for imbalanced problem, therefore we 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 [55].

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

Despite of these advantages, aggregating fraud network-level features is not easy and recently had received more consideration. Van Vlasselaer et al. have proposed APATE [179], a framework for credit card fraud detection that allows including network information as additional features describing a transaction. In a next year after APATE, they also proposed GOTCHA! [180], 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 a best choice for a first version of fraud detection system.

3.7 Concept Drift

As mentioned in section 3.3, the non-stationary environment could occurs when we modify the dataset. In case a data generating source changes itself over time in unforeseen ways, it's known as Concept Drift [66] or Dataset Shift [150]. With Bayes rule, we have a joint distribution of a sample (x, y) is:

$P(x,y) = P(y|x)P(x) = P(x|y)P(y)$.

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

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

From this Bayes formula, the change of data distribution at time t and $t+1$ could come from several sources[98]:

- $P(y|x)$: 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 [138],
- $P(y)$: 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 $P(x)$ doesn't affect y so we could ignore it [86]. In general it hard to say where the change comes from since we only see the generated data and we don't know or can't 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 tries 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 [70].

In summary, dealing with Concept Drift problem, our fraud detection system have 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 an assumption that labels of the dataset are correct, however in the real world environment it's maybe not true. In our case, 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, 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 bank, it makes our dataset is non-accurate and hard to modelling 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 maybe 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 is never updated will lose their accuracy over the time.

Pozzolo et al. [36] 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 $\{F_t, \dots, F_{t-(\delta-1)}, D_{t-\delta}, \dots, D_{t-(\delta+M-1)}\}$
- an ensemble of classifiers $\{\mathscr{M}_1, \mathscr{M}_2, \dots, \mathscr{M}_M, \mathscr{F}\}$ 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 $\{F_t, \dots, F_{t-(\delta-1)}\}$

Their solutions used two basic approaches for handling concept drift that can be further improved by adopting dynamic sliding windows or adaptive ensemble sizes [51]. 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's 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 didn't care about 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 run a training process very slow.

In the age of Big Data, there are some frameworks for distributed computing system and a most common one currently used in many biggest system around the world is Spark [204]. 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 [132]. However with a current version of Spark (2.2.0 [60]), there are not many algorithms are supported, it means that for those companies they have only a few options to build the detection system. Fortunately, some 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 these challenges. Based on that, in this section we analyze and propose a Fraud Detection Pipeline, which is not only easy to implement but also reliable version of Fraud Detection System.

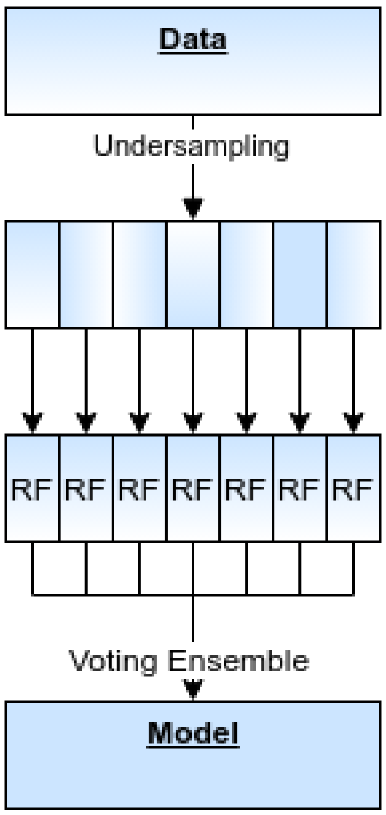
Imbalanced dataset and Algorithm

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

- 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 scope of this report. Among the rest approaches, Pozzolo et al. [39] have shown that the Random Forest combined with undersampling or SMOTE are often have a best result. In our case, we prefer undersampling over SMOTE because it could manages the huge dataset. An usage details of the undersampling and Random Forest algorithm are provided later.

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



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 ensemble method that combines results from multiple samples [52]. The dataset will be splitted into multiple dis-joint samples, each sample is used to train a Random Forest classifier then a final prediction could be mean or 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, 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's a complicated task, therefore we skip this level in our pipeline.

We propose that aggregating features to describe customer's behavior (user level and/or card level) consist in 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 is 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 most simplest and suitable measure to deal with it is F1-score. Or inside financial company, with all cost information of each transaction, we could use cost-based measures for those cost-based methods. In this report 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 pattern as soon as possible. We assume that we could update the system daily or at least every time frame, which is possible with small dataset, and in the rest of this chapter we will update all models at all time frames.

Performance

As we've 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 development phase and also in production.

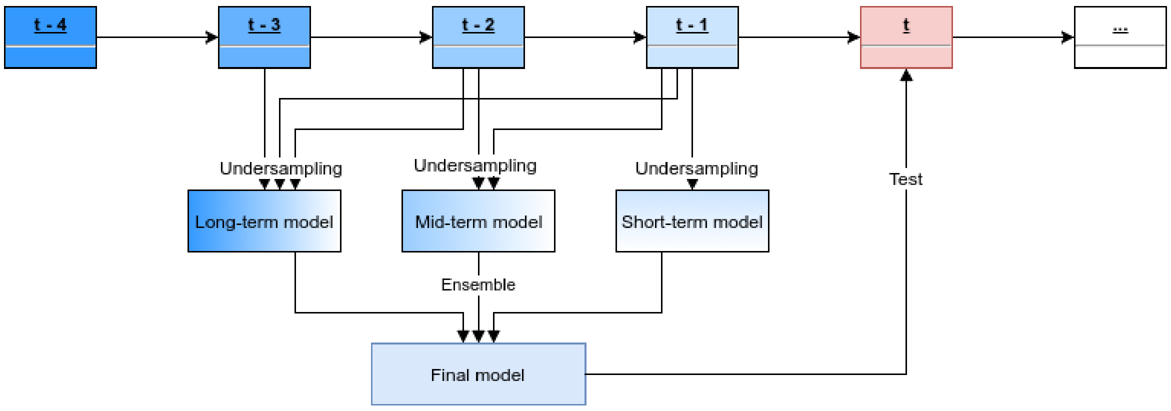


Fig. 3.3 Recommended Fraud Detection Pipeline

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 a last day or a 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 a last month or last year data,
- 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). A size of the data and its training time increase from short-term to long-term model to keep more older fraudulent patterns. The long-term model needs long time to prepare its new version and maybe not available for a 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 reduces workload for the slowest model and makes sure the system is always prepared for any possible situation.

3.11 Upgrade the Fraud Detection Pipeline

Our proposed Fraud Detection Pipeline above is a simple and easy to implement approach, there are several things could improve the system and here we summarize some most potential things that will upgrade the current system easily:

- Imbalanced dataset: there are two main ideas to resolve this problem, one is sampling methods and the other is cost-based methods. Our pipeline using the simpler way that doesn't need to know about the cost but it is very promising for 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've 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 a simplest way to upgrade the pipeline,

- Measurement: the F1 score is the most suitable choice for the imbalance problem, therefore we don't 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 resolve all of 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 fail,

- Performance: the FDS usually need to process a very large number of transactions. While the distributed-computing Spark framework could handles it and also fits with other industry's requirements, therefore we don't have any reason to find an alternative solution and we suggests using 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 old and don't have too many transactions. Fortunately there is one new and large transaction dataset which is published by Pozzolo et al. [37] 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.

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 report 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 a 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. A 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 doesn't re-train anymore and compare with daily update, the model will be improved significantly (81%). Only the long-term model with daily update has a bit lower accuracy than a never update model (1%). As we forecasted above, the more data we use in longer term model the more accuracy we have, 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.

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

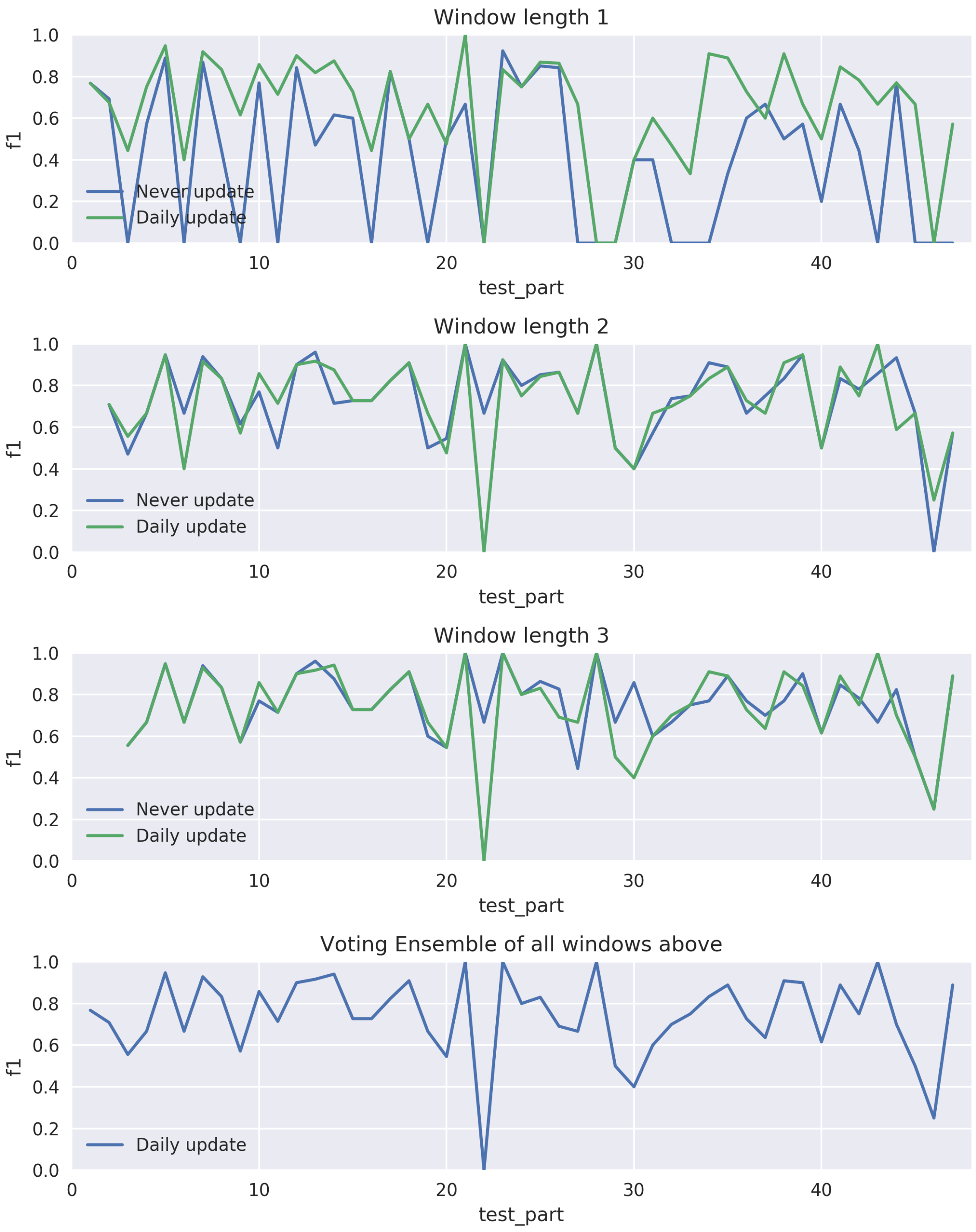


Fig. 3.4 Accuracy of our Fraud Detection Pipeline

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

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 take more than one day to complete then it's costly task for our system. In chapter 4, we propose a method to decide which model needs to update with a 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 knowlegde, 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 techinque 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 update. In fact that it only needs to update to adapt with new patterns in the data. The Concept Drift is usually seen in streaming data and there are 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 should we update our models and a result shows that our models not only need to update a few times but also have a better performance.

4.1 Introduction

One of a typical assumption not only in statistic but also in machine learning is that we assume a source of data doesn't change, i.e an current available training data and a future testing data have the same distribution. However real world problems rarely fit with this assumption, the fraud detection problem is an example since fraudsters always try to create new fraudulent strategies or new fraudsters give a try with old fraudulent strategies. In this non-stationary environment, the data characteristics could change over time yielding the phenomenon of Concept Drift [161, 194].

Each data point in dataset is evolving with time, let $X_t \in \Re^p$ is a vector in p-dimensional feature space observed at time t and y_t is the label respectively. The goal is to predict a target

y_t given a set of input feature X_t. Let S_t be a source generating the data X_t, the data source is a set of a prior probabilities of classes $P(y_t)$ and its conditional pdf $p(X \mid y_t)$. From the Bayesian decision theory [45] we can estimate the class y_t of the given instance X:

$$p(y_t \mid X) = \frac{P(y_t)p(X \mid y_t)}{p(X)},$$

Kely et al. [98] presented that based on the above formula, Concept Drift could happen in three ways:

1. The priors $P(y)$ might change
2. The distribution of target $p(X \mid y)$ might change
3. The posterior distribution $p(y \mid X)$ might change

Based on change types of the pattern of the data source over time, we can define strutural types of change as four types which are summaried in figure 4.1 [208], let consider two data sources S_1 and S_2:

- Sudden drift: a simplest type of Concept Drift when the current data source S_1 is suddenly replaced by the new data source S_2,
- Gradual drift: both source S_1 and S_2 are active however a probability of each source change over time,
- Incremental drift: similar with gradual drift but the probability of one data source is decreasing and the other is increasing,
- Reoccurring context: a slightly different kind of Concept Drift when a previous active source disappears then re-appears after some time. It is not seasonality because it's not clear when the source might appear.

4.2 Related works

Over the last decade, the Concept Drift problem has received more attention and many handling methods have been developed. Schlimmer and Granger [161] is a very first one who formulated an issue of incremental learning from noisy data and proposed an adaptive learning algorithm STAGGER that dynamically tracks changes of concepts via a sliding window. A concept drift detection framework FLORA which was proposed by Widmer et al

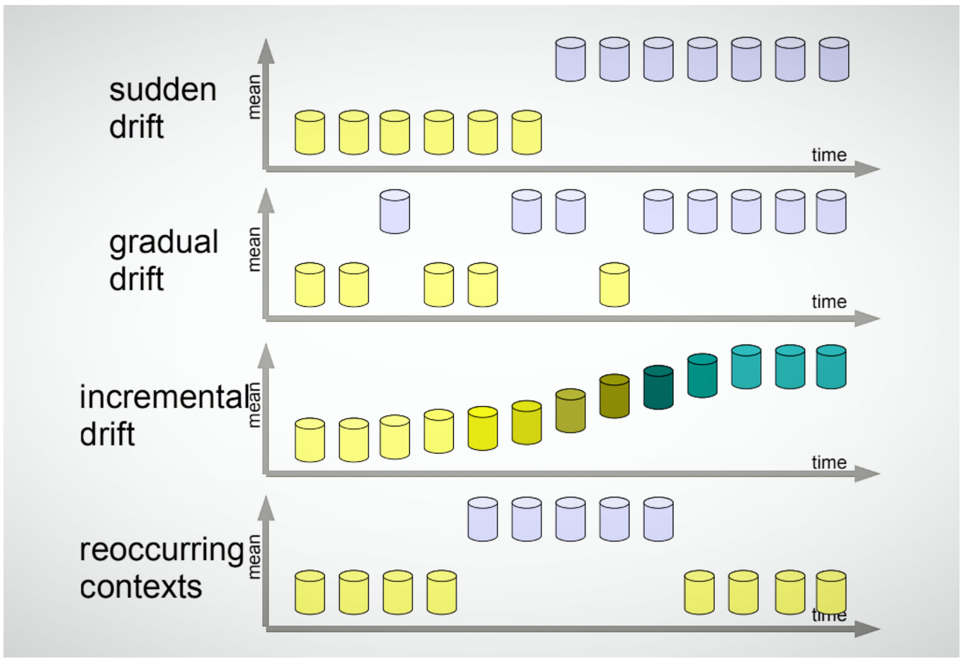


Fig. 4.1 Illustration of four types in Concept Drift

[194] used the passive batch-based instance selection method, some versions in the FLORA family consists of an active approach, or using an adaptive window narrowing. Moreover, there are several methods using heuristics for determining window length [194, 115, 4, 199] or using base learning specific methods for determining training windows [90, 207, 113, 176].

The window-based above assumes that latest data is important, another technique is weight-based is a little bit different which assign an important weight for each data point base on its age or information. A weighting function could be linear decay [106, 107] or exponential decay [101]. An adaptivity of these techniques don't come from a strength of combination, it is achieved by systematic data formation.

Bassevila et al. [8] proposed to detect Concept Drift by identifies change points. Other studies tried to detect change points via monitoring performance indicators or comparing characteristic of data[194, 206], e.g Klinkenberg et al. [103] with their pre-defined threshold then monitor three indicators: accuracy, recall and precision. The monitored indicators could not only be performance indicators, but be also any other indicators, e.g change rate of error [65].

Another approach for Concept Drift is design an algorithm that could handle this problem by itself. At a beginning a model will use all of availabel data then whenever it receives new data, a new model will use a most recent data to reconstruct itself. This approach is suitable with bath data learning [65, 167, 206, 102]. Instead of discarding old model, a model could update itself with the most recent data to reduce its workload. This incremental learning processes the data one-by-one and update parameters in the model [90, Roy et al.]

Non-directly technique to handle Concept Drift is using ensemble learning which combines multiple algorithms that are biased by Concept Drift to have an un-biased result. An idea behind this is at any time some algorithms will be biased while some are not. Street at el. presented Streaming Ensemble Algorithm (SEA) [167] which is an ensemble from fixed number of classifiers. Many other studies proposed the concept drift detection using ensemble technique like Hoeffding tree bagging [13], Dynamic Integration [177], Dynamic Weighted Majority (DWM) [104] and others [166, 167, 186, 177, 97, 163, 10, 141, 187, 51, 207, 152]

In summary, approaches in Concept Drift could be categorized in different ways based on their technique, for example:

- Model management: *single model* builds a complex model to handle Concept Drift, or *ensemble* which get results of un-biased models to support its biased models,
- Learning method: *incremental* learning includes classifiers are designed to training online for every coming data points so it could adapt with a new environment, or *retraining* approach that wait for a batch of data then discard a current model and build a new one,
- Adaption methods: *active* approach designs a system or classifier could respond to the Concept Drift automatically by update itself, or *passive* approach which only be informed after it lose accuracy,
- Selection methods: *window-based* approaches try to group the data into parts with a dynamic or fixed window length, or *weight-based* approaches score each instance by their age or their other characteristics to select relevant instances.

4.3 Methodology

In the fraud detection problem, we are dealing with the Concept Drift problem occurs in the dataset which is not only imbalanced but also very huge. In the previous section, we proposed the Fraud Detection Pipeline that could handle these issues by update the system frequently. The problem here is a cost for the update process may not be low, e.g long training time, therefore update all models in our system is not always possible.

We propose a method that predict should we update the models after we receive a most up-to-date data. The fraud dataset in usually a batch data that is available at the end of working day, e.g data in banking system. Several approaches mainly focus on the streaming data, i.e they are not suitable in our case. We propose a hybrid approach of *passive* approach and *window-based* approach that run only one time before we update the models.

Using the proposed architecture in previous section, we aim to predict an expected improvement of each model in our system at every time frame, to decide which model should be updated. With a pre-defined threshold that we expect if we update the model then it will improve its accuracy greater than or at least equals our threshold. If this threshold is too small then the models tend to update more frequent like our system above, if it's high then we rarely update our models. A main regressor in this prediction is Random Forest, a dataset is used to train the regressor including:

- Current error: a main metric to evaluate the system is F_1 score and the error will be $1 - F_1$. We don't use directly F_1 score because we want to find a relationship between current error and the improvement,
- Other metrics: we could use some other metrics, e.g precision, recall, because it maybe more useful than F_1 score,
- Pseudo score: before update our models, i.e train models on all data, take a small sample then train model on this sample to get an pseudo F_1 score of the next model fastly, a detail of this step is in below,
- Improvement: this is our target, which is the improvement on F_1 score of a model that will be trained on a current data with a current model.

Using data from a last training time $i = 1$ to current time frame $i = k$, we also apply undersampling but with a lower sampling ratio β. From the multiple time frames data, we undersampling from each time frame with different weight which is called error weight. Let $E_i = 1 - F_1^i$ is a error score of our model on data frame i for $i = 1, \ldots, k$ and k is the number of time frames, then the error weight W_i could be computed by formula:

$$W_i = \frac{E_i}{\sum_{j=1}^{k} E_j} \cdot w \cdot \frac{\beta}{k} \tag{4.1}$$

where w is a window length of the current model.

A first part of this formula from a reasoning that we want to collect more data, i.e more patterns, from a data frame with higher error score, and the last part to normalize the weight since we might have multiple data frames. From the k data frames, a final undersampling ratio W will be:

Table 4.1 The experimental results of Update Point Estimation

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
	UPE	7	N/A	N/A	**0.6615**	0.2619
2	Never update	0	0.7506	0.1632	0.72	0.2192
	Daily update	24	0.7325	0.2304	**0.722**	0.1893
	UPE	5	N/A	N/A	0.7121	0.1903
3	Never update	0	0.7808	0.1511	0.7351	0.1665
	Daily update	24	0.7571	0.2267	0.7268	0.1871
	UPE	5	N/A	N/A	**0.77**	0.1781
Voting Ensemble	Daily update	72	0.7554	0.2164	0.7261	0.1865
	UPE	17	N/A	N/A	**0.7461**	0.189

$$W = \sum_{i=1}^{k} W_i = \sum_{i=1}^{k} \left(\frac{E_i}{\sum_{j=1}^{k} E_j} w \frac{\beta}{k} \right) = w \frac{\beta}{k} \tag{4.2}$$

Respecting to the original model with window length w, our models use the undersampling data with its size is proportional to original undersampling data size, it makes sure that this step faster than building completely model. This undersampling method is a common basic way to handle imbalanced dataset, however it's not stable than our proposed pipeline and we only use this to get the pseudo result, e.g F_1 score, of next updated models.

4.4 Experiment and Conclusion

Using the same dataset and our proposed Fraud Detection Pipeline in the previous chapter, a result of this experiment is in figure 4.2 and a detail is in table 5.2, red points are when those models will be updated. It shows that our pipeline using Voting Ensemble and daily update give us a very promising results. Its F_1 score is similar with our highest accuracy and has a standard deviation as low as the lowest. For each model of three window length periods, there are only a few necessary update points (23.7%) when using our Update Point Estimation method, and it's also has a higher accuracy than our original pipeline (2%). In final, our method give the stable result but it is sensitive with the pre-defined threshold, in the future work we want to have a method with dynamic threshold to reduce its sensitivity.

Algorithm 2 Update Point Estimation

Input:
- Training dataset D_{train} with N time frames
- Pseudo sampling ratio β
- Window length w of a model

Procedure:

```
 1: $dataset \leftarrow$ a dataframe
 2: for $i := 1$ to $N-1$: do
 3:     for $j :=$ i to $N$: do
 4:         $model \leftarrow$ train a classifier on $D_{train}^{i-w,i}$
 5:         $F_1^j \leftarrow$ test the model on $D_{train}^j$ by metric $F_1$
 6:         $improvement \leftarrow F_1^j - F_1^i$
 7:         $current\_error \leftarrow 1 - F_1^j$
 8:         $errors \leftarrow$ a list
 9:         for $k := i$ to $j$: do
10:             $F_1^k \leftarrow$ test the model on $D_{train}^k$ by metric $F_1$ score
11:             $E_k \leftarrow 1 - F_1^k$
12:             append $E_k$ into the list $errors$
13:         end for
14:         $sample \leftarrow$ a dataframe
15:         $total\_error \leftarrow$ sum of the $errors$
16:         for $k := i$ to $j$: do
17:             an error weighted sampling ratio $\beta_k \leftarrow \frac{E_k}{total\_error} \cdot w \cdot \frac{\beta}{j-i}$
18:             $subsample \leftarrow$ undersampling from $D_{train}^k$ with ratio $\beta_k$
19:             append $subsample$ into $sample$
20:         end for
21:         $pseudo\_model \leftarrow$ build a classifier on $sample$ dataframe
22:         $pseudo\_score \leftarrow$ test the $pseudo\_model$ on $D_{train}^j$ by metric $F_1$
23:     end for
24: end for
25: append $E_k, pseudo\_score, improvement$ into $dataset$
26: $regressor \leftarrow$ build a Random Forest Regressor on the $dataset$
27: return $regressor$
```

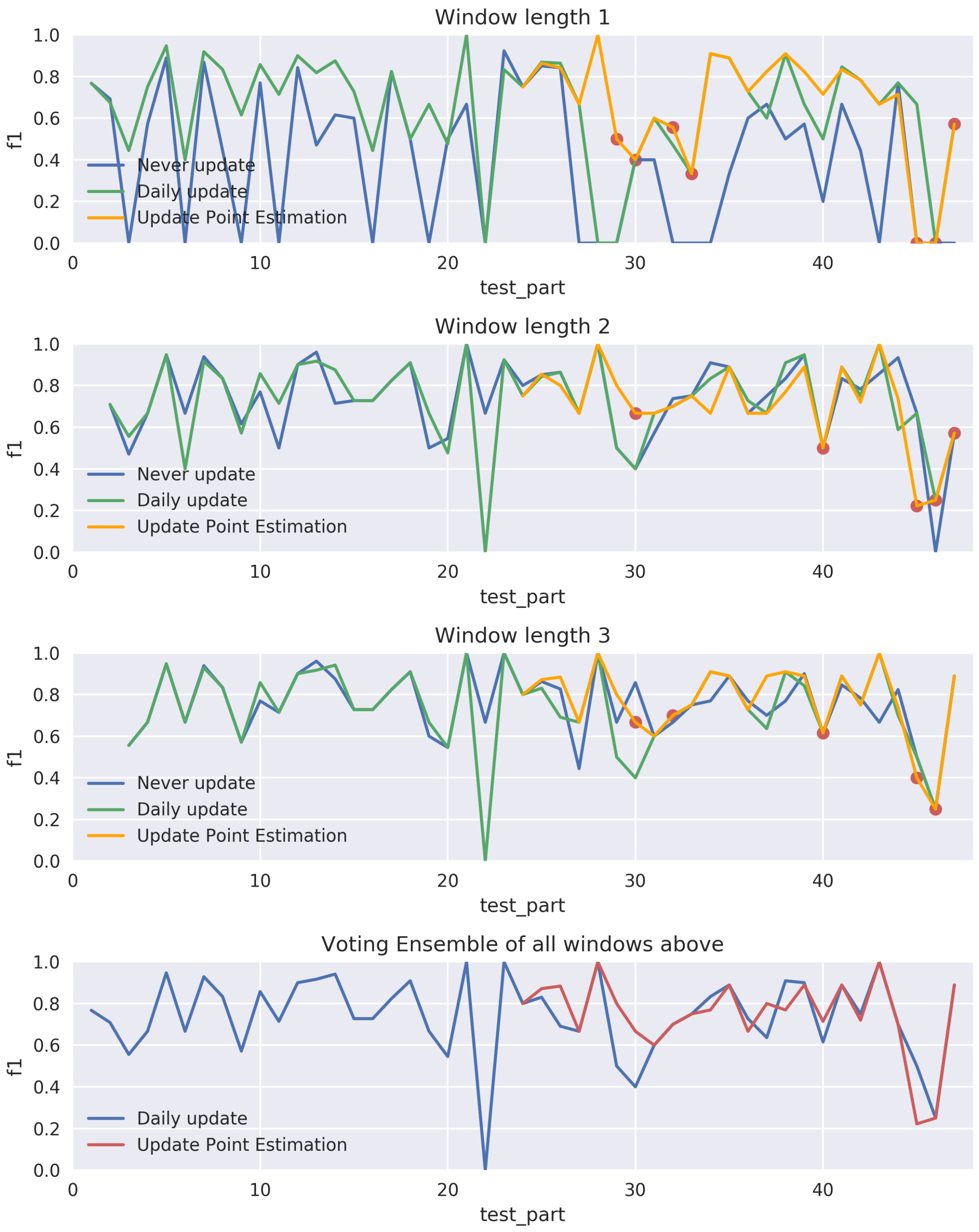


Fig. 4.2 Result of concept drift detection

Chapter 5

An Experimental Study on Undersampling and Ensemble for Extremely Imbalanced Big Data Classification

Imbalance dataset could be found in many applications from various domains, e.g fraud detection, 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 another big problem that concerns about volume and complexity of the data, the big data within extremely imbalance is a different question from those original issues and it has received attention very recently. In this study, we show that a simplified combination of under-sampling and ensemble could be more effective than old methods.

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 1 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 a worst case they always have the accuracy of 99% by doing nothing. This lazy classifier marks all samples in a dataset are 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 [29, 32, 188, 88].

To handle the imbalance problem, there are many proposed methods and these could be grouped into 2 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 [17, 33]. At the data level, original imbalanced data will be modified by pre-processing step before apply to normal classification algorithms, e.g under-sampling, over-sampling [44] or SMOTE [27]. Several studies [191, 114, 49] have shown that a balanced training set will give a better performance when using normal algorithms. Cost-sensitive approach [] 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 information about the cost which is not always available.

In the age of computing world, the size of data quickly increasing to a huge volume due to the advantages of computer technologies, a big dataset could be seen in many domains like genome biology, banking system. The size of the dataset could be million or billion records, it leads to a question that is our solution effective with a massive amount of data. Particularly in an imbalanced big data classification [40, 175, 57], old approaches could be not suitable solutions since it may create a bigger dataset then cannot fit into our resources or has a poor result. Especially, an extremely imbalance problem in a big dataset is another story that receives attention very recently [174, 109].

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 [95]. In this gap of two hard questions, we need an effective way to counter both problems at the same time and therefore, in this study we want to show a simplified combination of under-sampling 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 a gap between them. Section 3 presents our experimental analysis and comparative result on many datasets. Then finally end with a conclusion in section 4.

5.2 A new problem of extreme imbalance in big data classification

In the class imbalance problem, Mikel Galar at el. [64] 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. [40] implemented a distributed version of those common sampling techniques based on MapReduce framework [133] and they found that the under-sampling method could manage large datasets. Isaac Triguero at el. [174] faced with the extremely imbalanced big data bioinformatics problem and their solution is a complex combination of Random Forest [19] and oversampling that balanced the distribution of classes. In a review of Alberto Fernández at el. [57] 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 have been used [83, 67].

Recently, some researchers notice the extremely imbalance problem on datasets with imbalance ratio greater than 100 [170, 174]. 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 can not find any pattern 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 [40] tested their study on datasets with a largest size is over 5 million data points or a maximum IR is 74680. This new gap of two problems rises a question about how effective are we use our data, a proposed 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 balances between them and could be computed as formula 5.3. There is no study on an affect of metrics in the extreme imbalance case, therefore in this study we will 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 Experimental analysis

Based on the question identified in the previous section, we propose another way to build samples that effectively uses all of the data points in the dataset once time. Using the undersampling method, all negative data points will be randomly divided into several dis-joint equal-size subsets then each subset will merge back with all positive data points then it's 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 (K usually less than IR), a pseudocode is in Algorithm 3. A main algorithm in this study is Random Forest because of its robustness and good performance.

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

```
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:
For $k = 1$ to K:
Draw a subset $K_k^{major}$ without replacement from the majority class with $n = \frac{D^{major}}{K}$
Let $K_k$ is a merged subset of $K_k^{major}$ with all data points in minority class $D^{minor}$
Train a classifier $f^k$ by applying Random Forest algorithm to the subset $K_k$
Combine all $f_k$ into an aggregated model $F$
Return F
```

Table 5.2 Summary of datasets and the experimental results

dataset	IR	#sample	#feature	UB		3-SUS		5-SUS	
				F1	time	F1	time	F1	time
ecoli	8	336	7	**0.5758**		0.5669		0.4850	
car_eval_34	11	1728	21	**0.7173**		0.4445		0.2418	
car_eval_4	25	1728	21	**0.6078**		0.4413		0.2183	
letter_img	26	20000	16	0.6133		**0.8681**		0.8381	
webpage	34	344780	300	0.3783		0.1842		0.3130	
mammography	42	11183	6	0.4033	5	0.6478	1	**0.6606**	1
protein_homo	111	145751	74	0.4877	24	0.7908	7	0.7953	7
fraud_detection	577	284807	29	0.2341	148	**0.8113**	30	0.6231	30
kddcup SF PROBE	7988	703067	30	0.0106		0.7895		0.8034	
kddcup DOS vs. R2L	3448	4856151							
kddcup DOS vs. U2R	74680	4856151							

Our method duplicates minority class K times, but $IR >> K$ so the size of a new dataset is roughly the same as the original one. Comparing with a Balanced Bagging algorithm (BB), to have the same number of negative data points used to train classifiers it needs to duplicate minority class IR times, i.e the dataset is at twice. It means that we are twice as effective than the old balanced bagging method in the final number of used data points.

In this experiment, we use xx datasets with a variety of aspect, e.g from small to large datasets, from low to very high imbalance ratio and tries 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 a running time is also reported in seconds. This study tested on one single machine with 16 cores and 32 gigabytes memory. Details of result are summarized in a table 5.2.

The results in bold indicate the best F_1 scores and with those datasets have low imbalance ratio (i.e IR < 50) that the Under Bagging algorithm could handle the problem, but when the imbalance ratio increases then the F_1 score also decreases to zero (in case IR = 94199). From a high to extreme imbalance cases, i.e IR $>= 50$, our method can beats the Under Bagging in all datasets with any value of our chosen K values, except K = 20. The running time is similar with the F_1 score, we need to run multiple hours for the Under Bagging but with our method it only takes a few minutes.

5.4 Conclusion

In this study, we have presented the K-Segments Under Bagging algorithm in the case of extreme imbalance and big data classification. Based on the experimental result, it shows that our method not only always outperforms the old method but also run extremely fast. We also noticed the K in range 5 to 10 could give the best results, it suggests that in other applications with the similar problem we could tune this method with K belong to this range. With this new and challenging gap of two problems, we have shown that a simplified combination of undersampling technique and ensemble learning could have a good result instead of using a complicated method. In future works, we want to investigate deeper on this gap to have better methods.

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

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 card holder 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 [1], k-nearest neighbors technique [1], Local Outlier Factor technique [1], to name a few. The supervised learning techniques used to detect credit cards' fraud transactions are Hidden Markov Model technique [2], neural network technique [3], Support Vector Machine technique [4], to name a few.

To the best of our knowledge, the graph based semi-supervised learning techniques [5] 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 [6, 7] 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 [8] and the un-normalized graph p-Laplacian based semi-supervised learning algorithms combined with Cluster Centroids technique [8]. 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 $w_{ij} > 0\,(1 \leq i, j \leq n)$.

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

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

$$d_i = \sum_{j \sim i} w_{ij}, \; (1)$$

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

$$< f, g >_{V} = \sum_{i \in V} f_i g_i \; (2)$$

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

$$< F,G >_E = \sum_{(i,j)\in E} F_{ij} G_{ij} \ (3)$$

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

$$(df)_{ij} = \sqrt{w_{ij}} \left(f_j - f_i\right) \ (4)$$

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

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

$$< df, F >_{H(E)} = < f, -divF >_{H(V)}, \ (5)$$

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

Thus, we have

$$(divF)_j = \sum_{i\sim j} \sqrt{w_{ij}} \left(F_{ji} - F_{ij}\right) \ (6)$$

6.4 Laplace operator

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

$$\Delta f = -\frac{1}{2} div(df) \ (7)$$

Thus, we have

$$(\Delta f)_j = d_j f_j - \sum_{i\sim j} w_{ij} f_i \ (8)$$

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**

$$D_f S_2 = 2\Delta f \ (9)$$

The proof of the above theorem can be found from [6, 7].

6.5 Curvature operator

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

$$\kappa f = -\frac{1}{2} div\left(\frac{df}{||df||}\right) \ (10)$$

Thus, we have

$$(\kappa f)_j = \frac{1}{2}\sum_{i\sim j} w_{ij}\left(\frac{1}{||d_i f||} + \frac{1}{||d_j f||}\right)\left(f_j - f_i\right) \ (11)$$

From the above formula, we have

$$d_i f = \left((df)_{ij} : j \sim i\right)^T \ (12)$$

The local variation of f at i is defined to be

$$||d_i f|| = \sqrt{\sum_{j\sim i} (df)_{ij}^2} = \sqrt{\sum_{j\sim i} w_{ij}\left(f_j - f_i\right)^2} \ (13)$$

To avoid the zero denominators in (11), the local variation of f at i is defined to be

$$||d_i f|| = \sqrt{\sum_{j\sim i} (df)_{ij}^2 + \varepsilon}, \ (14)$$

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**

$$D_f S_1 = \kappa f \ (15)$$

The proof of the above theorem can be found from [6, 7].

6.6 p-Laplace operator

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

$$\Delta_p f = -\frac{1}{2} div \left(\|df\|^{p-2} df \right) \ (16)$$

Thus, we have

$$(\Delta_p f)_j = \frac{1}{2} \sum_{i \sim j} w_{ij} \left(\|d_i f\|^{p-2} + \|d_j f\|^{p-2} \right) \left(f_j - f_i \right) \ (17)$$

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

$$D_f S_p = p \Delta_p f \ (18)$$

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

$$argmin_{f \in H(V)} \{ S_p(f) + \frac{\mu}{2} \|f - y\|^2 \} \ (19)$$

The first term in (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 (19) is

$$argmin_{f \in H(V)} \{ \frac{1}{p} \sum_i \|d_i f\|^p + \frac{\mu}{2} \|f - y\|^2 \} \ (20)$$

By theorem 3, we have

Theorem 4: The solution of (30) satisfies

$$\Delta_p f + \mu\,(f - y) = 0, (21)$$

The *p-Laplace* operator is a non-linear operator; hence we do not have the closed form solution of equation (21). Thus, we have to construct iterative algorithm to obtain the solution. From (21), we have

$$\frac{1}{2}\sum_{i\sim j} w_{ij}\left(\|d_i f\|^{p-2} + \|d_j f\|^{p-2}\right)\left(f_j - f_i\right) + \mu\left(f_j - y_j\right) = 0 \text{ (22)}$$

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

$$m_{ij} = \frac{1}{2} w_{ij}\left(\|d_i f\|^{p-2} + \|d_j f\|^{p-2}\right) \text{ (23)}$$

Then equation (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

$$\left(\sum_{i\sim j} m_{ij} + \mu\right) f_j = \sum_{i\sim j} m_{ij} f_i + \mu y_j \text{ (24)}$$

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

$$p_{ij} = \begin{cases} \frac{m_{ij}}{\sum_{i\sim j} m_{ij} + \mu} & \text{if } i \neq j \text{ (25)} \\ \frac{\mu}{\sum_{i\sim j} m_{ij} + \mu} & \text{if } i = j \end{cases}$$

Then

$$f_j = \sum_{i\sim j} p_{ij} f_i + p_{jj} y_j \text{ (26)}$$

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 (20).

6.8 Experiments and results

Datasets

In this paper, we use the transaction dataset available from [9]. 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 [8] 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

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

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 shows 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 particularly challenging and complex problem in many real-world applications. This thesis investigated on using Machine Learning and Data Science to address some of issues in this problem. This chapter summarizes the main results of this thesis, discusses open issues and present our future works.

The chapter 3 is the survey on several challenges of the fraud detection problem, with each challenge we also presented some most suitable approaches. After they are analyzed, we have proposed the most simple and effective approach to build the Fraud Detection Pipeline that could be used in production. With this comprehensive survey, we want to write more details to contribute to the community.

Based on our Fraud Detection Pipeline, which requires to update all models in the system every time frame, 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 imbalance. 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 more bigger or more imbalance. In future work, we want to investigate more deeper on this new gap of two hard problems, e.g feature selection on the extremely imbalanced big data classifcation.

Most of the Machine Learning algorithms are applied to the fraud detection problem, but the graph-based learning is still not 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 most hardest task in the fraud detection problem and the graph-based learning receives more attention recently. In future work, we want to apply the graph-based learning to detect the fraud networks on 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] Bach, S. H. and Maloof, M. A. (2008). Paired learners for concept drift. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 23–32. IEEE.

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

[6] 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.

[7] 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.

[8] Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs.

[9] 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.

[10] Becker, H. and Arias, M. (2007). Real-time ranking with concept drift using expert advice. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 86–94. ACM.

[11] Bentley, P. J., Kim, J., Jung, G.-H., and Choi, J.-U. (2000). Fuzzy darwinian detection of credit card fraud. In *the 14th Annual Fall Symposium of the Korean Information Processing Society*, volume 14.

[12] 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.

[13] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148. ACM.

[14] 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.

[15] 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.

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

[17] 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.

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

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

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

[21] Buschkes, R., Kesdogan, D., and Reichl, P. (1998). How to increase security in mobile networks by anomaly detection. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, pages 3–12. IEEE.

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

[23] 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.

[24] Charniak, E. (1991). Bayesian networks without tears. *AI magazine*, 12(4):50.

[25] 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.

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

[27] 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.

[28] 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.

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

[30] Chen, R., Chen, T., Chien, Y., and Yang, Y. (2005a). Novel questionnaire-responded transaction approach with svm for credit card fraud detection. *Advances in Neural Networks–ISNN 2005*, pages 821–821.

[31] Chen, R.-C., Chen, T.-S., and Lin, C.-C. (2006). A new binary support vector system for increasing detection rate of credit card fraud. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(02):227–239.

[32] Chen, X.-w., Gerlach, B., and Casasent, D. (2005b). 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.

[33] 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.

[34] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

[42] Dheepa, V. and Dhanapal, R. (2009). Analysis of credit card fraud detection methods. *International journal of recent trends in engineering*, 2(3):126.

[43] 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.

[44] 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.

[45] Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

[46] 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.

[47] Duman, E. and Ozcelik, M. H. (2011). Detecting credit card fraud by genetic algorithm and scatter search. *Expert Systems with Applications*, 38(10):13057–13063.

[48] 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.

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

[50] Estévez, P. A., Held, C. M., and Perez, C. A. (2006). Subscription fraud prevention in telecommunications using fuzzy rules and neural networks. *Expert Systems with Applications*, 31(2):337–344.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

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

[57] 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.

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

[59] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188.

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

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

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

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

[64] 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.

[65] Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer.

[66] 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.

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

[68] 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.

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

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

[71] 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.

[72] Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[73] 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.

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

[75] 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.

[76] Hansen, J. V., Lowry, P. B., Meservy, R. D., and McDonald, D. M. (2007). Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. *Decision Support Systems*, 43(4):1362–1374.

[77] Har-Peled, S., Roth, D., and Zimak, D. (2003). Constraint classification for multiclass classification and ranking. In *Advances in neural information processing systems*, pages 809–816.

[78] 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.

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

[80] 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.

[81] 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.

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

[83] 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.

[84] 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.

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

[86] 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.

[87] Holland, J. (1975). Adaptation in natural and artificial systems: an introductory analysis with application to biology. *Control and artificial intelligence*.

[88] 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.

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

[90] Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM.

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

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] Karnick, M., Ahiskali, M., Muhlbaier, M. D., and Polikar, R. (2008). Learning concept drift in nonstationary environments using an ensemble of classifiers based approach. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 3455–3462. IEEE.

[98] 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.

[99] 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.

[100] 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.

[101] Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300.

[102] Klinkenberg, R. and Joachims, T. (2000). Detecting concept drift with support vector machines. In *ICML*, pages 487–494.

[103] Klinkenberg, R. and Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts. *Learning for Text Categorization*, pages 33–40.

[104] Kolter, J. Z. and Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(Dec):2755–2790.

[105] Kou, Y., Lu, C.-T., Sirwongwattana, S., and Huang, Y.-P. (2004). Survey of fraud detection techniques. In *Networking, sensing and control, 2004 IEEE international conference on*, volume 2, pages 749–754. IEEE.

[106] Koychev, I. (2000). Gradual forgetting for adaptation to concept drift. Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning,.

[107] Koychev, I. (2002). Tracking changing user interests through prior-learning of context. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 223–232. Springer.

[108] 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.

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

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

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

[112] 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.

[113] Last, M. (2002). Online classification of nonstationary data streams. *Intelligent data analysis*, 6(2):129–147.

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

[115] Lazarescu, M. M., Venkatesh, S., and Bui, H. H. (2004). Using multiple windows to track concept drift. *Intelligent data analysis*, 8(1):29–59.

[116] 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.

[117] 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.

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

[119] 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.

[120] Lin, J. W., Hwang, M. I., and Becker, J. D. (2003). A fuzzy neural network for assessing the risk of fraudulent financial reporting. *Managerial Auditing Journal*, 18(8):657–665.

[121] 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.

[122] 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.

[123] 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.

[124] 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.

[125] Lu, Q. and Ju, C. (2011). Research on credit card fraud detection model based on class weighted support vector machine. *Journal of Convergence Information Technology*, 6(1).

[126] Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (2002). Credit card fraud detection using bayesian and neural networks. In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, pages 261–270.

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

[128] 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.

[129] 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.

[130] 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.

[131] Menard, S. (2002). *Applied logistic regression analysis*, volume 106. Sage.

[132] 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.

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

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

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

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

[137] Monedero, I., Biscarri, F., León, C., Guerrero, J. I., Biscarri, J., and Millán, R. (2012). Detection of frauds and other non-technical losses in a power utility using pearson coefficient, bayesian networks and decision trees. *International Journal of Electrical Power & Energy Systems*, 34(1):90–98.

[138] 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.

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

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

[141] Nishida, K., Yamauchi, K., and Omori, T. (2005). Ace: Adaptive classifiers-ensemble system for concept-drifting environments. *Multiple Classifier Systems*, 3541:176–185.

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

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

[144] Panigrahi, S., Kundu, A., Sural, S., and Majumdar, A. K. (2009). Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning. *Information Fusion*, 10(4):354–363.

[145] 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.

[146] 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.

[147] Pun, J. and Lawryshyn, Y. (2012). Improving credit card fraud detection using a meta-classification strategy. *International Journal of Computer Applications*, 56(10).

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

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

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

[151] RamaKalyani, K. and UmaDevi, D. (2012). Fraud detection of credit card payment system by genetic algorithm. *International Journal of Scientific & Engineering Research*, 3(7):1–6.

[152] Rodríguez, J. J. and Kuncheva, L. I. (2008). Combining online classification approaches for changing environments. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 520–529. Springer.

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

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

[Roy et al.] Roy, S., Patel, B., Purandare, S., and Kucheria, M. Efficient classification of big data using vfdt (very fast decision tree).

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

[157] 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.

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

[159] 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.

[160] 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.

[161] Schlimmer, J. C. and Granger, R. H. (1986). Incremental learning from noisy data. *Machine learning*, 1(3):317–354.

[162] Schölkopf, B. and Smola, A. J. (2002). Learning with kernels. 2002.

[163] Scholz, M. and Klinkenberg, R. (2007). Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28.

[164] 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).

[165] Srivastava, A., Kundu, A., Sural, S., and Majumdar, A. (2008). Credit card fraud detection using hidden markov model. *IEEE Transactions on dependable and secure computing*, 5(1):37–48.

[166] Stanley, K. O. (2003). Learning concept drift with a committee of decision trees. *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*.

[167] Street, W. N. and Kim, Y. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM.

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

[169] Syeda, M., Zhang, Y.-Q., and Pan, Y. (2002). Parallel granular neural networks for fast credit card fraud detection. In *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, volume 1, pages 572–577. IEEE.

[170] 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.

[171] 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.

[172] 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.

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

[174] 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.

[175] 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.

[176] Tsai, C.-J., Lee, C.-I., and Yang, W.-P. (2009). Mining decision rules on data streams in the presence of concept drifts. *Expert Systems with Applications*, 36(2):1164–1178.

[177] Tsymbal, A., Pechenizkiy, M., Cunningham, P., and Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information fusion*, 9(1):56–68.

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

[179] 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.

[180] 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*.

[181] 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.

[182] 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.

[183] 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.

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

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

[186] Wang, H., Fan, W., Yu, P. S., and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. AcM.

[187] Wang, H., Yin, J., Pei, J., Yu, P. S., and Yu, J. X. (2006a). Suppressing model overfitting in mining concept-drifting data streams. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 736–741. ACM.

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

[189] 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.

[190] 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*.

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

[192] 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.

[193] 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.

[194] Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101.

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

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

[197] 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.

[198] 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.

[199] Yang, Y., Wu, X., and Zhu, X. (2006). Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data mining and knowledge discovery*, 13(3):261–289.

[200] 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.

[201] 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.

[202] 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.

[203] 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.

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

[205] Zareapoor, M., Seeja, K., and Alam, M. A. (2012). Analysis on credit card fraud detection techniques: Based on certain design criteria. *International Journal of Computer Applications*, 52(3).

[206] Zeira, G., Maimon, O., Last, M., and Rokach, L. (2004). Change detection in classification models induced from time series data. In *Data mining in time series databases*, pages 101–125. World Scientific.

[207] Zhang, P., Zhu, X., and Shi, Y. (2008). Categorizing and mining concept drifting data streams. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 812–820. ACM.

[208] Žliobaitė, I. (2010). Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*.

Appendix A

How to install LaTeX

Windows OS

TeXLive package - full version

1. Download the TeXLive ISO (2.2GB) from https://www.tug.org/texlive/
2. Download WinCDEmu (if you don't have a virtual drive) from http://wincdemu.sysprogs.org/download/
3. To install Windows CD Emulator follow the instructions at http://wincdemu.sysprogs.org/tutorials/install/
4. Right click the iso and mount it using the WinCDEmu as shown in http://wincdemu.sysprogs.org/tutorials/mount/
5. Open your virtual drive and run setup.pl

or

Basic MikTeX - TeX distribution

1. Download Basic-MiKTeX(32bit or 64bit) from http://miktex.org/download
2. Run the installer
3. To add a new package go to Start » All Programs » MikTex » Maintenance (Admin) and choose Package Manager

4. Select or search for packages to install

TexStudio - TeX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Run the installer

Mac OS X

MacTeX - TeX distribution

1. Download the file from
   https://www.tug.org/mactex/

2. Extract and double click to run the installer. It does the entire configuration, sit back and relax.

TexStudio - TeX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Extract and Start

Unix/Linux

TeXLive - TeX distribution

Getting the distribution:

1. TexLive can be downloaded from
   http://www.tug.org/texlive/acquire-netinstall.html.

2. TexLive is provided by most operating system you can use (rpm,apt-get or yum) to get TexLive distributions

Installation

1. Mount the ISO file in the mnt directory

   ```
   mount -t iso9660 -o ro,loop,noauto /your/texlive####.iso /mnt
   ```

2. Install wget on your OS (use rpm, apt-get or yum install)

3. Run the installer script install-tl.

   ```
   cd /your/download/directory
   ./install-tl
   ```

4. Enter command ‘i’ for installation

5. Post-Installation configuration:
   http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#x1-320003.4.1

6. Set the path for the directory of TexLive binaries in your .bashrc file

For 32bit OS

For Bourne-compatible shells such as bash, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/i386-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

For 64bit OS

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/x86_64-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
```

```
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

Fedora/RedHat/CentOS:

```
sudo yum install texlive
sudo yum install psutils
```

SUSE:

```
sudo zypper install texlive
```

Debian/Ubuntu:

```
sudo apt-get install texlive texlive-latex-extra
sudo apt-get install psutils
```

Appendix B

Installing the CUED class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.