Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Master Thesis

A Systematic Framework for Machine Learning Classifier on an Imbalanced Dataset: Challenges and Application.

Author: Futong Han (2660555, 12581135)

1st supervisor: Drona Kandhai
daily supervisor: Behrouz Raftari, Sabiha Majumder
2nd reader:

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

March 29, 2021

"I am the master of my fate, I am the captain of my soul"
from Invictus*, by William Ernest Henley*

Abstract

Here goes the abstract of this thesis.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

1

Introduction

With the rapid development of modern science and technology, different types of data are growing exponential. These increasing data have enabled computer technologies such as artificial intelligence, data mining and machine learning to develop tremendously in various application fields. Among these techniques, the classification of imbalanced data has received widespread attention in different application fields such as disease detection, face recognition, fraud detection, and fault diagnosis.

Imbalanced data refers to classification problems where the samples are not evenly distributed across different classes(1). Take the binary classification problem as an example, if the number of samples in the positive class is much larger than the number of samples in the negative class, then the data in this case is called imbalanced data. And usually we called the class with more samples as the majority class and the class with less samples as the minority class. In most cases, the minority class are more useful and they are our main classification target.

Because of the extremely distribution, there are some problems when we apply the classification algorithms on it. The first problem is when we evaluate the model performance, the accuracy can not show the classification result properly. Due to the small size of minority samples, the accuracy of traditional classifiers is biased towards majority classes, so the classification result can not always be trusted.

The second problem is that there are always not clear dividing line between minority samples and the majority samples. For example, there is a visualization shows the samples distribution in an imbalanced dataset in the Fig1. The brown points are minority samples and the blue points are majority samples and we can see that there are too many minority samples appear in areas where majority samples are dense. The classification algorithms

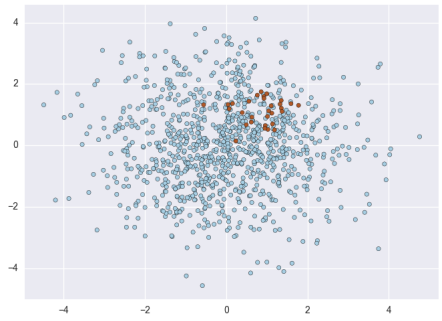

Figure 1.1: Samples distribution in imbalanced data

can not always identify these minority class samples correctly and increase the difficulties of building a model.

1.1 Research question

As we can see, the classification on imbalanced data has a wildly application and there are also different requirements for the classification result in different situations. Here I use a confusion matrix to show the classification result of an imbalanced dataset in Fig 1.2. If a sample is predicted to class 0 but actually it belongs to class 1, we call it an 'escape' sample, and if a sample is predicted to class 1 but actually it belongs to class 0, we call it a 'misjudgement' sample. Both the 'escape' samples and the 'misjudgement' samples will influence the business. Sometimes we want the 'escape' samples as few as possible such as fraud detection or credit risk because all of the fraud transactions will be caught and it will save a lot of money for the company. Sometimes we want less 'misjudgement' samples because it 'misjudgement' samples will waste a lot of time and efforts on it. So here is the research question: Is it possible to have a systematic framework for building a machine learning algorithms on an imbalanced dataset?

In order to make the question much clear, we can also ask some sub questions about it: what should we do when we use the imbalanced data to build a model? What should we do if we want a better classification result according to the requirements on the imbalanced data?

Based on the previous research, most of the papers focus on improving the classification algorithms or creating new resampling algorithms. It is hard to find a clear framework

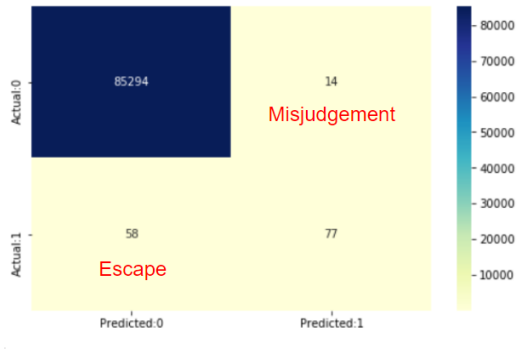

Figure 1.2: Confusion Matrix

combine these techniques together and give a guide or an instruction about which technique is the best choice according to different business requirements. In this thesis, I will focus on having a systematic framework for building a machine learning classification algorithms on an imbalanced dataset. By following the framework, we can develop or evaluate the classification model on imbalanced dataset. Also, I will discuss the application and the challenge about imbalanced data.

The paper is structured as follows:

- Chapter 2 will introduce the background of the research and some of the most common classification algorithms and resampling algorithms on imbalanced data.
- Chapter 3 will introduce the dataset I used in the research and details about data exploration and data preprocessing.
- Chapter 4 will describe the design and implementation process of the systematic framework.
- Chapter 5 will discuss the result of the implementation and compare different algorithms.
- Chapter 6 will draw a conclusion about the framework and also discuss about the future work on it.

2

Background

In recent years researchers used various algorithms and frameworks to solve imbalanced learning problems and I will discuss imbalanced learning research progress from different perspectives.

First, I present the initial statistics for the research trends in imbalanced learning. I collected the conference publications from Springer Link and plot the research trending from 2011 to 2020. Search 'imbalanced learning' in all the conference papers and then write down the number of papers in each year in Fig2.1.

We can see that the number of conference papers about imbalanced learning each year grow from 357 to 1736 and there is a significant growth between 2018 and 2019. This trend shows that imbalanced learning is a popular research topic and a growing number of algorithms and techniques are used to solve it. Focus on the classification of imbalanced dataset, data resampling, feature selection, the improvement of classification algorithms and the way of model evaluation are the most discussed topics.

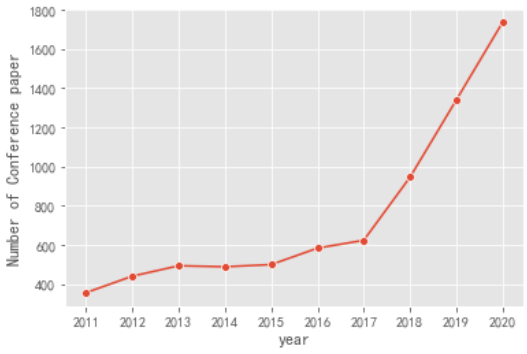

Figure 2.1: Conference paper trending

Second, there are some highly cited papers give a description of the panorama in imbalanced learning. Learning from imbalanced data: open challenges and future directions(2) concentrates on discussing the open issues and challenges in imbalanced learning, such as extreme class imbalance, dealing imbalance in online learning, multi-class imbalanced learning, and semi/unsupervised imbalanced learning. Learning from class-imbalanced data: Review of methods and applications(3) is an exhaustive survey of imbalanced learning methods and applications, a total of 527 papers were included in this study. It provides several detailed taxonomies of existing methods and also the recent trend of this research area. These papers help to find initial way of research.

2.1 Application of imbalanced data classification

As we can see, imbalanced data is a hot topic in the academic field and at the same time, the inaccuracy of imbalanced data classification has also become a very common problem in the industry field. Here I will give some examples about the application of the classification on imbalanced data in the industry field.

2.1.1 Financial Risk Management

Financial risk management is the practice of protecting economic value in a firm by using financial instruments, and it is one of the most widely used fields for imbalanced data classification. For example, in the fraud detection in banking service, most transactions are just normal records and only a very small part of transactions are fraudulent transactions, and it will be a big challenge to recognize the abnormal transaction records. Also, imbalance classification methods are used in credit card default prediction and credit risk. As an efficient and fast payment tool, credit cards are popular all over the world, but with the rapid growth of the number of users, credit risks are also increasing. Defaulting users will bring huge losses to the bank, so it is very important to evaluate user credit and identify default risks in advance. The default users are only a small part of the total number of users, which makes the dataset extreme imbalanced. Furthermore, predicting corporate bankruptcy or management of cash flow also needs the classification of imbalanced data a lot.

2.1.2 Medical science and bioinformatics

Classification of imbalanced data also plays an important role in medical science and bioinformatics. The application often involves building a decision support system to detect

and predict disease and biomedical activities. In the analysis of biomedical data, cancer samples are often limited compared with healthy samples, and the classification method can effectively improve the recognition rate of early cancer. In bioinformatics, protein datasets are usually imbalanced, and protein detection tries to identify new protein structures and functions based on their sequence expressions.

2.1.3 IT and software engineering

With the explosive growth in web data and rapidly development of big data techniques, finding useful information is more and more important in IT industries. Most of the internet companies such as Facebook and Spotify will meet customer retention problems, the vast majority of customers will continue to use the services and only a very small number of customers will not continue to use the services, so it is very important to find the lost customers and make them continue to use the service. Also, imbalanced classification can be used on software quality evaluation, network intrusion detection, software security and so on.

After describing the applications on imbalanced data, I will introduced some common algorithms and techniques which are widely used on imbalanced data. From previous research, I will introduced the related techniques in 4 levels: Resampling algorithms, Feature Selection, Classification algorithms and Model Evaluation.

2.2 Resampling Algorithms

Data preprocessing plays an important role before building a machine learning classifier because it can make the input data clean and improve the classification result. Resampling the data can change the size of different categories, and affect the original data distribution ratio and then the classification results can have a better performance. Resampling method includes oversampling of minority samples, undersampling of majority samples, and mixed sampling.

2.2.1 Oversampling algorithms

For imbalanced datasets, one of the most common ways to alleviate the imbalance is to oversample the minority samples in the dataset. By generating new minority samples, the distribution of imbalanced will be better and one of the most basic oversampling method is Random oversampling. It replicates majority samples randomly and adds them

to the dataset to increase the number of minority samples. However, random oversampling will lead to overfitting and decrease the generalization ability of the classifier. So we will choose other improved oversampling algorithms such as SMOTE algorithm and Borderline-SMOTE algorithm. Following is the introduction of some common oversampling algorithms.

2.2.1.1 SMOTE

SMOTE is a data preprocessing technique applied to imbalanced dataset proposed by Chawla.(4) Different from the simple duplication in random oversampling, SMOTE creates synthetic instances of the minority class by operating in the "feature space" rather than the "data space". First, select each sample x in turn from the minority samples as the root sample for the synthesis of the new sample; secondly, according to the up-sampling magnification n, randomly from the k (k is generally an odd number, such as k=5) neighbor samples of the same category of x Select a sample as the auxiliary sample for synthesizing the new sample, repeat n times; then perform linear interpolation between sample x and each auxiliary sample, and finally generate n synthesized samples.

2.2.1.2 Borderline-SMOTE

Smote is widely used in many application fields but there are still some disadvantages on it. For example, according to the SMOTE algorithm, the synthesis of new samples depends on the choice of root samples and auxiliary samples. If both the root sample and the auxiliary sample are in the minority region, the synthesized new sample is considered reasonable. However, if one of the root sample and the auxiliary sample is a noise sample, the new sample will most likely fall in the majority class area, that is, the new sample will become noise and disturb the correct classification of the dataset. At this time, the new sample is usually treated as an unreasonable sample.

In order to solve this problem, Borderline-SMOTE algorithm proposed by Han(5) improved SMOTE algorithm and it only considered the minority samples distributed near the classification boundary and took them as the root samples. First, the k-NN method is used to divide the minority samples in the original data into three categories: "Safe", "Danger" and "Noise". The "Danger" samples are the samples closed to the classification boundary. According to the principle of SMOTE interpolation, over-sampling of minority

samples belonging to the "Danger" category can increase the minority samples used to determine the classification boundary.

2.2.1.3 ADASYN

The ADASYN(6) algorithm shorts for adaptive syntheticand it adaptively changes the weights of different minority samples according to the distribution of minority samples, automatically determines the number of new samples that each minority sample needs to synthesize, and synthesizes more new samples for more difficult samples to compensate for bias state distribution. On the one hand, adasyn reduces the bias caused by the unbalanced distribution of the original data set. On the other hand, it adaptively transfers the classification decision boundary to samples that are difficult to learn, thereby optimizing the classification effect for minority classes.

2.2.2 Undersampling algorithms

Contrary to the oversampling technique, the undersampling technique starts from the majority class, and it balances the dataset by filtering and deleting part of the majority class samples. The current main undersampling method is random undersampling and it can randomly take some samples from the majority class for undersampling, but it will lose some important sample information useful for classification and then influence the classification result. Based on random undersampling, there are also many improved undersampling algorithms.

2.2.2.1 CNN

CNN stands for condensed nearest neighbour algorithm and it was originally proposed to find consistent subsets. The nearest neighbor algorithm is used to classify the data samples, and the samples with incorrect classification results are added to the subset. This method can retain the majority class samples near the more valuable decision boundary, and also remove the majority class samples far away from the decision boundary. In this way, the majority class samples are reduced to alleviate the imbalance of the dataset.

2.2.2.2 ENN

ENN is short for edit nearest neighbour algorithm and it uses the nearest neighbor algorithm to edit the data set. For a sample, if most of its k-nearest neighbor samples are different from the category it belongs to, we will delete this sample, and those most or

all of the nearest neighbor samples are of the same type as the sample to be tested will be retained In the dataset. Since most of the samples existing near the majority class of samples also belong to the majority class of samples, the ENN algorithm also has certain limitations.

2.2.3 Hybird methods

We have described common over-sampling algorithms and under-sampling algorithms, but both sampling methods still need improvement. Over-sampling may cause data fitting, and under-sampling may cause information loss while deleting samples. Especially in the processing of imbalanced data sets, large differences in the number of samples will significantly reduce the classification effect. For simple under-sampling and over-sampling, they are basically only for a certain class of samples. For example, over-sampling is to expand the number of minority classes, and under-sampling is to delete and simplify the majority of classes. Therefore, some researchs have proposed a hybrid sampling method, which combines oversampling and undersampling to solve the problem of imbalanced sample distribution. The following is some common hybird sampling methods.

2.2.3.1 ROS+RUS

This hybrid sampling methods combine random oversampling and random undersampling together(7).Randomly oversampling the minority samples while randomly under-sampling the majority samples.

2.2.3.2 SMOTE+Tomek links

While using SMOTE algorithms to synthesize minority samples, it will also lead to the increase of minority noise in the dataset of the majority class ,and then it will affect the classification result. In order to solve this problem, we can use Tomek Links(8) as a data cleaning method to decrease the number of noise data. This algorithm first use SMOTE to generate new minority class samples and then use Tomek Links to clean the noise data and classification boundary data.

2.3 Feature Selection

Feature selection refers to the measurement of the importance for features in the original dataset and then choose the subset with highly important features. Feature selection can eliminate irrelevant or redundant features to reduce the number of features,and improve

model accuracy. Also, selecting the highly relevant features can simplify the model and help understand the process of data generation.

According to the degree of contribution of different features to the classification results, we can divided the feature into 4 kinds: Strongly related features, weakly related but not redundant features, weakly related and redundant features, unrelated features and noise features. In order to improve the classification effect, we need to select strong correlation features first, and weak correlation but not redundant features because the strongly related features and weakly related but not redundant features will help improve the classification result and the unrelated features will not help.

The process of feature selection includes four main steps, including subset generation, subset evaluation, stopping criterion, and result validation. The process of generating a subset is a search process, and the candidate feature subset is selected according to a specific search strategy. Each candidate subset will be evaluated according to the evaluation criteria, and if the evaluation result of the new subset is better than the previous one, the selected subset will be updated to the current best one. The generation of subsets and the evaluation of subsets continue to loop until meeting the given stop criterion, and finally the results are verified for the optimal feature subset.

According to the way of feature selection, it can be divided into two categories: Feature selection based on search strategy and feature selection based on evaluation criteria.

2.3.1 Feature selection based on search strategy

The search process needs to consider two basic issues: The first is to determine the starting point of the search. Starting with an empty set, and continuously add features into it, or starting from a complete set, and then remove features out of it. The second is to determine the search strategy. According to the different search strategies, feature selection can be divided into complete search, random search and heuristic search.

2.3.1.1 Complete search

Exhaustive method and branch and bound method are the main methods used in Complete search. The exhaustive method searches each existing feature subset to find and select the optimal feature subset that meets the requirements, such as backtracking methods and variant methods. Since it can traverse all feature sets, it can always find the optimal feature combination in the whole dataset, but when the number of original features is

large, the search space will also be large, then the execution efficiency of the algorithm will decrease. Therefore, the practicality of exhaustive is not strong.

The branch and bound method shortens the search time through pruning. It is also the only method that can obtain the best results in complete search search so far. However, it requires the optimal feature subset to be preset before the search starts and the subset evaluation function must satisfy monotonicity. At the same time, when the dimensionality of the features is too high, the algorithm must be repeated too many times, which greatly limits its application.

2.3.1.2 Random search

This method first selects a subset of features at random, and then uses two methods: The first is to use random factors into the traditional sequence search, which is called a probabilistic random method, for example, the hill climbing algorithm and simulated annealing algorithm. The other is called completely random method, because the generation of candidate subsets is completely random. In these methods, the stochastic process helps to avoid falling into the local optimum in the search space, but whether it can get the optimal result depends on the available resources.

2.3.1.3 Heuristic search

It is an approximate algorithm that considers both the optimality of search and the amount of calculation. It generates the optimal feature subset through the design of reasonable heuristic rules and repeated iterative operations. According to the different initial feature set and search direction, it can be divided into single optimal feature selection, sequence forward selection, sequence backward selection, and two-way selection. The complexity of heuristic search is low ,and the execution efficiency is high, which made it widely used in practical applications. However, in the process of feature selection ,once a feature is selected or deleted, it cannot be withdrawn, and it will easily lead to the generation of partial optimal result.

2.3.2 feature selection based on evaluation criteria

Each generated subset will be evaluated using evaluation criterion and according to the way of evaluation criterion: Filter method, wrapper method, and embedded method.

Filter method is an efficient method that uses appropriate evaluation criteria to judge the quality of features. These evaluation criteria are used to enhance the correlation between

features and classes or weaken the redundancy between different features. At present, Commonly used evaluation criteria are based on distance measurement, information measurement, correlation measurement and consistency measurement. The Filter method can quickly remove some noise features based on the relevant evaluation metrics, and reduce the space for feature search.

The evaluation of Wrapper method for feature subsets depends on the subsequent classification algorithm. It directly uses the selected feature subset to train the classifier, and compare different feature subset according to the classification effect on the test set. Due to the complexity of the classification operation, the training speed will be relatively slow, but the scale of the selected feature subset is relatively small, and the classification accuracy is relatively high.

The embedded method performs the two actions of feature selection and classifier training simultaneously. This method embeds the feature selection process into the training process of the classifier, and the classifier does not need to be trained after the feature selection process ends, so the model after the training also include the result of feature selection. A typical embedded feature selection algorithm is decision tree. Each recursion of the decision tree needs to select a feature to decompose the dataset into smaller subsets.

2.4 Classification Algorithms

The traditional classification algorithms used on imbalanced dataset includes KNN(K Nearest Neighbors), SVM(Support Vector Machine), decision tree and so on. Two main methods for solving imbalanced learning problems in algorithms level are: Algorithmic classifier modifications and ensemble methods. Also both of the methods targeted binary classification problems and I will give a brief introduction to these methods.

2.4.1 Algorithmic classifier method

Improving the learning ability of existing machine learning classification algorithms to improve the classification performance for imbalanced data is another main research direction. Considering that classification algorithms need to have strong interpretability in practical applications, I will mainly introduce decision trees, KNN and SVM algorithms in imbalanced data.

2.4.1.1 Decision Tree

Decision tree is one of the most widely used machine learning algorithms. It uses a top-down recursive method to compare features between nodes and judge the downward branch from the node based on different features. Compared with other machine learning classification algorithms, the decision tree classification algorithm is relatively simple. As long as the training sample set can be represented by feature vectors and categories, the decision tree classification algorithm can be considered. The complexity of the predictive classification algorithm is only related to the number of layers of the decision tree, and the data processing efficiency is very high and robust, so it is also very suitable for the classification of imbalanced data. The modification of decision tree algorithm is a good way to make classification on imbalanced dataset. Krawcyzk(9) introduced an effective ensemble of cost-sensitive decision trees for imbalanced classification. Base classifiers are constructed according to a given cost matrix, but are trained on random feature subspaces to ensure sufficient diversity of the ensemble members.

2.4.1.2 KNN

KNN is a concise and easy-to-understand classification algorithm. Its classification idea is to find the k nearest neighbors of the sample to be tested based on the distance metric, and then analyze the category distribution of the k neighbor samples and divide the sample to into the category it belongs to. When the knn algorithm is applied to an imbalanced data set, due to the small number of minority neighbor samples, it is difficult to predict the potential minority samples correctly. So the modification of KNN algorithms needs to find a better way to get the $k-value$ and improve the performance. When dealing with highly imbalanced data, a salient drawback of existing kNN algorithms is that the class with more frequent samples tends to dominate the neighborhood of a test instance in spite of distance measurements, which leads to suboptimal classification performance on the minority class. To solve this problem, Wei Liu(10) propose CCW (class confidence weights) that uses the probability of attribute values given class labels to weight prototypes in kNN.

2.4.2 Ensemble method

The ensemble method is mainly to form a more powerful classifier method by combining multiple weak classifiers. Integration methods can be divided into two categories: bagging and boosting.

The main representative of bagging is the random forest algorithm. It mainly uses the bootstrap re-sampling method to extract multiple samples from the original sample data, and performs decision tree modeling on each bootstrap sample, and finally combines the prediction results of multiple decision trees , and use voting method to get the final prediction result.

The main representative of boosting is the GBDT algorithm. The GBDT algorithm is also called the gradient boosting decision tree algorithm. It uses the CART decision tree as the base model and uses the forward distribution algorithm for iteration. By continuously updating the sample weight of the training data set, the weight of the correctly divided data item will be reduced, and the weight of the incorrectly divided data item will be increased.

The ensemble learning method focuses on combining a data-level or algorithm-level method with ensemble learning to obtain a powerful ensemble classifier. Because of its excellent performance in unbalanced tasks, ensemble learning is becoming more and more popular in practical applications. Most of them are based on a specific ensemble learning algorithm (for example, Adaptive Boosting(11)) and an other imbalanced learning method (for example, SMOTE), then train them together to get classification result.

SMOTEBoost is a basic ensemble method. It creates synthetic examples from the rare or minority class, thus indirectly changing the updating weights and compensating for skewed distributions(12).

2.5 Model Evaluation

For traditional machine learning classification algorithms, the overall accuracy is usually used as the evaluation indicator. However, the accuracy rate tends to measure the performance of the classification model by the overall classification result, but it is not suitable for measuring the performance of imbalanced datasets. For imbalanced dataset, the accuracy rate can not describe the classification result for the minority class and sometimes a good overall accuracy rate can not grantee the same result for the minority class. In order to take into account the evaluation of minority classes, we need to find evaluation methods that can describe the performance of imbalanced datasets. First, we will talk about the concept of Confusion Matrix(13)in the Fig.

The TP, FP, FN, TN shows the number of different classification results:

- TP: True positive. Predicted positive and it's true

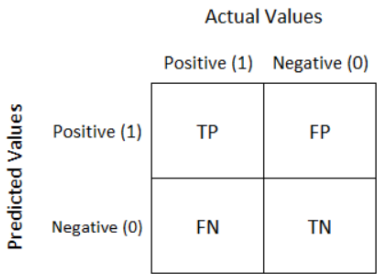

Figure 2.2: Confusion Matrix

- TN: True negative. Predicted negative and it's true.
- FP: False positive. Predicted positive and it's false
- FN: False negative. Predicted negative and it's false.

The number of positive samples P and the number of negative samples N satisfy the formula:

$$P = TP + FN \tag{2.1}$$

$$N = FP + TN \tag{2.2}$$

And we can also use the precision rate, the recall rate, $F - score$ and $G - mean$ to evaluate the model. The recall rate is also known as sensitivity, and it is the fraction of relevant instances that were retrieved. Also it can be viewed as the probability that a relevant document is retrieved by the query. The precision rate is also called positive predictive value, and it is the fraction of relevant instances among the retrieved instances. Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. The equation of the recall rate and precision rate:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

Usually, the recall rate and the precision rate is the higher the better. However, sometimes the recall rate is the opposite of the precision, and the classification result can not make both of them at a high level. Then we need to consider then together using

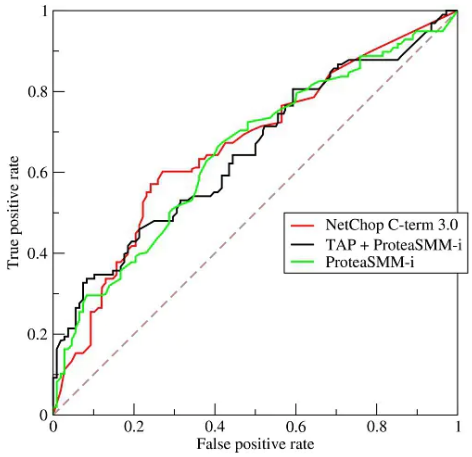

Figure 2.3: ROC

$F-score$(also we call it $F-measure$). The higher $F-score$ means better performance. The equation of $F-score$ shows at follows:

$$F-score=\frac{2\cdot Recall\cdot Precision}{Recall+Precision} \tag{2.5}$$

$G-mean$ is also a common way to measure the overall performance of imbalanced dataset. The equation shows as follows:

$$G-mean=\sqrt{\frac{TP}{TP+FN}\times\frac{TN}{TN+FP}} \tag{2.6}$$

$TP/(TP+FN)$ shows the classification ability for the minority class, and $TN/(TN+FP)$ shows the classification ability for the majority class. The higher $G-mean$ is, the better classification model is. Then we can use $G-mean$ to evaluate the overall performance of the classification result on imbalanced dataset.

Besides the $F-score$ and $G-mean$, ROC and AUC are also used to evaluate the performance of imbalanced datasets. ROC(Receiver Operating Characteristic) is a comprehensive indicator reflecting the sensitivity and specificity of continuous variables. Fig is an example of ROC.

As we can see in this sample graph of the ROC curve, the horizontal axis of the ROC curve is false positive rate (FPR), and the vertical axis is true positive rate (TPR). TPR means the proportion of samples predicted to be positive but actually negative to all negative samples, and FPR means the proportion of samples that are predicted to be positive and actually are positive to all positive samples. For the sample data, the classifier will give the probability for each data being a positive example, and then We can set a threshold

for it. When the probability of a sample being a positive example is greater than this threshold, this sample will be considered as a positive example, if less than considered it as the negative example. Finally we can get a (TPR, FPR) pair, that is, a point on the image. By adjusting the threshold, we can get several points and draw the curve.

AUC is short for Area Under Curve, and it is defined as area under the ROC curve. The meaning of AUC is the probability that when a positive sample and a negative sample are randomly selected, the positive sample will be ranked in front of the negative sample according to the score calculated by the current classifier. The AUC value is used as the evaluation criterion because in many cases the ROC curve does not clearly indicate which classifier performs better, and as a value, a classifier with a larger AUC performs better. From AUC, we can tell if a classifier is good:

- AUC=1: Perfect classifier. When using this predictive model, there is at least one threshold to get a perfect prediction. In most prediction situations, there is no perfect classifier
- $0.5 <$ AUC < 1: Better than random guessing. This classifier (model) can have predictive value if the threshold is properly set.
- AUC $= 0.5$: Like random guessing, the model has no predictive value.
- AUC < 0.5: Worse than random guessing; but as long as you always go against predictions, it is better than random guessing.

The ROC curve has a very good feature: when the distribution of positive and negative samples in the test set changes, the ROC curve can remain unchanged. When the data set is an imbalanced data set, the ROC can better measure the performance of the model.

...

3

Data

The number of open source imbalanced dataset is quite limited and in this thesis I will focus on the dataset in financial risk field. There are some famous datasets for fraud detection and credit risk such as 'Synthetic Financial Dataset for Fraud Detection(14)' published by Norwegian University of Science and Technology and 'Credit Card Fraud Detection(15)' dataset published by Université Libre de Bruxelles Machine Learning Group. Also there are some available open source datasets in different imbalanced ratio from UCI Machine Learning Repository(16) published by University of California Irvine. After comparing different datasets, I will start my research on 'Credit Card Fraud Detection' dataset because of its better data privacy and extremely imbalanced. The difference between different methods will be more obvious on the dataset and we can draw better conclusion on it.

3.1 Data Introduction

'Credit Card Fraud Detection' dataset contains 284,807 transaction records in two days by european cardholders in 2013, and only 492 of them are fraud transaction records. The dataset is extremely imbalanced, and the negative category (fraud) accounts for 0.173% of all transactions.

There are 31 columns in the dataset and in order to protect the privacy of data, all the numerical variables are the result after PCA transformation, and the features are named using V1, V2,...,V28. The features not converted by PCA are 'Time' and 'Amount'. The 'Time' contains the number of seconds between each transaction and the first transaction in the data set, and the 'Amount' shows the transaction amount.

The mean of the 'Amount' is not very big and it is around 88 dollars, and the maximum value of fraud transactions is only 2,125 dollars. The 'Class' is the label and when the

transaction is a fraud transaction the 'Class' is 1, otherwise it is 0. Furthermore, there is no 'Null' value in the dataset, which means we do not need to replace the null value.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...

Figure 3.1: Head of data

3.2 Data Exploration

After a basic introduction of the dataset, I start to explore the data and try to learn from it. First is the distribution of data. We can see the data is extremely imbalanced in Fig3.1. More than 99% of the transaction records are not fraud transactions and only 0.173% of the transactions are the fraud one. If we use this dataset as the basis of our classification model and analysis the result, we may get a lot of errors because the model may overfit, and the model will assume that most transactions are not fraudulent.

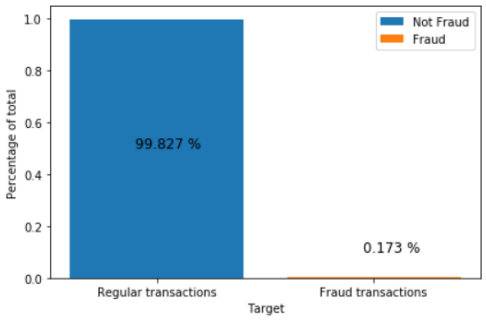

Figure 3.2: Transactions

Also I plot the distribution of 'Time' and 'Amount' in Fig3.2 and Fig3.3. By seeing the distributions we can have an idea how skewed are these features. It is clear that the 'Time' distribution is uneven and the 'Amount' is not huge.

There are also some relationship between fraud transactions, transaction amount and number of transactions. The amount of fraud transactions are scattered and small compared with the amount of normal transactions, which shows that fraud transactions prefer to be small amounts in order not to attract the attention of credit card owners.

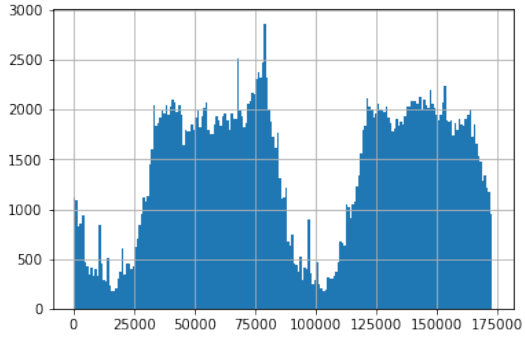

Figure 3.3: Time distribution

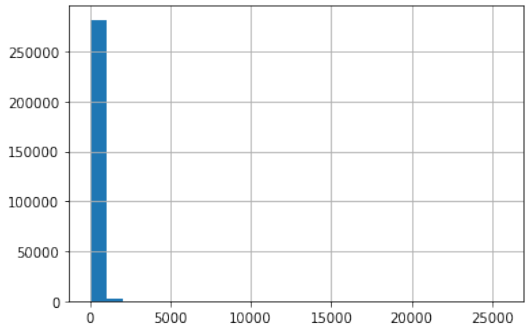

Figure 3.4: Amount distribution

The relationship between fraud transactions, transaction amount and transaction time also shows the when the fraud transactions happen. In the sample of fraud transactions, outliers occurred in the time period when people used credit cards less frequently. The peak of the number of fraud transactions reached 43 at 11 am on the first day, and also we can see that the rest of transactions occurred between 11 p.m. and 9 a.m. Because the fraud transactions will attract the attention of credit card owners less when the credit card owners sleep.

In order to understand the data better, I will use Pearson Correlation Coefficient to explore the correlations between different features. Pearson correlation coefficient is a measure of data similarity. The value in the output range -1 to +1 represents the degree of correlation of the data: 0 means no correlation, negative value is negative correlation, and positive value is positive correlation. The formula of the Pearson correlation coefficient is as follow:

$$R_{xy} = \frac{\sum_{k=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_i - \bar{x})^2 \sum_{k=1}^{n}(y_i - \bar{y})^2}} \tag{3.1}$$

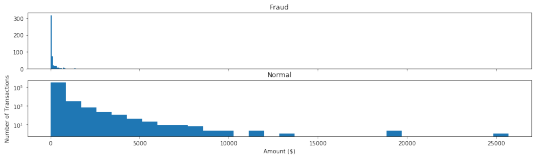

Figure 3.5: relationship between fraud transactions, transaction amount and number of transactions

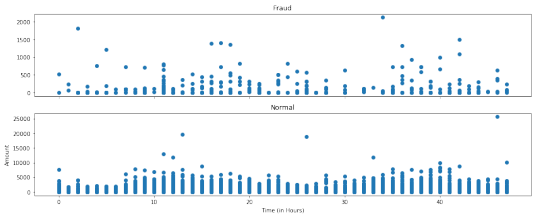

Figure 3.6: relationship between fraud transactions, transaction amount and transaction time

By calculating the Pearson correlation coefficient and plot the coefficient matrices of the features, I can learn if some features largely affect the transaction fraud or not. Below is the correlation matrix of the dataset:

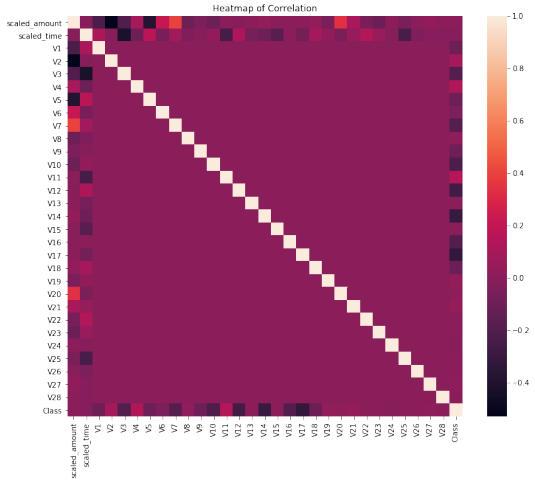

Figure 3.7: correlation matrix of imbalanced data

We can see that the correlations between features are not very strong and most of the pearson correlation coefficient is around 0. This is caused by the extremely imbalance and it also influence the correlations between features. In order to learn about the real correlations, I make a subset to make sure the data in it is evenly distributed and it can influence the feature correlations properly.

The sub-data set is a data set with a 50/50 ratio of positive and negative samples, which means that the number of fraud samples and non-fraud samples are the same. In the above analysis, we have seen that the original dataset is highly imbalanced, and we can not know the correct correlations between labels and features, and between features and features. So I randomly use 492 non-fraud samples and the original 492 fraud samples to form a new subset. Then I calculate the Pearson correlation coefficient again on the subset and plot it in Fig 3.7:

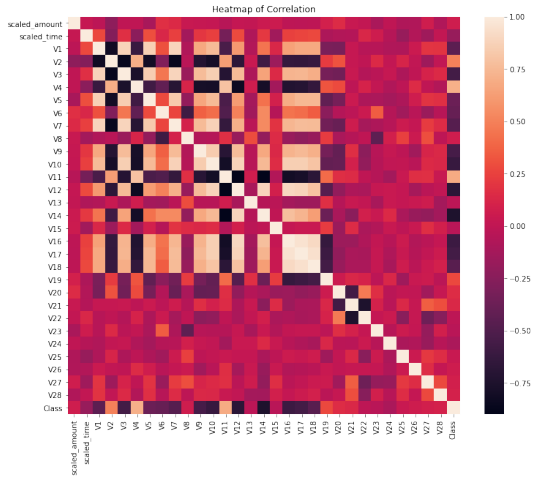

Figure 3.8: correlation matrix of balanced data

We can see that the correlation between features are much more clear on the subset. V17, V14, V12, and V10 are negatively correlated, which means if their value is lower, the final classification result is more likely to be a fraud transaction. V2,V4,V11 and V19 are positively correlated, which means if their value is lower, the final classification result is more likely to be a non-fraud transaction.

Furthermore, I also check the difference between normal transaction records and fraud transaction records in Fig3.8, and we can also find the correlation between features in normal transactions are not clear. At the same time, the correlation between features in Fraud transactions are closed.

3.3 Feature Selection

In the applications of machine learning classification, too many features will cause training time longer and the model will also be more complex. Also, too many features will

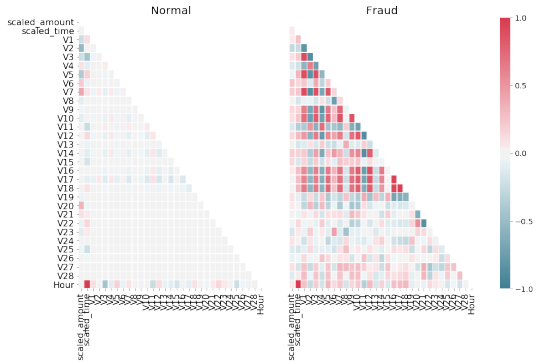

Figure 3.9: correlation matrix difference between Fraud and Non-fraud transactions

lead to sparse features, which will reduce the effect of the model. From the previous data exploration, we can see some of the features in the dataset are weak correlation with the final classification result. So I will select useful features and drop irrelevant, redundant features in the dataset. Dropping redundant features can reduce the number of features, reduce training time, and improve model performance.

Here I use feature selection methods such as information value, random forest and Chi-square to evaluate different features. After calculating the feature importance using different methods, I vote '1' for strong correlations and vote '0' for weak correlations. Then I choose the features with a score higher than 1, and these features are regarded as high important features. The feature importance and voting process are in the figures below.

	Variable_Name	Information_Value	Random_Forest	Recursive_Feature_Elimination	Extra_Trees	Chi_Square	L_One
0	V4	2.478942	0.025897	0.540022	0.040904	62.119086	0.019495
1	V14	2.181399	0.140642	-0.802032	0.124331	33.419519	-0.056643
2	V12	1.943136	0.121750	0.000000	0.109536	29.439713	-0.027667
3	V3	1.788241	0.016108	0.000000	0.036341	7.350719	-0.000203
4	V11	1.736968	0.076855	0.000000	0.073217	70.044292	0.014590
5	V10	1.654638	0.077805	-0.349937	0.077627	12.002423	-0.028755
6	V16	1.309069	0.053207	-0.259138	0.061334	15.467812	-0.028446
7	V2	1.247971	0.008010	0.000000	0.013398	0.696702	-0.006411
8	V17	1.118942	0.175451	0.000000	0.134611	19.820486	-0.044108
9	V9	1.092088	0.031775	-0.138169	0.034623	7.933077	-0.021694
10	V7	1.020535	0.020510	0.000000	0.027048	1.584670	-0.009124
11	V1	0.813381	0.015532	0.000000	0.014790	2.542097	0.008920
12	V21	0.775414	0.017266	0.314886	0.016438	0.082569	0.006301
13	V6	0.595277	0.014490	0.000000	0.014479	0.281381	-0.000524
14	V27	0.560380	0.013474	-0.368351	0.013454	0.036618	0.000000
15	V18	0.559415	0.037393	0.000000	0.042790	14.199656	-0.011720
16	V28	0.538220	0.010356	-0.253888	0.012279	0.003060	0.000000
17	V5	0.475995	0.011244	0.000000	0.016015	0.221278	0.000000
18	V8	0.457128	0.011970	-0.188627	0.011921	0.025812	-0.012198
19	V20	0.431854	0.011840	0.000000	0.013346	0.010586	0.000000
20	V19	0.392858	0.014143	0.000000	0.015938	1.893450	0.001313

Figure 3.10: Feature importance

To make sure the selected features are effective on both imbalanced dataset and the balanced subset, I also repeat the feature selection method on the balanced subset and get

	Variable_Name	Information_Value	Random_Forest	Recursive_Feature_Elimination	Extra_Trees	Chi_Square	L_One	Votes
0	V4	1	1	1	1	1	1	6
1	V14	1	1	1	1	1	1	6
9	V9	1	1	1	1	1	1	6
5	V10	1	1	1	1	1	1	6
6	V16	1	1	1	1	1	1	6
8	V17	1	1	0	1	1	1	5
12	V21	1	1	1	1	0	1	5
10	V7	1	1	0	1	1	1	5
4	V11	1	1	0	1	1	1	5
2	V12	1	1	0	1	1	1	5
11	V1	1	1	0	0	1	1	4
3	V3	1	1	0	1	1	0	4
15	V18	0	1	0	1	1	1	4
7	V2	1	0	0	0	1	1	3
18	V8	0	0	1	0	0	1	2
24	V26	0	1	1	0	0	0	2
20	V19	0	0	0	1	1	0	2

Figure 3.11: Vote selection

another list of chosen features. Then I combine the two chosen feature lists together and get the final 15 features as the result: 'V1','V2','V3','V4','V7','V8','V9','V10','V11','V12','V14' ,'V16','V17','V18','V19'. Using selected features can improve the model performance and speed up the training process.

4

Methodology

As mentioned before the central research question considered in the thesis is: 'Is it possible to have a systematic framework for building machine learning classification algorithms on an imbalanced dataset?'
To be able to answer the question, 4 sub questions are treated:
1. How does the adjustment of threshold influence the recall rate and precision rate in the classification result?
2. When we should use cost-sensitive with the classification algorithms?
3. Which resampling algorithm is the best choice arrording to different requirements such as the best recall rate, best precision rate and best overall F1 score?
4. What is the effect of the combination of different classification algorithms and different sampling algorithms?

The first sub question relates to the threshold adjustment in machine learning algorithms and we want to see the influence of threshold on imbalanced data classification. By lowering the threshold or raising the threshold can change the classification result a lot and usually we have to make a bunch of graphs or repeated training to find a threshold that best suits our needs. The second sub question tackles the importance of class weight in the classification algorithms. Sometimes changing class weight can be an easy method to increase the recall rate or precision rate. The third sub question focus on the performance of different resampling algorithms. With the compare of different resampling algorithms, we can learn which algorithm we should use in different situations. The forth sub question addresses the different choices of classification algorithms and the answer will try to show which one is a proper method.

Model	Precision	Recall	F1 score	Accuracy	ROC
Imbalanced data - Logistic Regression	0.846	0.570	0.681	0.999	0.969

4.1 Baseline-model

First we need to train a baseline model on the imbalanced dataset. The imbalanced dataset after feature selection was split to training set and test set with a ratio of 7 to 3. Based on the training set after feature selection, I build a logistic regression classifier to train the model. And the table below is the classification result on the test set.

The precision rate is 0.846 and the recall rate is 0.570, and we can see that the accuracy rate is 0.999 which means the model is nearly 100 percent accurate if we use the accuracy rate to evaluate the model. However, the F1 score is only 0.681 and it shows that the accuracy rate is useless when we evalute a model based on the imbalanced dataset.

4.2 Resampling algorithms

4.3 classification algorithms

4.4 Model Evaluation

5

Result and Discussion

6

Conclusion

Appendix

References

[1] H. He and E. A. Garcia. **Learning from Imbalanced Data**. *IEEE Transactions on Knowledge and Data Engineering*, **21**(9):1263–1284, 2009. 1

[2] Bartosz Krawczyk. **Learning from imbalanced data: open challenges and future directions**. *Progress in Artificial Intelligence*, **5**(4):2192–6360, 2016. 6

[3] Guo Haixiang, Yijing Li, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Bing Gong. **Learning from class-imbalanced data: Review of methods and applications**. *Expert Systems with Applications*, **73**, 12 2016. 6

[4] Nitesh V. Chawla. **SMOTE: Synthetic Minority Over-sampling Technique**. *Journal of Artificial Intelligence Research*, (16):321–357, 2002. 8

[5] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. **Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning**. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *Advances in Intelligent Computing*, pages 878–887, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 8

[6] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. **ADASYN: Adaptive synthetic sampling approach for imbalanced learning**. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, 2008. 9

[7] C. Seiffert, T. M. Khoshgoftaar, and J. Van Hulse. **Hybrid sampling for imbalanced data**. In *2008 IEEE International Conference on Information Reuse and Integration*, pages 202–207, 2008. 10

[8] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. **A Study of the Behavior of Several Methods for Balancing Ma-**

chine Learning Training Data. *SIGKDD Explor. Newsl.*, **6**(1):20–29, June 2004. 10

[9] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. **Cost-sensitive decision tree ensembles for effective imbalanced classification**. *Applied Soft Computing*, **14**:554 – 562, 2014. 14

[10] Wei Liu and Sanjay Chawla. **Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets**. In Joshua Zhexue Huang, Longbing Cao, and Jaideep Srivastava, editors, *Advances in Knowledge Discovery and Data Mining*, pages 345–356, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 14

[11] Yoav Freund and Robert E Schapire. **A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting**. *Journal of Computer and System Sciences*, **55**(1):119 – 139, 1997. 15

[12] Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. **SMOTEBoost: Improving Prediction of the Minority Class in Boosting**. In Nada Lavrač, Dragan Gamberger, Ljupčo Todorovski, and Hendrik Blockeel, editors, *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 15

[13] Hongyu Guo and Herna L. Viktor. **Learning from Imbalanced Data Sets with Boosting and Data Generation: The DataBoost-IM Approach**. *SIGKDD Explor. Newsl.*, **6**(1):30–39, June 2004. 15

[14] A. Elmir E. Lopez-Rojas and S. Axelsson. **Paysim: A financial mobile money simulator for fraud detection**. page 249–255. 28th European Modeling and Simulation Symposium, EMSS, 2016. 19

[15] Andrea Dal Pozzolo, Olivier Caelen, Reid Johnson, and Gianluca Bontempi. **Calibrating Probability with Undersampling for Unbalanced Classification**. 12 2015. 19

[16] Dheeru Dua and Casey Graff. **UCI Machine Learning Repository**, 2017. 19