MACHINE LEARNING FOR SHILL BIDDING CLASSIFICATION MODELS

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfilment of the Requirements
For the Degree of
Doctor of Philosophy
In
Computer Science
University of Regina

By
Ahmad Atea Alzahrani
Regina, Saskatchewan
September, 2019

Copyright 2019: Ahmad A. Alzahrani
Ahmad Atea Alzahrani, candidate for the degree of Doctor of Philosophy in Computer Science, has presented a thesis titled, *Machine Learning for Shill Bidding Classification Models*, in an oral examination held on August 22, 2019. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: Dr. Issa Traoré, University of Victoria*

Supervisor: Dr. Samira Sadaoui, Department of Computer Science

Committee Member: Dr. Lisa Fan, Department of Computer Science

Committee Member: Dr. Irfan Al-Anbagi, Faculty of Engineering & Applied Science

Committee Member: Dr. Habib Louafi, Department of Computer Science

Chair of Defense: Dr. Fanhua (Bill) Zeng, Faculty of Graduate Studies & Research

*Present via Skype
ABSTRACT

As online auctions become more prevalent worldwide, they are increasingly targeted by various types of cyber-crimes. In-auction fraud, such as shill bidding (SB), is considered the most challenging to detect. SB has been recognized as the predominant form of online auction fraud. It is difficult to identify due to its similarity to normal bidding behavior. The complexity of finding and defining SB patterns makes it resistant to discovery. Also, the unavailability of SB datasets that are based on actual e-auctions makes the development of SB detection and classification models challenging. Therefore, the prerequisite task that is necessary to perform, in order to achieve our goals in this work, is to scrape a large number of eBay auctions of a popular product, which we did successfully. After preprocessing the raw data which is a very difficult and time consuming operation, we build a high-quality SB dataset based on reliable SB strategies. One of our goals is to share the SB dataset with other researchers, to provide them with an opportunity to test their prediction models based on real fraud data.

Labeling multi-dimensional data is an essential yet difficult task in machine learning, since the classification quality relies mainly on the quality of the data labels. In the generated SB dataset, a record defines the behaviour of each bidder in each auction, yet the records are not classified. To implement robust binary classification models, the training instances must be efficiently categorized. So, another aim of this study is to create a labelled SB dataset for SB detection systems based on
classification techniques. The capabilities of hierarchical clustering algorithms, such as CURE, have been proven to be outstanding for isolating similar instances in a group. Thus, we introduce a systematical labelling procedure based on CURE to distinguish suspicious bidders’ behaviour from the behaviour of normal bidders. The experimental outcomes are very satisfactory, which indicates that clustering provides excellent results. However, the labeled SB dataset is imbalanced. Class imbalance is a serious issue that has been comprehensively studied, yet, the experimental results obtained and researchers’ views differ on how to handle this issue. Some researchers prefer data level techniques, while others prefer the algorithmic level. To overcome the problem of imbalanced SB datasets, we investigate the most common techniques used at the data level and the algorithmic level, which are over- and under-sampling and cost-sensitive learning (CSL), respectively.

An auction system can be viewed as a data stream application since thousands of auctions and bids occur daily. Instance-incremental learning is known to be useful in this type of application, since the model normalization is modified based on the arrival of new instances for prediction. In this research, the feasibility of instance-incremental classification is investigated, where the selected lazy classifiers are Locally Weighted Learning, K*instance based, and K-Nearest Neighbours. Additionally, we consider the Hoeffding Tree classifier, since it is also based on instance-incremental learning. We developed several instance-incremental SB classifiers using data sampling and CSL. According to the experimental results, incremental classification returns a high performance for both over- and under-sampled SB datasets. However, over-sampling slightly outperforms under-sampling for both normal and suspicious classes across all four classifiers and quality metrics. Moreover, the predictions on instance-based algorithms combined with CSL, also, return a high accuracy. We note that data sampling is slightly superior to CSL for the suspicious class, in general.
Acknowledgement

(O, Allah! All of the Al-Hamd is due to You, You own all the ownership, all types of good are in Your Hand, and all affairs belong to You. O Lord, to You is praise as befits the Glory of Your Face and the greatness of Your Might.)

(He who does not thank the people is not thankful to Allah.)

Prophet Mahammed Peace be upon Him

To my supervisor Dr. Samira Sadaoui, I would like to express my sincere gratitude for her help, comments, support, valuable advice, and friendship. It was a great honor to work under her supervision and to learn from her experience in the field of fraud detection. Without her inspirational guidance, continuous encouragement, great effort, and selfless help, this work would not have been completed.

My special thanks go to the committee members, Dr. Habib Louafi, Dr. Lisa Fan, and Dr. Irfan Al-Anbagi, for agreeing to be a part of the examining committee and taking the time to read my thesis. Also, I appreciate all the support I received from the Faculty of Graduate Sciences and Research and the Department of Computer Science.

I would like to express my deepest regards for all the valuable advice and support I received from my friends especially, Abdulaziz Al-Hubaishi, Abdulrahman Al-Dhahiri, and Saleh Al-Balwi. Finally, I would like to acknowledge the generous financial support and scholarship provided by the Ministry of Education and Umm Al-Qura University in Saudi Arabia.
Dedication

I dedicate this work to my beloved parents, who always believed in me and continuously pray for me. Their love, unlimited support, and courage have always inspired me and kept me standing to step over the difficulties. With my sincere, humble, and heartfelt acknowledgment, I dedicate this achievement to them for their patience, care, prayer, wisdom, and great love.

Also, this work is dedicated to my lovely wife, Salihah Al-Ghamdi, who takes care of the family and assures our convenience with all love, patience, determination and persistence. She lovingly stood beside me and tolerated me under all different circumstances. Thank you so much for being in my life.

To God’s gift, my daughter Nawal, my older son Moath (God rest his soul), and my junior son Moath, you have been the source of our family joy and happiness. We are very thankful to Allah for having you. You are our family pride.

To my amazing siblings, Sadiyah, Ali, Sa’ad, Nawal, Rahmah, Sharifah, Muhammed, and Hind, I would like to show my sincere gratitude to all of them for their support and prayers. My heartfelt love and respects go to my brothers-in-law Ahmad Mussedd Alomari, and Salem Ibrahim Alomari for their support, and care. Also, my wholehearted love goes to my beautiful nieces and nephews. May All protect them all.

I’m very grateful for many people around me, including my aunts, uncles, and my cousins. Finally, special thanks goes to Abdullah M. Alomari, Hassan A. Alkiri, Eisa Alanazi and other many friends.
# TABLE OF CONTENTS

ABSTRACT

ACKNOWLEDGEMENT

DEDICATION

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

1 Introduction
  1.1 Problem Statement ........................................ 1
  1.2 Motivations .................................................. 4
  1.3 Objectives .................................................... 5
  1.4 Contributions ................................................. 5
  1.5 Thesis Organization ......................................... 10

2 Scraping and Preprocessing Commercial Auction Data for Fraud Classification
  2.1 Introduction .................................................. 13
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.2</td>
<td>3-Days Subset</td>
<td>48</td>
</tr>
<tr>
<td>3.4.3</td>
<td>5-Days Subset</td>
<td>49</td>
</tr>
<tr>
<td>3.4.4</td>
<td>10-Days Subset</td>
<td>50</td>
</tr>
<tr>
<td>3.5</td>
<td>Labeling SB Data</td>
<td>51</td>
</tr>
<tr>
<td>3.6</td>
<td>Conclusion</td>
<td>55</td>
</tr>
<tr>
<td>4</td>
<td>Instance-Incremental Classification of Imbalanced SB Data</td>
<td>56</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>57</td>
</tr>
<tr>
<td>4.2</td>
<td>Related Work</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Sampling SB Data</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Instance-Incremental Classification</td>
<td>62</td>
</tr>
<tr>
<td>4.5</td>
<td>Classifying SB Data</td>
<td>65</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Locally Weighted Learning</td>
<td>65</td>
</tr>
<tr>
<td>4.5.2</td>
<td>KStar Instance-Based</td>
<td>68</td>
</tr>
<tr>
<td>4.5.3</td>
<td>K-Nearest Neighbors</td>
<td>70</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Hoeffding Tree</td>
<td>73</td>
</tr>
<tr>
<td>4.6</td>
<td>Discussion</td>
<td>75</td>
</tr>
<tr>
<td>4.7</td>
<td>Conclusion</td>
<td>77</td>
</tr>
<tr>
<td>5</td>
<td>CSL-based on Instance-Incremental Classification</td>
<td>78</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>79</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Research Scope</td>
<td>79</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Contributions</td>
<td>80</td>
</tr>
<tr>
<td>5.2</td>
<td>Related Work</td>
<td>82</td>
</tr>
<tr>
<td>5.3</td>
<td>CSL for SB Classification</td>
<td>86</td>
</tr>
<tr>
<td>5.3.1</td>
<td>An Overview on CSL</td>
<td>86</td>
</tr>
<tr>
<td>5.3.2</td>
<td>CSL Setup</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>CSL and Instance-based Classification</td>
<td>89</td>
</tr>
</tbody>
</table>
5.4.1 Cost-Sensitive LWL ................................................. 89
5.4.2 Cost-Sensitive KStar ............................................. 90
5.4.3 Cost-Sensitive KNN .............................................. 91
5.4.4 Cost-Sensitive HT ................................................ 92
5.4.5 Discussion ......................................................... 94

5.5 Data Sampling vs. CSL ............................................. 95
5.5.1 Comparison ....................................................... 95
5.5.2 Discussion ....................................................... 101

5.6 Conclusion ......................................................... 101

6 Conclusion and Future Work ...................................... 103
6.1 Summary .............................................................. 103
6.2 Future Work ........................................................ 106

REFERENCES .......................................................... 108

Appendix A SB Metrics in MSSQL & Python ........................ 119

Appendix B Tools and Libraries ....................................... 126
# LIST OF FIGURES

1.1 Thesis Overview and Contributions .............................................. 9

2.1 High-Quality SB Dataset Construction and Classification ..................... 35

3.1 Labeling Process of SB Training Data .............................................. 38

3.2 Overview of CURE [GRS01] .......................................................... 42

3.3 CURE Clustering Algorithm Procedures Description ................................. 43

3.4 Optimal number of clusters for 7 day bidding duration. Silhouette score is examined 19 times. We show the top Silhouette scores. .................... 45

3.5 Optimal number of clusters for 1 day bidding duration .......................... 47

3.6 Optimal number of clusters for 3 day bidding duration .......................... 48

3.7 Optimal number of clusters for 5 day bidding duration .......................... 49

3.8 Optimal number of clusters for 10 day bidding duration ........................ 50

3.9 Decision line of a subset and its labeled clusters ................................ 54

3.10 High-Quality SB Dataset Construction and Classification ....................... 55

4.1 ROC of LWL on over- and under-sampled dataset ................................. 67

4.2 ROC of KStar on over- and under-sampled dataset ............................... 70

4.3 ROC of KNN on over- and under-sampled dataset ................................ 72

4.4 ROC of HT on over- and under-sampled dataset .................................. 75

4.5 Instance-incremental classifiers performance over the sampled datasets .......... 76

5.1 Performance of LWL on Imbalanced Data Techniques ............................ 97
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>KStar Performance on imbalanced data techniques</td>
<td>98</td>
</tr>
<tr>
<td>5.3</td>
<td>KNN Performance over imbalanced data techniques</td>
<td>99</td>
</tr>
<tr>
<td>5.4</td>
<td>HT performance over the handling imbalanced data techniques</td>
<td>100</td>
</tr>
</tbody>
</table>
LIST OF TABLES

2.1 Number of Auctions for Each Duration .......................... 19
2.2 Statistics of iPhone 7 Auctions ................................. 20
2.3 Combining Date and Time into a Single Attribute ............... 22
2.4 Attribute Reformatting .............................................. 23
2.5 Auction Attributes for Shill Bidding .............................. 24
2.6 Properties and Weights of SB Patterns ............................ 29
2.7 Statistics of SB Dataset .............................................. 32
2.8 SB Patterns in the Auction Dataset ................................ 33

3.1 SB Dataset Partitioning According to Bidding Durations ....... 44
3.2 CURE Clustering for 7-day Subset on 8 Clusters with different RPs and $\alpha$ ......................................................... 46
3.3 CURE Clustering for 1-day Subset on 7 Clusters with different RPs and $\alpha$ ......................................................... 47
3.4 CURE Clustering for 3-days Subset on 7 Clusters with different RPs and $\alpha$ ......................................................... 48
3.5 CURE Clustering for 5-days Subset on 5 Clusters with different RPs and $\alpha$ ......................................................... 49
3.6 CURE Clustering for 10-days Subset on 2 Clusters with different RPs and $\alpha$ ......................................................... 51
3.7 Optimal Number of Clusters and Optimal CURE Parameters .... 51
3.8 SB Patterns’ Means and Standard Deviations for Each SB Dataset . . 52
3.9 SB Instances and their Labels . . . . . . . . . . . . . . . . 54
3.10 Final Results of Instance Labeling . . . . . . . . . . . . . . . . . 54
4.1 Imbalanced vs. Sampled SB Datasets . . . . . . . . . . . . . . . . 61
4.2 Instance-Incremental Classifiers Settings . . . . . . . . . . . . . . 65
4.3 LWL Performance on Sampled SB Datasets . . . . . . . . . . . . 67
4.4 Data Sampling Confusion Matrix Produced by LWL . . . . . . . . 68
4.5 KStar Performance on Sampled SB Datasets . . . . . . . . . . . . 69
4.6 Data Sampling Confusion Matrix Produced by KStar . . . . . . . . 69
4.7 KNN Performance on Sampled SB Datasets . . . . . . . . . . . . 71
4.8 Data Sampling Confusion Matrix Produced by KNN . . . . . . . . 72
4.9 HT Performance on Sampled SB Datasets . . . . . . . . . . . . . 74
4.10 Data Sampling Confusion Matrix Produced by HoeffdingTree . . . . 75
5.1 Cost Matrix in CSL . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
5.2 Performance of CSL+LWL where N and S denote normal and suspi-
   cious respectively . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
5.3 Confusion Matrix of CSL+LWL . . . . . . . . . . . . . . . . . . . 90
5.4 Performance of CSL+KStar . . . . . . . . . . . . . . . . . . . . . . 91
5.5 Confusion Matrix of CSL+KStar . . . . . . . . . . . . . . . . . . . 91
5.6 Performance of CSL+KNN . . . . . . . . . . . . . . . . . . . . . . . 92
5.7 Confusion Matrix of CSL+KNN . . . . . . . . . . . . . . . . . . . . 92
5.8 Performance of CSL+HT . . . . . . . . . . . . . . . . . . . . . . . . 93
5.9 Confusion Matrix of CSL+HT . . . . . . . . . . . . . . . . . . . . . 93
5.10 Confusion Matrix of CSL+Classifiers . . . . . . . . . . . . . . . . 94
5.11 CSL+Classifiers Performances on Suspicious Class . . . . . . . . . 94
5.12 LWL: Data Sampling vs. CSL on Suspicious Class . . . . . . . . . 96
ABBREVIATIONS

ML       Machine Learning
SB       Shill Bidding
BT       Bidder Tendency
EB       Early Bidding
BR       Bidding Ratio
LB       Last Bidding
ASP      Auction Starting Price
SOB      Successive Out Bidding
WR       Winning Ratio
AB       Auction Bids
B        Participated Bidder
S        Auction Seller
A        Targeted Auction
CURE     Clustering Using REpresentatives
RPs      Representative Points
Cl.#     The number of targeted cluster
STD      Standard Deviation
Avg STDs The average of the collected Standard Deviations
Avg Means The average of the collected means
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL</td>
<td>Decision Line</td>
</tr>
<tr>
<td>SMOTE</td>
<td>Synthetic Minority Over-sampling TEchnique</td>
</tr>
<tr>
<td>KNN</td>
<td>$k$-Nearest Neighbor</td>
</tr>
<tr>
<td>TPR</td>
<td>True Positive Rate</td>
</tr>
<tr>
<td>FPR</td>
<td>False Positive Rate</td>
</tr>
<tr>
<td>F1</td>
<td>F-Score</td>
</tr>
<tr>
<td>MCC</td>
<td>Matthews Correlation Coefficient</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic</td>
</tr>
<tr>
<td>PRC</td>
<td>Precision/Recall curve</td>
</tr>
<tr>
<td>LWL</td>
<td>Locally Weighted Learning</td>
</tr>
<tr>
<td>IBK</td>
<td>The classifier represents KNN in WEKA</td>
</tr>
<tr>
<td>VFDT</td>
<td>The Hoeffding Tree classifier</td>
</tr>
<tr>
<td>CSL</td>
<td>Cost-Sensitive Learning</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Problem Statement

During the last three decades, the exchange of goods and services over the Web has increased significantly. According to the World Trade Organization, the merchandise sold online globally between 1995 and 2014 was worth over $18 billion \(^1\). With technology in hand, people’s lifestyles are moving faster in various regards, such as communications, advertisements, education, etc. One of the most critical aspects is commercial agreements over the Internet, such as online auctions. Buying and selling products and services through online auctions is convenient for many people, since it saves time and effort. However, this creates an attractive environment for fraudsters to carry out their suspicious activities, due to the vulnerability of online auctions to cyber crimes. As stated in [CC11], auction fraud has persisted since 2004 as one of the top two cyber crimes.

The harm caused by malicious moneymakers targeting online auctions not only affects the victims (genuine bidders) but the online auction businesses and the authorities as well. Regarding the victims who are regular users, thousands of complaints have been filed with the authorities and millions of dollars have been lost. As the

federal Internet Crime Complaint Center (IC3) stated, 9,847 victims lost $11 million due to online auction fraud in 2014. Also, the IC3 reported that 5% of the Internet complaints were associated with automobile auctions scams, where $51 million was lost in 2013. This undoubtedly frustrates the authorities, as a vast amount of work is required to track down the fraudsters. On the business side, if the number of victims keeps increasing, online auctions will be abandoned and will eventually go out of business. Thus, to protect society and provide safe online auctions, it is important for e-auction administrators to employ and support researchers who investigate, understand, and handle the methods, tricks, and security bugs that fraudsters use to engage in their criminal activities.

Typically, fraud operations related to online auctions fall under three categories: 1) pre-auction fraud, where the fraud is conducted before the auction starts; 2) in-auction fraud performed during the auction bidding; 3) post-auction fraud done after the auction is completed (see more details in chapter 2). The second category of fraud is considered the most challenging to handle, since bidders are unaware that they have been victimized. For example, Shill Bidding (SB) is a form of in-auction fraud where a bidder keeps outbidding others to inflate an item’s price, in order to maximize the seller’s profits. In addition, as the number of bids increases, regular bidders are deceived that the item is desirable. As a result, many buyers end up paying more than what a product is actually worth. Unfortunately, SB is difficult to discover at the right time, since it looks similar to normal bidding. After the payment is processed and the item has been received, then the buyer may find that she or he has been duped, but it is too late.

Online auction systems are complex and involve highly sophisticated security techniques that control the systems’ users such as sellers, bidders, banks inquiries, and processing online payments transactions. Nevertheless, various of fraud activities

---

could be committed such as inflating the products prices, selling misrepresented items, non-delivered of purchased items, and non-payment received for delivered item. Systems like feedback reputation, fraud awareness (fraud prevention system), and user authentication are not enough to deter scam attempts, due to the computational complexity of these types of fraud (such as SB).

The scope of this work is about fraud in online auctions. More precisely, we concentrate on the SB fraud that is widely associated with online auctions, and which is exceedingly burdensome to systematically detect, due to its similarity to genuine bidding behaviour. To implement an effective and robust e-auction system that protects actual bidders from being deceived by SB activities, the following points must be adequately addressed:

- Analyze sellers’ and bidders’ behaviours to identify the SB patterns.
- Recognize fraudsters and their various ways of committing fraud, in order to know how to deal with it.
- Define the most effective SB metrics for measuring the SB behaviour.
- Handle a huge volume of bidding transactions, which include bidder name, auction ID, bid amount, a bid time, number of bids, etc.
- Determine the most feasible ML methods to help developers build fraud detection systems, such as the best classification algorithm for detecting SB.
- Develop an ability to systematically learn from existing fraudulent characteristics data.
- Extract and preprocess actual data with a relatively large number of bids to effectively handle and manage the above-mentioned points.
These points will be covered throughout the thesis chapters, as this research seeks to gain knowledge about how to face the problem of SB, including how to protect regular users and e-auction systems from it.

1.2 Motivations

The usefulness of Machine Learning (ML) has been proven over the years in many applications, such as intrusion detection in network systems, cancerous tumours discovery, facial recognition systems, stock market analysis, and credit card fraud identification. An SB detection system is one of the most critical detection systems that has yet to be successfully integrated into e-auction platform, due to the difficulties surrounding this system’s implementation, such as the challenge of computing the SB metrics that identify bidder behaviour. Additionally, the lack of labelled SB datasets based on actual auction data makes the process of building SB detection systems more challenging. SB detection is a very critical and challenging problem that requires a great deal of work for it to be sufficiently addressed. Other challenges considered in this work are how to generate a well-presented SB dataset based on real online auction data and how to correctly label the bidding behaviour in the dataset. We have read many papers that prefer one ML algorithm over another. In this work, we mainly focus on instance-incremental classification algorithms, since they have been recommended for data-stream mining applications such as ATM transactions, phone communications, and network traffic. It is important to examine the feasibility of applying different types of instance-incremental classifiers to online auction data, since the bidding in online auctions is also considered a data stream. Finally, handling imbalanced data is another critical topic in ML and DM that is investigated in this thesis.
1.3 Objectives

The objective of this research is to use ML in online auctions to determine user behaviour. More specifically, we would like to employ instance-incremental learning to recognize the auctions, sellers, and bidders involved in SB. These classification methods will allow us to systematically base fraud detections models on precise, well-proven SB patterns. In 2017, we collected data from the most popular e-auction site that is eBay to create a high-quality SB dataset; to do so, the extracted raw data were hard-loaded during preprocessing, in order to obtain clean, useful auction data. One of the most critical phases when building classification models is labelling multi-dimensional data. Indeed, classification quality relies on how accurately labelled the instances are in the dataset. Moreover, there is no available training SB datasets; thus, one of the research goals is to create a high-quality labelled SB dataset and make it public for researchers and developers from the same domain. Since the lazy learning classifiers are claimed to be useful for data streaming mining, we investigate the feasibility of some of these classifiers by applying them to our labelled SB dataset. In addition, one of our objectives is to find out which technique for handling imbalanced data is the most suitable for the imbalanced SB dataset. Therefore, we examine the most practical handling imbalanced techniques (on both data and algorithm levels) for our case study. These techniques are integrated with the instance-based classifiers to build the detection models based on the produced SB dataset. Finally, we perform a comprehensive comparison between the experimental results.

1.4 Contributions

This research aims to address the issue of SB fraud in online auctions and to develop detection models that are based on ML. As demonstrated in Figure 1.1, this work makes five key contributions. We have successfully extracted actual and recent online
auction data for about three months (end of March to beginning of June, 2017). In accordance with the nature of raw data, the extracted data was a mess. There were over a half-million bids, yet, unfortunately, there were many duplicate bids, missing values, and inconsistent bidding transactions records. Thus, our first contribution in this work is to obtain a cleaned, organized, processable online auction dataset. In binary classification, each record presented in a dataset is classified into a specific class. This is not the case with the bidders in online auction datasets, where a bidder may participate more than once in an auction; thus, we cannot classify each individual bid transaction. To overcome this problem, instead of looking at each bid in each auction in the cleaned auction dataset, we study the behaviour of each bidder in each auction in that dataset. This is done by computing eight different SB patterns that efficiently describe bidder and seller behaviours. Thus, we generate a new high-quality SB dataset that accurately defines the behaviour of each participating bidder in each auction. The first and second contributions are presented in chapter 2 and, also, published in the following paper.


The produced high-quality SB dataset is not labelled. As it is known, labelling multi-dimensional data is a critical step in ML, since classification quality relies on the quality of labelling instances in datasets. Thus, our third contribution in this research is the implementation of a systematic labelling strategy that precisely defines the class of each participating bidder in each auction, where a bidder is categorized as normal (0) or suspicious (1), as depicted in Figure 1.1. As a result, we create a labelled, high-quality SB dataset, which is publicly available at UCI to help other researchers in our field. The third contribution is exhibited in chapter 3, and can be found in the following conference paper.

As it is common in most real-world datasets, the labelled training dataset is moderately imbalanced, as the number of suspicious instances is far lower than the number of normal instances. The recorded imbalanced ratio is about 1:8. In traditional supervised classification, the classification of imbalanced data is one of the most challenging complications that must be solved. One technique for handling imbalanced data that has been suggested is data sampling, where the number of instances in each class is modified to create a new balanced dataset. These techniques are considered in this work in order to determine their feasibility with regards to our labelled dataset.

On the other hand, there are many classification algorithms that have been proven as useful in real-world applications. In this research, we focus on instance-incremental learning, as online auction data are viewed as streaming data that lazy classification has been proven to be able to handle. We compare several instance-based classifications on sampled data to determine the most optimal SB classification model. Therefore, our fourth contribution is comprehensively comparing different data sampling techniques, combined with different instance-incremental classifiers, as illustrated in Figure 1.1. The fourth contribution is given in chapter 4 and, also, published in the following paper.


Moreover, the handling of imbalanced SB datasets is investigated by applying Cost-Sensitive Learning (CSL), which is an algorithmic approach that assigns cost for class of instances. We examine CSL to the original dataset, where an instance-
incremental classifier is used as the base classifier for every experiment. This work consists of two parts. First, examining CSL based on instance-based classifiers using the original SB dataset to find the best CSL algorithm. Second, comparing data sampling with CSL approach on the selected instance-based classifiers to look for the optimal fraud classifier to be used in real-life scenarios in online auctions. In chapter 5, the fifth contribution is presented and has been submitted to the following journal paper.

Figure 1.1: Thesis Overview and Contributions
1.5 Thesis Organization

The thesis consists of six chapters where the related work, backgrounds, and methods’ descriptions are fully covered in their main chapter. The rest of the thesis is organized as follows.

Chapter 2: In this chapter, we demonstrate the auctions frauds in general, the SB fraud, the difficulties of obtaining SB dataset, and the generation of SB data. Then, we review past studies on crawling auctions data. In details, we explain how we extracted raw in-auction data. Also, the preprocessing steps of eBay auction data are described. The characteristics and metrics of SB patterns are intensively explored. Next, we present the generation of SB data from bidding data. Finally, the findings of this study are summarized, and essential directions for the next chapter are highlighted.

Chapter 3: In chapter 3, we discuss related work on data clustering in the context of online auctions. Then, we show how to employ hierarchical clustering, CURE, according to different bidding durations. The properties of other SB subsets are illustrated in details. Next, we describe the systematical labelling approach to assign labels for each cluster of bidders. Finally, we summarize the results of the study and highlights the research directions for the next chapter.

Chapter 4: Past studies on handling imbalanced data problem are viewed. The sampling techniques are implemented to the original SB dataset. Then, we demonstrate four instance-incremental classifiers as well as their quality metrics. Next, we examine the performance of each instance-based classifier on the re-balanced datasets and compare the four classifiers. The findings obtained from the experiments are disscussed. Finally, we present the conclusion of this study and highlight essential research directions as well.

Chapter 5: We exhibit past studies on handling the imbalanced data problem and utilizing CSL. Then, we give an overview of CSL and how to customize it in
the content of SB fraud. The four examined instance-incremental classifiers that are implemented in CSL are described. Next, the performance of each instance-based classifier is examined and compared with the four classifiers. We show a detailed comparison between the two approaches (data sampling and CSL) on the suspicious class. Finally, we illustrate the findings of this chapter.

Chapter 6: The final chapter draws the final conclusions of the thesis and discusses directions for future research.
Chapter 2

Scraping and Preprocessing

Commercial Auction Data for Fraud Classification

Abstract: In the last three decades, we have seen a significant increase in trading goods and services through online auctions. However, this business created an attractive environment for malicious moneymakers who can commit different types of fraud activities, such as SB. The latter is predominant across many auctions but this type of fraud is difficult to detect due to its similarity to normal bidding behaviour. The unavailability of SB datasets makes the development of SB classification models burdensome. Furthermore, to implement efficient SB detection models, we should produce SB data from actual auctions of commercial sites. In this chapter, we first scraped a large number of eBay auctions of a popular product. After preprocessing the raw auction data, we build a high quality SB dataset based on the most reliable SB strategies. The aim of our study is to share the preprocessed auction dataset as well as the SB (unlabelled) dataset, thereby we and other researchers can apply various machine learning techniques by using authentic data of auctions and fraud.


2.1 Introduction

2.1.1 Auction Fraud

Online auctions have become a very lucrative e-commerce application. As an example, eBay, which is the largest auction site, recorded a net revenue of 9.7 billion U.S dollars in 2017, and the number of active users reached 170 million worldwide. Despite their popularity, e-auctions are very attractive to malicious moneymakers because auctions are vulnerable to cyber-crimes. This vulnerability is due to several facts, such as very low fees of auction services, anonymity of users, flexibility of bidding, and restricted legal policies. According to the Internet Crime Complain Centre (IC3), auction fraud represents one of the top cyber-crimes. For instance, in 2016, the count of auction complaints in only three states, California, Florida and New York, reached 7,448, and the victims’ financial loss increased to $5,900,977. Auction users can commit different kinds of fraud, which are classified w.r.t. the time periods in which fraudulent activities can happen:

- Pre-auction fraud, such as misrepresentation of products, auctioning of black market merchandise and stolen products.

- In-auction fraud, which happens during the bidding period, such as shill bidding, bid sniping and bid shielding.

- Post-auction fraud, such as non-delivery of products, product insurance and fees stacking.

Both pre- and post-auction fraud can be noticed by users as it relies on concrete evidence. Nevertheless, in-auction fraud does not leave any clear evidence, and worst of all, it is not noticed by honest bidders and victims i.e. auction winners.

\footnote{https://www.statista.com}
Indeed, it is challenging to detect fraud occurring during the bidding period, and in our study, we are concerned about this kind of fraud.

2.1.2 Shill Bidding Fraud

Shill Bidding (SB) is the most common auction fraud but the most difficult to detect due to its similarity to normal bidding behaviour [DSX09, FXV12]. A shill bidder is a malicious user (the fraudulent seller and/or his accomplices) who bids aggressively in order to drive up the price of the product only to benefit the owner of the auction. SB may cause a massive money loss for genuine sellers and bidders in the context of high priced products and also products with unknown value in the market, such as antiques [Bra10]. As mentioned in [Tre18, DSX09], excessive SB could lead to a market failure. Online auctions may affect the users’ confidence, which may negatively impacts the auctioning business. In fact, several sellers and their accomplices have been prosecuted due to SB activities, including:

- In 2001, three sellers were charged of SB fraud worth a pay-off of $300,000 through 1100 auctions of art paintings. The fraud was conducted on eBay with more than 40 fake accounts [Tre18].

- In 2007, a jewellery seller was accused of conducting SB fraud on eBay, and had to pay $400,000 for a settlement. Also, he and his employees were prevented from engaging in any online auctioning activities for four years.

- In 2010, a seller faced a £50,000 fine after being found outbidding himself on eBay. He claimed that: “eBay let me open up the second account and I gave all my personal details and home address to do so.”

---

• In 2012, the online auction ”TradeMe” had to pay $70,000 for each victim after the investigation discovered SB fraud conducted by a motor vehicle trader in Auckland. The fraud was carried out for one year, and caused a significant loss for the victims. Trade Me blocked this trader from using their site, and referred the case to the Commerce Commission for a further investigation ⁴.

• In 2014, a lawsuit was filled against ”Auction.com” by VRG in California claiming that the website allowed SB. The bid of $5.4 million should have secured the property as the plaintiff declared, and yet the winning price was 2 million more. Auction.com was accused of helping the property’s loan holder, which is not fair for genuine bidders. The California state passed a law on July 1, 2015, which requests the property auctioneers to reveal bids they submit on a seller’s behalf ⁵. The spokeswoman for the California Association of Realtors said: “To the best of our knowledge, we’re the only state to pass this sort of legislation, even though we believe SB to be prevalent all over the country.”

eBay policy states that “Shill bidding can happen regardless of whether the bidder knows the seller. However, when someone bidding on an item knows the seller, they might have information about the seller’s item that other shoppers aren’t aware of. This could create an unfair advantage, or cause another bidder to pay more than they should. We want to maintain a fair marketplace for all our users, and as such, shill bidding is prohibited on eBay.” ⁶. This statement clearly demonstrates that SB is troublesome and tough to be addressed.

2.1.3 Complexity of Generation SB Data

Safeguard against SB fraud is lacking because of several issues, including:

⁴https://www.trademe.co.nz/trust-safety/2012/9/29/shill-bidding
⁵https://nypost.com/2014/12/25/lawsuit-targets-googles-auction-com
• The difficulty of extracting authentic data from auction websites.

• The laboriousness of preprocessing the raw auction data.

• The difficulty of identifying proper SB patterns.

• The complexity of defining metrics for the SB patterns and measuring them from the original auction dataset.

To produce a reliable fraud training dataset, SB patterns must be measured from original auction data. The latter represent the real behaviour of auction users, which is important to develop robust SB classification models and perform valid empirical assessment. In commercial auctions, there is a tremendous amount of data that can be collected to provide useful information about the behaviour of users. However, obtaining online auction data is a very tedious task as demonstrated in the related work section. In our study, we have employed a robust commercial Web scraper called Octoparse that is able to capture a large volume of data from completed auctions of a certain product. Nevertheless, the original dataset contains irrelevant and redundant attributes, missing values and inappropriate value formatting. Therefore, preprocessing the auction dataset is a critical step before measuring the SB patterns and building robust fraud classifiers. Still, data preprocessing is a very time consuming phase as shown in this present report. The aim of this chapter is to share the preprocessed auction dataset as well as the SB training (unlabelled) dataset, thereby we and other researchers can employ various machine learning classification algorithms by using authentic auction and fraud data.

The remaining of this chapter is as follows: Section 1 demonstrates the auctions frauds in general, the SB fraud, the difficulties of obtaining SB dataset, and the generation of SB data. Section 2 reviews past studies on crawling auctions data. Section 3 explains how we extracted raw in-auction data. Section 4 describes the preprocessing steps of eBay auction data. Section 5 illustrates the characteristics and
metrics of SB patterns. Section 6 presents the generation of SB data from bidding data. Finally, Section 7 summarizes the findings of this study and highlights essential directions for the next chapter.

2.2 Related Work

Online auction sites, such as eBay, Taiwan, Yahoo!, TradeMe and uBid, provide massive amounts of valuable data (made public) that can be utilized by researchers to understand the behaviour and trends of auction users as well as the popularity of products. Nevertheless, extracting data from websites and then structure them into a proper dataset format are very challenging tasks. To this end, researchers may employ commercial (general) Web scrapers, or implement their own auction scrapers. But very few developers built their own auction scrapers to pursue their studies [YL08, BTLR09, PCWF07]. However, for unknown reasons, the software of the auction crawling are not available for the public use.

Due to the complexity and cost of collecting auction data, developers introduced different techniques to improve the scraping efficiency. For instance, [YL08] implemented a parallel crawling tool based on multi-agent technology. Two types of crawling agents have been introduced: one to find users in the auctions and store them in a list of user IDs to avoid duplication; the other one to check the user ID list and crawl the unvisited users, and create the users’ profile by capturing the auction contents. The experiment took place on eBay where only 7,682 pages were collected in an eight-hour time period, but without collecting the bidding list of each auction. If those lists were captured, the time may significantly increase.

Another extraction tool is suggested in [BTLR09], where the completed auctions are extracted immediately while the ongoing auctions are first traced by the software, and once finished, data are collected. This tool is concurrently set up on several
desktop computers to gather data using different search criteria of different auctions. Even though it was running on multiple computers, the tool extracted only 1,300 auctions during a one-month period. On the other hand, [PCWF07] applied a queue technique to ensure that a user is not crawled more than once. Every seen user who has not been crawled yet is stored in the queue. For each loop, the first user is popped and all his feedback ratings are crawled. Then the popped user is marked as visited and stored in a different queue. To increase the crawling time efficiency, parallelism is engaged using a naive breadth-first search technique.

Another option of collecting data is to utilize commercial Web crawlers, which provide portable software and services for users. The latter can purchase the license to download the scraper software, then extract the desired data. Also, users can have a contract with the web scraping provider to capture data for them and then deliver data in a certain format (CSV, Excel and API). There are two main issues faced by any Web scraper, be it general or auction specific: 1) the developers must obtain permission to crawl data from the desired auction site; 2) after a certain period (usually 2 to 3 months), auction sites delete data because of the storage limitation issue [GS17].

2.3 Extraction of Raw In-Auction Data

There are several professional web scrapers, such as Octoparse and Mozenda [DAM17], which extract data from any websites. We employed the fully automated scraping service provided by Octoparse ⁷ that is able to effectively collect a large number of auctions of a certain product, including all the details about the auctions, bidders and bids. In the main page containing the auction listing, each product page is parsed and information are then extracted.

We crawled from eBay the auctions of iPhone 7 for a period of three months,

⁷https://www.octoparse.com
March to June 2017 (see Appendix B). We selected this product for reasons that may increase the chance of SB activities:

- It is in high demand since it attracted a high number of bidders. After tracing the eBay search results for all types of iPhones, we obtained a daily average of 3808 iPhones auctioned in June 2017.\(^8\)

- It is marked as a “hot” product on eBay, which means it is among the most sold items in its category. According to Terapeak website [Ter16], 93% of iPhone 7 sales belong to the “cellphones and accessories” category.

- It has a good price range with the average of $610.17 (U.S currency). More the item price is high, more the possibility of fraud. Indeed, there is a direct relationship between SB activities and the auction price [SW17].

- The bidding duration varies between 1, 3, 5, 7 and 10 days. In long auction duration, a shill bidder may easily mimic usual bidding behaviour [DSX09]. However, as claimed in [CC14], fraudulent sellers receive positive rating in short bidding duration. Thus, we considered long and short bidding duration as shown in Table 2.1.

Table 2.1: Number of Auctions for Each Duration

<table>
<thead>
<tr>
<th>No. of Days</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Auctions</td>
<td>166</td>
<td>187</td>
<td>131</td>
<td>309</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>(20.57%)</td>
<td>(23.2%)</td>
<td>(16.23%)</td>
<td>(38.3%)</td>
<td>(1.7%)</td>
</tr>
</tbody>
</table>

More precisely, we focused on the most popular auction protocol [CS03]: Forward (one seller and multiple buyers) and English (open ascending price bidding). In addition, we utilized three filters to capture the iPhone 7 auctions: 1) from eBay.com

\(^8\)http://www.ebay.com/sch/Cell-Phones-Accessories
of North America, 2) from “cellphones and accessories” category, and 3) the winning price in each auction must be more than $100. In Table 2.2, we provide statistical information about the collected iPhone 7 auctions before the preprocessing task.

Table 2.2: Statistics of iPhone 7 Auctions

<table>
<thead>
<tr>
<th></th>
<th>Raw Data</th>
<th>Preprocessed Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Auctions</td>
<td>2551</td>
<td>807</td>
</tr>
<tr>
<td>Number of Records</td>
<td>399206</td>
<td>15145</td>
</tr>
<tr>
<td>Number of Bidder IDs</td>
<td>1226</td>
<td>1054</td>
</tr>
<tr>
<td>Number of Seller IDs</td>
<td>1727</td>
<td>647</td>
</tr>
<tr>
<td>Avg. Winning Price</td>
<td>$610.17</td>
<td>$578.64</td>
</tr>
<tr>
<td>Avg. Auction duration</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Number of attributes</td>
<td>28</td>
<td>12</td>
</tr>
</tbody>
</table>

2.4 Preprocessing of In-Auction Data

Original datasets often contain defective, missing and duplicated data that must be resolved to avoid misleading the learning process and returning an undesirable classification performance. This task consumes significant time and effort around 60% to 80% of the workload [RRRB11, CC11]. Our preprocessing phase consists of transforming auction data into suitable data to be able to produce high quality SB data. More precisely, we conducted several preprocessing tasks, including data cleansing, re-formatting, aggregation, and addition.

2.4.1 Data Cleansing

Firstly, we need to remove noisy data possessing the following characteristics:

- Irrelevant and duplicated attributes: several attributes in the raw dataset are
not needed to compute the SB metrics, such as the product location and ID, feedback ratings of sellers and bidders, and bidders’ account links. Also, some data are displayed twice on the main auction page and inside the auction link, such as the seller name, auction starting time, number of bids, and seller rating. There attributes are removed.

- **Duplicated records:** during the scrapping process, some data have been collected more than once, e.g., when a bidder participates more than once in an auction, the crawler collects his history each time. For example, let us suppose a bidder has 10 records in his bidding history, and he participated two times in an auction, then the crawler will grab his history each time. As a result, we would receive 20 records of that bidder but 10 of them have been already captured.

- **Records with missing values:** there are several rows without the bidders’ IDs; and we are not certain whether it was caused by eBay or the scrapper itself. These IDs cannot be generated by using the inputting techniques. So, we need to delete them.

- **Auctions with less than 5 bids:** these auctions did not engage any SB fraud because the very few placed bids are genuine. In some of these auctions, the items were sold using the “Buy-It-Now” feature on eBay (bidders can pay off the price directly by clicking Buy-It-Now button). In some other auctions, sellers canceled their sales due to reasons like the items were not available anymore for sale, or error discovered in the listing.

- **Auctions with inconsistent data:** several attributes contain incompatible values. For instance, the last submitted bid is greater than the winning price, or the

---

9[http://pages.ebay.co.uk/help/sell/questions/endlist-now.html](http://pages.ebay.co.uk/help/sell/questions/endlist-now.html)
starting price is greater than the winning price. So, we decided to remove these auctions to not mislead the fraud classifiers.

The statistics after cleaning noisy data are presented in Table 2.2. As we can see, a good number of records have been deleted due to the issue of duplicating data.

### 2.4.2 Attribute Aggregation

We need to aggregate the date and time attributes into a single attribute for three features as presented in Table 2.3. Also, we converted the month into a number.

<table>
<thead>
<tr>
<th>Table 2.3: Combining Date and Time into a Single Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auction Starting Date</strong></td>
</tr>
<tr>
<td>Jun-01-17</td>
</tr>
<tr>
<td><strong>Auction End Date</strong></td>
</tr>
<tr>
<td>Jun-03-17</td>
</tr>
<tr>
<td><strong>Bid Submission Date</strong></td>
</tr>
<tr>
<td>Jun-02-17</td>
</tr>
</tbody>
</table>

### 2.4.3 Attribute Reformatting

In Table 2.4, four features, date, time, duration and price, must be reformatted into a quantitative value to be able to compute the SB metrics. The date and time were both changed into seconds by calculating the seconds from the date we received the raw dataset (2017-07-07 00:00:00) to the date-time attributes in each auction. Regarding the final price attribute, we deleted the currency and converted the value data-type from text to number.
### Table 2.4: Attribute Reformatting

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Time</td>
<td>2017-06-19 20:11:19</td>
</tr>
<tr>
<td></td>
<td>1741721</td>
</tr>
<tr>
<td>End Time</td>
<td>2017-06-24 20:11:019</td>
</tr>
<tr>
<td></td>
<td>1309721</td>
</tr>
<tr>
<td>Bid Submit Time</td>
<td>2017-06-24 20:11:07</td>
</tr>
<tr>
<td></td>
<td>1309733</td>
</tr>
<tr>
<td>Auction Duration</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>432,000</td>
</tr>
<tr>
<td>Winning Bid (varchar)</td>
<td>650.50 $</td>
</tr>
<tr>
<td></td>
<td>650.50</td>
</tr>
</tbody>
</table>

#### 2.4.4 Attribute Adjustment

After a thorough examination of the whole auction dataset, we found out that several auctions contain attributes that are incompatible with the existing list of bids. For example, the “Number of Bids” attribute is different from the actual number of submitted bids in an auction. We have the same issue with the attribute ”Number of bidders”. We believe that these values have been corrupted during the crawling process. We fixed all the inconsistent values.

#### 2.4.5 Attribute Addition

In our original data, a specific URL represents an auction, which is not of a proper format for measuring SB patterns. To overcome this problem, a new attribute called AuctionID is given to provide a unique integer identifier for each auction. However, the AuctionID is repetitive in the dataset w.r.t. number of bids in that auction. This will make the computation of the SB patterns time consuming. So, the AuctionID
cannot be the primary key for the records in the auction dataset. Thus, there is a need to represent several attributes: AuctionID, BidderID and Bid Submit Time Sec, with a single identifier for each record. Thus, a new attribute is added to the dataset called recordID to uniquely identify each individual record. The relevant set of auction attributes are presented in Table 2.5.

Table 2.5: Auction Attributes for Shill Bidding

<table>
<thead>
<tr>
<th>Auction Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Auction ID</td>
</tr>
<tr>
<td>2 Seller ID</td>
</tr>
<tr>
<td>3 Number of Bidders</td>
</tr>
<tr>
<td>4 Starting Price</td>
</tr>
<tr>
<td>5 Auction Duration Sec</td>
</tr>
<tr>
<td>6 Start Time Sec</td>
</tr>
<tr>
<td>7 End Time Sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bid Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Bidder ID</td>
</tr>
<tr>
<td>2 Bid Amount</td>
</tr>
<tr>
<td>3 Bid Submit Time Sec</td>
</tr>
<tr>
<td>4 Number of Bids</td>
</tr>
<tr>
<td>5 Winning Bid</td>
</tr>
<tr>
<td>6 Record ID</td>
</tr>
</tbody>
</table>

2.5 Characteristics and Metrics of SB

To increase the revenue of the seller, a shill bidder inflates the price of the product by bidding aggressively. Once the price is high enough for the seller and the last submitted bid is of a honest bidder, the shill bidder stops competing, and thus making
the normal bidder win the auction. Thus, the winning bidder pays much more than the real value of the product

2.5.1 Strategies of SB

By examining throughly the literature on the SB strategies [DSX10, FXV12, SW17], we compiled below the most common and most relevant SB patterns. Moreover, the selected SB patterns are non redundant as they represent one unique aspect of the bidding behaviour. This is very important to build robust classifiers.

1. **Bidder Tendency:** a shill bidder participates exclusively in auctions of few sellers rather than a diversified lot. This is a collusive act involving the fraudulent seller and an accomplice. The latter acts as a normal bidder to raise the price.

2. **Early Bidding:** a shill bidder tends to bid pretty early in the auction (less than 25% of the auction duration) to get the attention of auction users.

3. **Bidding Ratio:** a shill bidder participates more frequently to raise the auction price and attract higher bids from legitimate participants.

4. **Last Bidding:** a shill bidder becomes inactive at the last stage of the auction (more than 90% of the auction duration) to avoid winning the auction.

5. **Auction Starting Price:** a shill bidder usually offers a small starting price to attract legitimate bidders into the auction.

6. **Successive Outbidding:** a shill bidder successively outbids himself even though he is the current winner to increase the price gradually with small consecutive increments.

7. **Winning Ratio:** a shill bidder competes in many auctions but hardly wins any auctions.
8. **Auction Bids:** auctions with SB activities tend to have a much higher number of bids than the average of bids in concurrent auctions (i.e. selling the same product). Therefore, sellers of these auctions have a high probability of colluding with shill bidders to increase their profits.

We would like to mention that we did not use four other SB patterns proposed in the literature:

1. “Nibble Bidding” refers to a bidder who outbids others with a very small increment. We do not consider this pattern as a strong sign of fraud since normal bidders may do the same. Additionally, this pattern is not present in the iPhone7 auctions because the minimum bid increment has been fixed to $5.

2. “Reserve Price Shilling” denotes a shill bidder who aggressively outbids himself as long as the current auction price is less than the reserved price. We do not take into account this pattern for the simple reason that the reverse price is hidden by eBay. Nevertheless, Successive Outbidding and Bidding Ratio patterns are enough to cover this pattern.

3. “Buyer Rating” and “Seller Rating” patterns represent the bidder’s and seller’s reputations in the auction house where buyers and seller can leave a feedback rating once the transaction has been completed. We excluded these patterns due to the high potential of misusing this feature by fraudulent rings [FXV12, DSX09].

2.5.2 **Weights of SB**

As presented in Table 2.6, we have categorized SB patterns in two dimensions: the first one illustrates the pattern category (bid-wise, bidder-wise or auction-wise), and

---

10 https://community.ebay.com/t5/Archive-Buyer-Central/newbie-buyer-Q-re-quot-nibbling-quot/td-p/2789578
the second one its relative weight.

1. Bidder property refers to the SB pattern that describes the user’s participation in the whole auction dataset. These patterns are bidder tendency and winning ratio, which respectively examine how many times a user attended to a specific seller auction, and his intention to whether win the auction or just inflate the price.

2. Bid property illustrates features related to the submitted bid, such as early bidding to check the time between the auction starting time and the first placed bid, bidding ratio to compare the bidder’s participation with other users, successive outbidding to check whether the bidder outbidded himself or not, and last bidding to verify whether a bidder became idle at the last stage of the auction.

3. Auction property is inherent to the auction itself, such as auction starting price, which is compared to the average of all the auctions’ starting prices, and auction bidding is compared to the average of the number of bids in all auctions.

As mentioned earlier, shill bidders may behave similarly to usual bidders [GS17, FXV10]. Each SB pattern reflects a specific aspect of the bidder’s behaviour. Thus, it is necessary to study each pattern individually to provide a proper weight. But first let us consider the following two scenarios: (1) a bidder outbids others aggressively with the real intention of winning the auction, not to raise the item price. The problem with this scenario is that the bidding ratio is high; (2) A seller gives low starting price just to attract more bidders in order to sell the product quickly. Therefore, to differentiate between suspicious and normal bidders, we assign different levels of weights: low, medium and high [GS17, SW17].

1. **Low weight SB:** Early bidding and auction starting price are given low weight because they occur very early in an auction. The high number of auction bids
might refer to the high quality of an item or the excellent reputation of a seller [GS17]. Thus, low weight is assigned to it as well.

2. **Medium weight SB:** In general, last bidding indicates the genuineness of the bidders to not win the auction [DSX10]. However, the cost of an item might get higher than what it is worth from the bidder’s perspective, hence, he stops bidding. Therefore, medium weight is assigned to last bidding pattern. Medium weight is given to the bidder tendency pattern as well since on one side, some sellers make under table contract with bidders to increase their profits, but on the other side, the bidder is honest and desires to win the auction. Bidding ratio is also considered of low weight since high bidding ratio might indicate the motive to win the auction.

3. **High weight SB:** Since winning ratio pattern reflects the user behavior in the all auctions he participated, high weight is given to it. Also, the goal of shill bidders is not to win the auction but to increase the price of an item. A high weight is also assigned to successive outbidding pattern since normal users surely will not outbid themselves [SW17].

As suggested in [SW17], we manually assign the value of 0.3 to the low weight, 0.5 to the medium weight and 0.7 to the high weight.
Table 2.6: Properties and Weights of SB Patterns

<table>
<thead>
<tr>
<th>Fraud Pattern</th>
<th>Property</th>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidder Tendency</td>
<td>Bidder</td>
<td>Medium</td>
<td>0.5</td>
</tr>
<tr>
<td>Early Bidding</td>
<td>Bid</td>
<td>Low</td>
<td>0.3</td>
</tr>
<tr>
<td>Bidding Ratio</td>
<td>Bid</td>
<td>Medium</td>
<td>0.7</td>
</tr>
<tr>
<td>Last Bidding</td>
<td>Bid</td>
<td>Medium</td>
<td>0.5</td>
</tr>
<tr>
<td>Auction Starting Price</td>
<td>Auction</td>
<td>Low</td>
<td>0.3</td>
</tr>
<tr>
<td>Successive Outbidding</td>
<td>Bid</td>
<td>High</td>
<td>0.7</td>
</tr>
<tr>
<td>Winning Ratio</td>
<td>Bidder</td>
<td>High</td>
<td>0.7</td>
</tr>
<tr>
<td>Auction Bids</td>
<td>Auction</td>
<td>Low</td>
<td>0.3</td>
</tr>
</tbody>
</table>

2.5.3 Metrics of SB

The metrics are calculated from the auction dataset. Each metric is scaled to the range of \([0, 1]\). High values refer to a suspicious bidding behaviour. Below, \(B\) denotes a bidder (Bidder ID) who competed in an auction \(A\) (Auction ID) initiated by seller \(S\) (Seller ID). The metrics of the selected fraud patterns are provided below. In Appendix A, we provide the implementations of the eight SB metrics where all the metrics are built using MMSQL, except successive outbidding was implemented in Python due to its complexity.

1. **Bidder Tendency (BT):**

\[
BT(B, S) = 0 \\
\text{if } participateAllAuctions(B) > 1 \text{ then} \\
\quad BT(B, S) = \frac{participatewithSeller(B, S)}{participateAllAuctions(B)}
\]

\text{end if}

where \(participatewithSeller()\) denotes the total number of a bidder’s partici-
pations for a specific seller, and participateAllAuctions() the total number of auctions (from the iPhone 7 dataset) that a bidder participated in.

2. Early Bidding (EB):

\[
EB(B, A) = 1 - \frac{\text{StartTimeSec}_A - \text{BidSubmitTimeSec}_{B,A}}{\text{AuctionDurationSec}_A}
\]

3. Bidding Ratio (BR):

\[
BR(B, A) = \frac{\text{totalBids}(B, A)}{\text{NumberOfBids}_A}
\]

where totalBids() is the number of submitted bids by a bidder in an auction.

4. Last Bidding (LB):

\[
LB(B, A) = \frac{\text{BidSubmitTimeSec}_{B,A} - \text{EndTimeSec}_A}{\text{AuctionDurationSec}_A}
\]

5. Auction Starting Price (ASP):

\[
ASP(A) = 0
\]

\[
\text{if } \text{StartingPrice}_A < \text{avgAuctionsStartPrice(auctionDataset)} \text{ then}

ASP(A) = 1 - \frac{\text{StartingPrice}_A}{\text{avgAuctionsStartPrice(auctionDataset)}}
\]

end if

where avgAuctionsStartPrice() is the average starting price in all the auctions from the dataset.
6. **Successive OutBidding (SOB):**

\[
SOB = 0 \\
do \\
\text{if } (\text{successiveBid}(B, A) \geq 4) \text{ then} \\
\{SOB(B, A) = 1; \text{quit}\} \\
\text{else if } (\text{successiveBid}(B, A) \geq 3) \text{ then} \\
\{SOB(B, A) = 0.5; \text{quit}\} \\
\text{end if} \\
\text{while } (\text{notReached(lastBid(B, A)))}
\]

where \(\text{successiveBid}()\) denotes the number of successive outbids submitted by a bidder \(B\) in an auction \(A\).

7. **Winning Ratio (WR):**

\[
WR(B) = 1 - \frac{\text{AuctionWon}(B)}{\text{auctionPartHigh}(B)}
\]

where \(\text{auctionPartHigh} = \text{count}\{A|BR(B, A) > 0.1\}\). The \(\text{AuctionWon}()\) denotes the total number of auctions won by a bidder, and \(\text{auctionPartHigh}()\) is the total number of auctions where the bidding ratio of the bidder is higher than 10% of the total bids. This will eliminate the issue of non-active users.

8. **Auction Bids (AB):**

\[
AB(A) = 0 \\
\text{if } \text{NumberOfBids}_A > \text{avgBidAllAuctions(auctionDataset)} \text{ then} \\
AB(A) = 1 - \frac{\text{avgBidAllAuctions(auctionDataset)}}{\text{NumberOfBids}_A} \\
\text{end if}
\]

where \(\text{avgBidAllAuctions}()\) is the average number of bids in all the auctions in the dataset.
2.6 Generation of SB Data from Auctions

To produce SB data, the eight SB patterns are measured for all the bidders of the iPhone7 auctions. To accomplish this task, we use MSSQL to store the values of the 12 attributes for each auction, and then calculate each pattern against each bidder in each auction (see Appendixes A, and B). As a result, we generate a SB dataset with a tally of 6321 instances. An instance represents the conduct of bidders in a certain auction. It is a vector of 10 features: Auction ID, Bidder ID and the eight SB patterns. Table 2.7 provides some statistical information about our SB dataset. We note that we examined carefully the whole SB dataset to check for outliers (values that are outside the range of [0, 1]), and we did not find any. Outliers maybe present in a dataset due to corrupted data during the crawling process.

<table>
<thead>
<tr>
<th>Table 2.7: Statistics of SB Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of auctions</td>
</tr>
<tr>
<td>Number of instances</td>
</tr>
<tr>
<td>Number of bidder IDs</td>
</tr>
<tr>
<td>Avg. bidders in low starting price auctions</td>
</tr>
<tr>
<td>Avg. bidders in high starting price auctions</td>
</tr>
<tr>
<td>Winners and not aggressively participated</td>
</tr>
<tr>
<td>Not winners and aggressively participated</td>
</tr>
<tr>
<td>Not winners and not aggressively participated</td>
</tr>
</tbody>
</table>

From the SB dataset, we deduced the following facts as shown in Table 2.7:

• 26.25% of bidders aggressively outbidded themselves and others but did not win any auctions. The behavior of those bidders indicates that they committed SB.

• 8.5% of bidders did not highly participated in the auctions but won. Those bidders were fairly active at the last auction stage. All these refer to genuine
behaviour.

- 60.4% of bidders looked normal and did not win. Indeed, those bidders did not aggressively outbid others and did not submit successive bids.

- 4.9% of bidders extremely outbidded others and won the auctions. This indicates their desire to win the auction.

- The average number of bidders in auctions with a low starting price is 10, whereas the average is 6 in regular auctions with a regular starting price.

- 11.5% of bidders submitted bids at an early stage and aggressively outbid others but did not win any auctions. This indicates their intention to increase the item price.

Table 2.8: SB Patterns in the Auction Dataset

<table>
<thead>
<tr>
<th>SB Pattern</th>
<th>High Value (&gt; 0.7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidder Tendency</td>
<td>209 (3.3%)</td>
</tr>
<tr>
<td>Early Bidding</td>
<td>2112 (33.4%)</td>
</tr>
<tr>
<td>Bidding Ratio</td>
<td>43 (0.68%)</td>
</tr>
<tr>
<td>Last Bidding</td>
<td>1976 (31.26%)</td>
</tr>
<tr>
<td>Auction Starting Price</td>
<td>2944 (46.6%)</td>
</tr>
<tr>
<td>Successive Outbidding</td>
<td>1968 (31.13%)</td>
</tr>
<tr>
<td>Winning Ratio</td>
<td>1659 (26.24%)</td>
</tr>
<tr>
<td>Auction Bids</td>
<td>221 (3.5%)</td>
</tr>
</tbody>
</table>

As presented in Table 2.8, SOB pattern is relatively high since 31.13% of samples/bidders aggressively outbidding others. LB pattern also shows that 31.26% of bidders remained inactive at the last auction stage whereas 33.4% of bidders partici-
pated at the early stage. A good number of bidders (46.6%) tends to bid in auctions with a low starting price.

2.7 Conclusion

Here, we provided the achieved tasks in this chapter (specified in green Figure 2.1) and highlighted the directions of our research work (specified in red Figure 2.1). We have utilized a robust commercial Web scraper called Octoparse that captured a large data from completed auctions of a iPhone 7 product since is considered one of the most popular auctions on eBay. After the raw data is collected, we have conducted intensive preprocessing operations which are: 1) removing irrelevant and duplicated attributes as well as records with inconsistent values and missing values (like IDs of bidders and sellers); 2) merging several attributes into a single one; 3) converting several attributes into proper formats; and 4) generating IDs for the auctions. These tasks was very exhausted, difficult, and time consuming. To develop the SB dataset, we first determined the most effective SB patterns where each SB pattern is unique as it represents one aspect of the bidding behavior. Then, we produced a SB dataset with a tally of 6321 instances. The reprocessed auction dataset as well as the SB (unlabelled) dataset will be made public, thereby researchers, in the same field, can implement different detection models based on Machine Learning (ML) techniques by using our authentic data of auctions. In the coming chapter, we will concentrate on the following points:

- One of the most challenging tasks in ML and Data Mining (DM) is to label the training dataset. Subsequently, the next phase is to label the SB data into “Normal” and “Suspicious” by using two main methods: unsupervised learning, such as clustering.

- Fraud data are highly imbalanced, and this class imbalance has been shown to
reduce the performance of baseline classifiers as demonstrated in [GS18a]. In this situation, the classifiers are biased towards the normal class, which means that fraud instances tend to be classified as normal ones. The imbalanced learning problem is an active area of study [ZSM15], which could be solved via data sampling or cost-sensitive learning.

- Thousands of transactions occur in eBay daily, and auction data may be sent continuously to the optimal fraud classifier to detect potential SB activities. Consequently, instance-based classification is suitable for our particular fraud detection problem.

Figure 2.1: High-Quality SB Dataset Construction and Classification
Chapter 3

Clustering and Labelling SB Data

Abstract: Although SB is a common auction fraud, it is still very tough to be detected. In this thesis, we produced a high-quality SB training dataset based on a recently collected data from eBay, which is described in chapter 2. The generated high-quality SB training dataset is not classified; thus, it will be a very challenging task since the classification quality based on ML algorithms depends on how accurate the labels of the existing instances. The process of labeling instances is among the most difficult and complicated steps that confront traditional supervised classifications in real-world applications. Therefore, the objective of this chapter is to address the above mentioned difficulties and to provide a labeled SB patterns dataset to be used for SB detection systems based on machine learning. Since hierarchical clustering algorithms show outstanding results for determining and grouping similar data samples, we employ CURE algorithm to differentiate between normal and suspicious bidders behavior since it can handle large data and illuminate outliers. As illustrated in the experiment, the applied clustering produces excellent results.
3.1 Introduction

The bidder behavior in online auctions can be statistically identified and measured by different features. However, identifying relevant SB strategies, determining robust SB metrics, crawling and preprocessing commercial auction data, and evaluating the SB metrics against the extracted data make the study of SB fraud very challenging, as demonstrated in our technical paper [AS18]. The labeling process of an unclassified dataset is a highly critical phase in ML and DM for classification purposes. In addition, labeling SB instances with multi-dimensional features is a critical phase for the classification models. In the literature, labeling training data is usually done manually by the domain experts, which is a quite laborious task and prone to errors. Due to the lack of labeled SB training datasets, the prime contribution of this chapter is to produce high-quality labeled SB data based on commercial auction transactions that we extracted from eBay and preprocessed [AS18]. As illustrated in Figure 3.1, we introduce a new approach to efficiently label SB data with the help of data clustering. Firstly, we split the SB dataset into several subsets according to the different bidding durations of the extracted auctions. Secondly, we efficiently partition each SB subset into clusters of users with similar bidding behaviour. Last, we apply a systematic labeling method to each cluster to classify bidders into normal or suspicious.

Hierarchical clustering is significantly preferable over partitioning clustering because it provides clusters with a higher quality [GS18b]. This type of data clustering has been utilized successfully in numerous fraud studies [FXV12, GS18b]. In fact, we employ the Clustering Using REpresentatives (CURE) technique to effectively produce the differentiation between normal and suspicious bidders activities. CURE [GRS01] has proved over the years to be a highly efficient clustering method in terms of eliminating outliers and producing high quality clusters, especially for large-scale training datasets. The labeled SB dataset that we produced can be utilized by the state-of-the-art supervised classification methods. Furthermore, the accuracy of
new predictive models can also be tested using our SB training dataset.

The rest of this chapter is organized as follows. Section 2 discusses related work on data clustering in the context of online auctions. Section 3 explains how to employ hierarchical clustering CURE according to different bidding durations. Section 4 explains the properties of other SB subsets. Section 5 describes the systematical labelling approach to assign labels for each cluster of bidders. Finally, Section 6 summarizes the results of this study and highlights research directions for the next chapter.

### 3.2 Related Work

Many researchers utilized data clustering to examine auctions from different angles, such as studying the dynamics of auction prices and bidder behaviour. For instance, [JS09] introduced an approach to model and analyze the price formation as well as its dynamics to characterize the heterogeneity of the price formation process. The proposed functional objects represent the price process by accommodating the structure format of bidding data on eBay. Then, the curve clustering is used to partition auctions by grouping similar price profiles. Finally, differential equations are used to specify the price of each group.

Another work [FXV10] measured the similarity of bidder behaviour using specific
attributes, such as bidder feedback rating, average increment difference and number of bids. Then, a centroid-based hierarchical clustering approach is presented to group similar bidders. Each produced cluster is then labeled manually according to the overall bidder behaviour in that cluster.

The study [BHH+12] suggested a SB detection model utilizing k-mean clustering technique. The latter groups similar buyers in one class to differentiate between general buyers and shill bidders. There are four features that represent a buyer: “how long the buyer has been in the auction”, “the times of buyer bids”, “the average response time of the buyer” and ‘the absolute average discrepancy of buyer bids”. Based on these features, k-mean classifies bidders into one of the two clusters: general buyers and shill bidders. However, we believe the second and third features do not really reflect SB since a buyer might be very interested in winning the auction. Besides, there are stronger patterns that highly identify SB. Since SB behaviour is somehow similar to real bidding [DSX09], there is a possibility that some SB samples fell in the normal class.

In [CL13], the authors proposed a two-step clustering model to recognize bidding strategies. The hierarchical clustering is the first step to produce a dendrogram and an agglomeration schedule table to find the best number of clusters. The subsequent step employs k-mean clustering to provide more details about the bidders’ strategies in each cluster. The experiment was operated on an outdated data (2003) collected from Taobao.com. According to the agglomeration coefficients, the optimal number of clusters is three: “early bidding strategy”, “snipe bidding strategy”, and the third one groups bidders that enter the auction early and remain for a long time. The first cluster has shown low values for the given features, which indicates that the bidders’ strategy is to enter and exit auctions early and participate infrequently. The second cluster has displayed high values for some features and low values for others. This illustrates that bidders enter and exit auctions late and rarely participate. The last
cluster has administered the bidders’ strategy where bidders enter early and stay for a long time and highly participate in the auctions.

[ZPL13] proposed a model based on hedonic regression and fuzzy logic expert system (FLES) to analyze bidder behaviour. The hedonic regression is used to select key variables that are passed to FLES to produce a knowledge base about the relationships between variables, like auction characteristics. Since the examined data have no relational information, k-mean is employed to obtain the minimum squared-error clusters. So, each training sample is classified into low, medium or high membership degree. The issue here is that the study is based on an outdated dataset (2004). Also, there is a potential for fabricating feedback ratings conducted between shill bidders and fraudulent bidder rings [FXV10].

Recently, [GM15] applied k-mean clustering to categorize bidders’ habits. The observations are obtained by the k-mean and then passed to the Baum-Welch algorithm and Hidden Markov Model. Three main clusters were suggested according to the values of the given features, which are low, medium and high cluster values. A bidder habit with values beyond these clusters values is considered as a fraudster. The experiment showed that only two simple features were given to identify the clusters: the number of auctions that a bidder participated in and the number of submitted bids by that bidder. Thus, if more features were considered to define the clusters, then the samples distribution on each cluster might be changed. As a result, this may influence the outcomes of the detection model. Also, the clustering is based on a dataset that is not adequately described, and only ten samples were used for explaining the results.

Lastly, [GS18b] applied a hierarchical clustering to group users with similar bidding behaviour. The centroid linkage is used as the similarity measure. The described SB patterns were computed for all the bidders of each of the generated clusters. Then, the authors introduced a semi-automatic approach to label each cluster according to
the general behaviour of users in that cluster and the weights of the fraud patterns.

3.3 Hierarchical Clustering of SB Data

Since the produced SB data are not labeled, data clustering, an unsupervised learning method, can be utilized to facilitate the labeling operation. Clustering is the process of isolating instances into K groups w.r.t. their similarities. The clustering techniques fall into one of the following categories: 1) Partitioning-based, such as K-medoids and K-means; 2) Hierarchical-based, such as BIRCH, GRIDCLUST and CURE; 3) Density-based, such as DBSCAN and DBCLASD; 4) Grid-based, such as STING and CLIQUE. In our work, we select agglomerative (bottom-top) hierarchical clustering, where instances are arranged in the form of a tree structure, using a proximity matrix.

3.3.1 CURE Overview

Among the hierarchical clustering methods, we choose CURE because it is highly performant in handling large-scale multi-dimensional datasets, it determines non-spherical shapes of the clusters, and efficiently eliminates outliers [GRS01]. Random sampling and partitioning techniques are utilized to handle the large-scale problem and to speed up the clustering operation. Each instance is first considered as an individual cluster, and then the cluster with the closest distance/similarity is merged with it in order to form a new cluster, as described in Figure 3.2.

The procedure of CURE algorithm is described in Figure 3.3 (updated version of [GRS98]). Two novel strategies have been introduced in CURE:

- **Representative Points (RPs)**, which are selected data points that define the cluster boundary. Instead of using a centroid, clusters are identified by a fixed number of RPs that are well dispersed. Clusters with the closest RPs are merged into one cluster. The multiplicity of RPs allow CURE to obtain
arbitrary clustering shapes.

- **Constant shrinking factor** ($\alpha$), which is utilized to shrink the distance of RPs towards the centroid of the cluster. This factor reduces noise and outliers.

![Figure 3.2: Overview of CURE [GRS01]](image)

In the worst case, the computational complexity of CURE is estimated to $O(N^2 \log N)$, which is high when $N$ is large ($N$ is the number of instances) [XT15]. Since the SB data clustering is an offline operation, so the running time is not an issue. The only disadvantage of CURE is that the two parameters RP and $\alpha$ have to be set up by users.

To run the experiments, we utilize the Anaconda-Navigator environment for running Python 3, and incorporate the CURE program developed by Freddy Stein and Zach Levonian. CURE code (in Python) is available at GitHub.com\(^1\) (see Appendix

\(^1\)https://github.com/levoniaz/python-cure-implementation/blob/master/cure.py
Figure 3.3: CURE Clustering Algorithm Procedures Description
3.3.2 SB Data Preparation

Since the bidding duration is used as a denominator for the two patterns Early Bidding and Last Bidding, the large gap between different durations greatly affects the obtained results. The computed value of the fraud pattern for 10 days is far smaller than that of 1 day. Therefore, before applying CURE, we first partition the SB dataset into five subsets according to the five durations (1, 3, 5, 7 and 10 days) as presented in Table 3.1.

Table 3.1: SB Dataset Partitioning According to Bidding Durations

<table>
<thead>
<tr>
<th>Subset</th>
<th>1 Day</th>
<th>3 Days</th>
<th>5 Days</th>
<th>7 Days</th>
<th>10 Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Auctions</td>
<td>166</td>
<td>187</td>
<td>131</td>
<td>309</td>
<td>14</td>
</tr>
<tr>
<td>No. of Instances</td>
<td>1289</td>
<td>1408</td>
<td>1060</td>
<td>2427</td>
<td>137</td>
</tr>
</tbody>
</table>

3.3.3 Optimal Number of Clusters

It is always difficult to decide about the optimal number of clusters of a training dataset. Besides, normal and shill bidder behaviours are somehow similar. Thus, determining the best number of clusters is an essential step to achieve a better interpretation for classifying similar SB instances [YLW14]. There are several methods to address this problem, such as Elbow, Dendrogram, the Rule of Thumb and Silhouette. In our study, we employ the Silhouette method where each group is represented as a silhouette based on the separation between instances and the cluster’s tightness [Rou87]. The construction of a silhouette requires the clustering technique to generate the partitions and collect all approximates between instances [Rou87]. K-mean clustering algorithm has been successfully utilized for this task due to its simplicity and effectiveness [YLW14]. Consequently, we apply K-mean to estimate the number of clusters for each of the five SB subsets.

Next, for each subset, we examine the silhouette scores of 19 clusters (2 to 20 clusters), and choose the best number based on the best silhouette score. We have
noticed that once the peek (denoting the optimal number) of the silhouette score is reached on a certain number of clusters, the silhouette score gradually decreases with the increasing of the number of clusters. In Figure 3.4, we give an example for the 7-day bidding duration for which the optimal number of clusters is eight since the highest silhouette score (0.4669) is obtained on that number. In Table 3.7, we expose the best number of clusters for each of the five SB subsets. The total number of produced clusters is 29.

Figure 3.4: Optimal number of clusters for 7 day bidding duration. Silhouette score is examined 19 times. We show the top Silhouette scores.
3.3.4 Cluster Generation

CURE has three parameters that need to be setup: representative points (RPs), shrinking factor ($\alpha$) and optimal number of clusters. Based on the results of silhouette, we have obtained the optimal number of clusters for each of the five SB subsets. The two parameters RPs and $\alpha$ are defined by selecting the configuration that provides the best instance distribution among the specified clusters. Thus, CURE is applied with different values of RPs and $\alpha$ starting from the default values (5 for RPs and 0.1 for $\alpha$). The best parameters’ values are selected based on the best distribution of a subset population between the defined clusters. As an example, in Table 3.2, we present the results for the eight clusters that we have generated previously for the 7-day bidding duration subset. The best value configuration is shown in bold. Each cluster consists of users with similar bidding behaviour. As we can see, the clusters 4, 7 and 8 have very few bidders, which are most probably outliers i.e. suspicious.

Table 3.2: CURE Clustering for 7-day Subset on 8 Clusters with different RPs and $\alpha$

<table>
<thead>
<tr>
<th>RP</th>
<th>$\alpha$</th>
<th>Cl.#1</th>
<th>Cl.#2</th>
<th>Cl.#3</th>
<th>Cl.#4</th>
<th>Cl.#5</th>
<th>Cl.#6</th>
<th>Cl.#7</th>
<th>Cl.#8</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.1</td>
<td>136</td>
<td>1438</td>
<td>1</td>
<td>2</td>
<td>657</td>
<td>190</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.05</td>
<td>657</td>
<td>328</td>
<td>2</td>
<td>1408</td>
<td>1</td>
<td>28</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>1438</td>
<td>640</td>
<td>1</td>
<td>1</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>328</td>
</tr>
<tr>
<td>5</td>
<td>0.001</td>
<td>21</td>
<td>166</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>25</td>
<td>2209</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>2</td>
<td>1</td>
<td>657</td>
<td>1410</td>
<td>22</td>
<td>8</td>
<td>137</td>
<td>190</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>2</td>
<td>133</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>31</td>
<td>2066</td>
<td>190</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>135</td>
<td>31</td>
<td>2067</td>
<td>190</td>
</tr>
<tr>
<td>10</td>
<td>0.001</td>
<td>189</td>
<td>654</td>
<td>1410</td>
<td>3</td>
<td>137</td>
<td>31</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

3.4 Properties of Other Subsets

3.4.1 1-Day Subset

In this subset, the instances are grouped into 7 clusters since the highest silhouette score (0.4597) is obtained with this number, as shown in Figure 3.5 (D). The scores
gradually decrease with less and more than 7 clusters. Also, the selected RPs and \( \alpha \) parameters for CURE are 5 points and 0.05, as presented in Table 3.3.

![Figure 3.5: Optimal number of clusters for 1 day bidding duration](image)

<table>
<thead>
<tr>
<th>RP</th>
<th>( \alpha )</th>
<th>Cl.#1</th>
<th>Cl.#2</th>
<th>Cl.#3</th>
<th>Cl.#4</th>
<th>Cl.#5</th>
<th>Cl.#6</th>
<th>Cl.#7</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>148</td>
<td>1136</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.05</td>
<td>421</td>
<td>58</td>
<td>714</td>
<td>2</td>
<td>61</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>15</td>
<td>78</td>
<td>34</td>
<td>715</td>
<td>445</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.001</td>
<td>715</td>
<td>490</td>
<td>1</td>
<td>2</td>
<td>79</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>479</td>
<td>715</td>
<td>90</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>31</td>
<td>3</td>
<td>93</td>
<td>3</td>
<td>713</td>
<td>445</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>1186</td>
<td>1</td>
<td>89</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.001</td>
<td>34</td>
<td>1</td>
<td>537</td>
<td>714</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.3: CURE Clustering for 1-day Subset on 7 Clusters with different RPs and \( \alpha \)
3.4.2 3-Days Subset

The 3-days dataset instances are grouped into 7 clusters since the highest silhouette score (0.4672) is obtained on that number of clusters as shown in Figure 3.6 (C). The scores are gradually decrease on the less and more than 7 clusters. Also, the selected RPs and $\alpha$ parameters for CURE are 5 points and 0.01, as presented in Table 3.4.

![Figure 3.6: Optimal number of clusters for 3 day bidding duration](image)

Table 3.4: CURE Clustering for 3-days Subset on 7 Clusters with different RPs and $\alpha$

<table>
<thead>
<tr>
<th>RP</th>
<th>$\alpha$</th>
<th>Cl.#1</th>
<th>Cl.#2</th>
<th>Cl.#3</th>
<th>Cl.#4</th>
<th>Cl.#5</th>
<th>Cl.#6</th>
<th>Cl.#7</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.1</td>
<td>102</td>
<td>796</td>
<td>2</td>
<td>501</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.05</td>
<td>1</td>
<td>1</td>
<td>102</td>
<td>502</td>
<td>1</td>
<td>800</td>
<td>1</td>
</tr>
<tr>
<td><strong>5</strong></td>
<td><strong>0.01</strong></td>
<td><strong>1</strong></td>
<td><strong>102</strong></td>
<td><strong>803</strong></td>
<td><strong>453</strong></td>
<td><strong>1</strong></td>
<td><strong>47</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td>5</td>
<td>0.001</td>
<td>603</td>
<td>1</td>
<td>6</td>
<td>794</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>2</td>
<td>3</td>
<td>1223</td>
<td>1</td>
<td>4</td>
<td>174</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>799</td>
<td>1</td>
<td>1</td>
<td>602</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>1</td>
<td>603</td>
<td>795</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.001</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>794</td>
<td>557</td>
<td>47</td>
<td>1</td>
</tr>
</tbody>
</table>
3.4.3 5-Days Subset

The instances in 5-days subset are grouped into 5 clusters since the highest silhouette score (0.4758) is obtained with this number, as shown in Figure 3.5 (A). The scores gradually decrease with less and more than 5 clusters. Also, the selected RPs and $\alpha$ parameters for CURE are 5 points and 0.05, as presented in Table 3.5.

![Figure 3.7: Optimal number of clusters for 5 day bidding duration](image)

<table>
<thead>
<tr>
<th>RP</th>
<th>$\alpha$</th>
<th>Cl. #1</th>
<th>Cl. #2</th>
<th>Cl. #3</th>
<th>Cl. #4</th>
<th>Cl. #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>959</td>
<td>88</td>
</tr>
<tr>
<td>5</td>
<td>0.05</td>
<td>24</td>
<td>73</td>
<td>12</td>
<td>355</td>
<td>596</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>460</td>
<td>2</td>
<td>596</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.001</td>
<td>2</td>
<td>354</td>
<td>109</td>
<td>594</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>105</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>951</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>1</td>
<td>948</td>
<td>107</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1055</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1056</td>
<td>1</td>
</tr>
</tbody>
</table>
3.4.4 10-Days Subset

The highest silhouette score is 0.5549, which is obtained on 2 clusters in the 10-days dataset as shown in Figure 3.8 (A). The score gradually decreases on more than 2 clusters. Also, the selected RPs and $\alpha$ parameters for CURE are 5 points and 0.1, as presented in Table 3.6.

![Silhouette plots](image)

Figure 3.8: Optimal number of clusters for 10 day bidding duration

The summarize of the conducted trial-and-error experiments for all the clusters (29 clusters in total) of the five SB subsets to determine the best CURE parameters values are represented in Table 3.7.
Table 3.6: CURE Clustering for 10-days Subset on 2 Clusters with different RPs and $\alpha$

<table>
<thead>
<tr>
<th>RP</th>
<th>$\alpha$</th>
<th>Cl.#1</th>
<th>Cl.#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.1</td>
<td>135</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0.05</td>
<td>135</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>135</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0.001</td>
<td>135</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>135</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>135</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>135</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.001</td>
<td>135</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.7: Optimal Number of Clusters and Optimal CURE Parameters

<table>
<thead>
<tr>
<th>SB Subset</th>
<th>No. of Samples</th>
<th>No. of Clusters</th>
<th>Silhouette Score</th>
<th>RPs</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Day</td>
<td>1289</td>
<td>7</td>
<td>0.4597</td>
<td>5</td>
<td>0.05</td>
</tr>
<tr>
<td>3 Days</td>
<td>1408</td>
<td>7</td>
<td>0.4672</td>
<td>5</td>
<td>0.01</td>
</tr>
<tr>
<td>5 Days</td>
<td>1060</td>
<td>5</td>
<td>0.4758</td>
<td>5</td>
<td>0.05</td>
</tr>
<tr>
<td>7 Days</td>
<td>2427</td>
<td>8</td>
<td>0.4669</td>
<td>10</td>
<td>0.001</td>
</tr>
<tr>
<td>10 Days</td>
<td>137</td>
<td>2</td>
<td>0.5549</td>
<td>5</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### 3.5 Labeling SB Data

In algorithm 1, we show the steps followed to label the bidders of a given cluster. A cluster belongs to a certain SB subset. As shown in Table 3.8, for each subset, we first compute the mean and STandard Deviation (STD) of each fraud pattern for all the instances in that subset. Then, we compute the average of the means (Avg. Means) and average of the STDs (Avg. STDs) of all the patterns for that subset. We consider the value of \((\text{Avg.Mean} + \frac{1}{2}\text{Avg.STD})\) since it produces the best decision line that separates between normal and suspicious instances as depicted in Figure 3.9. Then, we calculate the average of the means of all the patterns for the cluster. Thus, if the average mean of the cluster is greater than the decision line of the subset, then instances are labeled as suspicious (1) in that cluster, otherwise, they are labeled normal (0).
Table 3.8: SB Patterns’ Means and Standard Deviations for Each SB Dataset

<table>
<thead>
<tr>
<th>Subset</th>
<th>1 Day</th>
<th>3 Days</th>
<th>5 Days</th>
<th>7 Days</th>
<th>10 Days</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean of each pattern per subset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BT</td>
<td>0.1434</td>
<td>0.1394</td>
<td>0.1419</td>
<td>0.1455</td>
<td>0.1162</td>
</tr>
<tr>
<td>BR</td>
<td>0.1287</td>
<td>0.1328</td>
<td>0.1235</td>
<td>0.1273</td>
<td>0.1021</td>
</tr>
<tr>
<td>SO</td>
<td>0.0996</td>
<td>0.1047</td>
<td>0.0872</td>
<td>0.1149</td>
<td>0.0620</td>
</tr>
<tr>
<td>LB</td>
<td>0.4624</td>
<td>0.4511</td>
<td>0.4676</td>
<td>0.4678</td>
<td>0.4746</td>
</tr>
<tr>
<td>EB</td>
<td>0.4314</td>
<td>0.4192</td>
<td>0.4318</td>
<td>0.4348</td>
<td>0.4575</td>
</tr>
<tr>
<td>WR</td>
<td>0.3812</td>
<td>0.3718</td>
<td>0.3810</td>
<td>0.3533</td>
<td>0.3496</td>
</tr>
<tr>
<td>AB</td>
<td>0.2120</td>
<td>0.1936</td>
<td>0.2403</td>
<td>0.2567</td>
<td>0.2926</td>
</tr>
<tr>
<td>ASP</td>
<td>0.5007</td>
<td>0.4301</td>
<td>0.4478</td>
<td>0.4801</td>
<td>0.7123</td>
</tr>
<tr>
<td>Avg. Means</td>
<td>0.2949</td>
<td>0.2802</td>
<td>0.2901</td>
<td>0.2975</td>
<td>0.3208</td>
</tr>
<tr>
<td></td>
<td>STD of each pattern per subset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BT</td>
<td>0.1973</td>
<td>0.1884</td>
<td>0.1984</td>
<td>0.2019</td>
<td>0.1811</td>
</tr>
<tr>
<td>BR</td>
<td>0.1246</td>
<td>0.1330</td>
<td>0.1243</td>
<td>0.1377</td>
<td>0.1165</td>
</tr>
<tr>
<td>SO</td>
<td>0.2764</td>
<td>0.2811</td>
<td>0.2583</td>
<td>0.2917</td>
<td>0.2215</td>
</tr>
<tr>
<td>LB</td>
<td>0.3773</td>
<td>0.3753</td>
<td>0.3917</td>
<td>0.3783</td>
<td>0.3931</td>
</tr>
<tr>
<td>EB</td>
<td>0.3775</td>
<td>0.3742</td>
<td>0.3921</td>
<td>0.3802</td>
<td>0.3968</td>
</tr>
<tr>
<td>WR</td>
<td>0.4356</td>
<td>0.4373</td>
<td>0.4402</td>
<td>0.4345</td>
<td>0.4398</td>
</tr>
<tr>
<td>AB</td>
<td>0.2323</td>
<td>0.2426</td>
<td>0.2646</td>
<td>0.2658</td>
<td>0.2575</td>
</tr>
<tr>
<td>ASP</td>
<td>0.4931</td>
<td>0.4831</td>
<td>0.4863</td>
<td>0.4908</td>
<td>0.4510</td>
</tr>
<tr>
<td>Avg. STDs</td>
<td>0.3142</td>
<td>0.3143</td>
<td>0.3194</td>
<td>0.3226</td>
<td>0.3071</td>
</tr>
</tbody>
</table>
Require: AvgMeans and AvgSTDs of the corresponding SB subset

1: Compute MeanCluster
2: if (MeanCluster ≥ (AvgMeans + \frac{AvgSTDs}{2})) then
3: for x=1 to NumberBiddersCluster do
4: LabelBidder_x = 1 (Suspicious)
5: end for
6: else
7: for x=1 to NumberBiddersCluster do
8: LabelBidder_x = 0 (Normal)
9: end for
10: end if

Algorithm 1: : Labeling Bidders in a Cluster

To validate our approach, we choose randomly 5 auctions among the 7-day duration auctions (in total 309) and select randomly one bidder in each auction (Table 3.9). As we can observe from this table, the SB instances were successfully labeled by our approach. For example, bidder “g***r” has 4 fraud patterns with very high values; among them 2 have a high weight and 1 a medium weight. Therefore, the activity of this bidder in auction ID # 2370 is suspicious. On the other hand, the bidder “k***a” has all his fraud patterns with very low values; this indicates that this bidder behaved normally in the auction ID# 900. All these results are consistent with the labels produced by our approach. Table 3.10 provides all the final results of the labeling task of our SB training dataset. There are 5646 instances categorized as normal and 675 instances as suspicious.
Figure 3.9: Decision line of a subset and its labeled clusters

Table 3.9: SB Instances and their Labels

<table>
<thead>
<tr>
<th>AuctionID</th>
<th>BidderID</th>
<th>BT</th>
<th>BR</th>
<th>SO</th>
<th>LB</th>
<th>EB</th>
<th>WR</th>
<th>AB</th>
<th>ASP</th>
<th>Generated Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1009</td>
<td>z***z</td>
<td>0.75</td>
<td>0.3461</td>
<td>1</td>
<td>0.5667</td>
<td>0.5409</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>900</td>
<td>k***a</td>
<td>0.4705</td>
<td>0.3076</td>
<td>0</td>
<td>0.1909</td>
<td>0.1909</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2370</td>
<td>g***r</td>
<td>0.8333</td>
<td>0.2</td>
<td>1</td>
<td>0.0350</td>
<td>0.0239</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>432</td>
<td>0***0</td>
<td>0.5</td>
<td>0.3333</td>
<td>0</td>
<td>0.2199</td>
<td>0.0043</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1370</td>
<td>o***-</td>
<td>0.04615</td>
<td>0.0857</td>
<td>0</td>
<td>0.2966</td>
<td>0.2060</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.10: Final Results of Instance Labeling

<table>
<thead>
<tr>
<th>SB Subset</th>
<th>1 Day</th>
<th>3 Days</th>
<th>5 Days</th>
<th>7 Days</th>
<th>10 Days</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Normal Instances</td>
<td>1135</td>
<td>1303</td>
<td>975</td>
<td>2098</td>
<td>135</td>
<td>5646</td>
</tr>
<tr>
<td>No. Suspicious Instances</td>
<td>154</td>
<td>105</td>
<td>85</td>
<td>329</td>
<td>2</td>
<td>675</td>
</tr>
</tbody>
</table>
3.6 Conclusion

There are limited classification studies on the SB fraud due to the difficulty of producing training data on one hand and labeling multi-dimensional instances on the other hand. Our aim in this chapter is to efficiently label SB instances based on the hierarchical clustering CURE that showed a remarkable capability for partitioning the online behavior of bidders. First, we divided the SB dataset into several subsets according to the different bidding durations of the auctions that we scraped from eBay. Then, we efficiently partitioned each SB subset into clusters of users with similar bidding behaviour. At last, we employed a systematic labeling approach to each cluster to classify bidders into normal or suspicious. The important research directions, as presented in the red area in Figure 3.10 are: The generated SB dataset is highly imbalanced, which will negatively impact the performance of classifiers. In our next chapter, we will investigate this problem by testing different types of techniques, such as data sampling and cost-sensitive learning, to determine the most suitable technique for our SB dataset.

![Figure 3.10: High-Quality SB Dataset Construction and Classification](image)
Chapter 4

Instance-Incremental Classification of Imbalanced SB Data

Abstract: SB has been recognized as the predominant online auction fraud and also the most difficult to detect due to its similarity to normal bidding behavior. Previously, we produced a high-quality SB dataset based on actual auctions and effectively labeled the instances into normal or suspicious. To overcome the serious problem of imbalanced SB datasets, in this chapter, we investigate over- and under-sampling techniques (handling imbalance class on data level) through several instance-based classification algorithms. Thousands of auctions occur in eBay every day, which may be sent continuously to the optimal fraud classifier to detect potential SB activities. Consequently, instance-based classification is appropriate for our particular fraud detection problem, since it is proven to be capable of classifying data stream. According to the experimental results, incremental classification returns high performance for both over- and under-sampled SB datasets. Still, over-sampling slightly outperforms under-sampling for both normal and suspicious classes across all the classifiers.
4.1 Introduction

The labeled SB dataset that we obtained is however imbalanced alike the training datasets of any fraud detection applications. In the auction context, the number of normal bidders far outnumbers the number of shill bidders. An imbalanced dataset is a serious concern for the classification algorithms, because on one hand the predictive performance is deficient as shown in [GS17, YRR+14], and on the other hand, the decision boundary of classifiers are biased towards the majority class. Since the classifiers receive little information about the minority class, it is difficult to produce meaningful predictions for this class. This is deplorable because, in fraud detection problems, the minority class represents the fraudulent instances that are the primary target for investigation, since they have the highest misclassification rate. Two main methods have been proposed to address imbalanced learning problems [YRR+14]. The first method handles the skewed class distribution at the data level by re-sampling the majority and/or minority class. The second method is cost-sensitive classification that assigns weights to the classes; for example, a high weight to the minority class. In this chapter, we explore data sampling since it is very efficient for training datasets with medium size and dimensionality, like our SB dataset consisting of 6321 instances and 8 SB predictors.

There are two major data sampling methods, over-sampling and under-sampling. Generally speaking, the former way adds instances to the minority class while the latter eliminates instances from the majority class. Nevertheless, with oversampling, there is a potential of overfitting the classifiers whereas with under-sampling, important examples maybe discarded [Cha10, KK06]. The goal of this research is to determine the most suitable sampling method for our imbalanced SB training dataset and build the optimal fraud classification model. More precisely, we develop several instance-based classifiers on the under- and over-sampled SB datasets. We selected four classifiers, which are Locally Weighted Learning, KStar instance-based,
K-Nearest Neighbours, and Hoeffding Tree. Instance-based learning is a promising technique for the classification of the data stream and for problems that are dynamic and complex [ZGZG11]. Every day, thousands of auctions occur in commercial auction sites, such as eBay, which may be sent continuously to the classifier to detect potential SB activities. Consequently, instance-based classification is appropriate for our specific fraud detection problem, because this type of classification can effectively handle data stream.

The remaining of this chapter is as follows. Section 2 examines past studies on handling the imbalanced data problem. Section 3 implements the sampling techniques to the original SB dataset. Section 4 presents four instance-incremental classifiers as well as their quality metrics. Section 5 discusses the performance of each instance-based classifier on the re-balanced datasets and compare the four classifiers. Section 6 discusses the findings obtained from the experiments. Finally, Section 7 summarizes the findings of this study and highlights essential research directions as well.

### 4.2 Related Work

Different data sampling techniques have been investigated to overcome the problem of imbalanced classes of numerous applications as well as to assess the performance of different classifiers after sampling. For example, [YM12] presented a hybrid technique based on data sampling and re-weighting to enhance the performance of AdaBoost classifier. The idea is to apply first the over-sampling method SMOTE to re-balance the instance distribution, and then Genetic Algorithms to re-weight the classifier. The authors validated this proposed method with three quality metrics, G-Mean, True Positive Rate, and True Negative Rate. The experiments were carried out on different benchmark imbalanced datasets, Abalone, Cancer, and Heart Disease. The hybrid method positively influenced the AdaBoost classifier.
Another work [SAI13] evaluated the performance of Naive Bayes and then compared its outputs to SVM, C4.5, ANN and KNN that have been tested on the same datasets by other authors. Because some authors tend to modify it claiming that it needs improvement, the ultimate goal of this study is to analyze the classification capabilities of the original Naive Bayes. The experiments were conducted on three imbalanced UCI datasets, which are Herbasans Survival, German Credit Card and Pima Indian. Sampling techniques, such as SMOTE, are used by some classifiers to re-sample the datasets and to achieve higher predictive performance. The authors considered G-means as the comparison measurement between the classifiers. They observed that Naive Bayes does not provide better results. Thus, the conventional Naive Bayes need to be improved before it can be efficiently used on imbalanced datasets.

The study [Zho13] analyzed the impact of two sampling methods (SMOTE and Random Under-sampling) on two imbalanced datasets, USA Bankruptcy and Japanese Bankruptcy, to evaluate the performance of five bankruptcy prediction models (called CBPM). The first dataset includes 918 bankrupts and 85,211 non-bankrupt samples collected during the period 1981-2009 while the second dataset consists of 58 bankrupts and 36,578 non-bankrupt samples, extracted during 1988-2009. Sensitivity, Specificity, Accuracy, F-measure, and Area under ROC curve are the quality metrics for assessing the performance of the five tested models. The experimental results showed that under-sampling is preferable over over-sampling in case there are hundreds of bankrupt instances. On the other hand, over-sampling returns better outcomes when dealing with only tens of bankrupt instances.

In [YRR+14], sampling methods have been applied to the highly imbalanced Cardiac Surgery dataset that contains 4976 instances (with only 4.2% of people who died after surgery). The authors employed IBM SPSS Modeler 15.2 to randomly under- and over-sample the dataset. Based on Accuracy, Sensitivity, Specificity and Pre-
cision, they evaluated the performance of the CART, CHAID AND C5 classifiers. Then, they compared the outcomes of their proposed approach with the bagging and boosting methods for handling imbalanced data. Data sampling methods provided better results in the experiments, and are much easier to implement than bagging and boosting methods.

Moreover, [CCV14] combined SMOTE and Random Under-sampling to produce a balanced dataset without a severe loss of information and without adding a large number of synthetic instances. Two metal industry datasets have been considered in this study. Based on True Positive, True Negative, False Positive, False Negative, and Global Accuracy, the feasibility of the hybrid sampling method underwent several trials with SVM, Decision Tree, Self-Organizing Map (SOM) and Bayesian. The authors observed that SVM and SOM returned the best detection results and less false alarms.

In [ZSM15], the authors compared the performance of SVM, Naive Bayes and Decision Tree C4.5 classifiers on several imbalanced UCI datasets with different sizes and dimensionality (Fertility, User Knowledge Modeling, Vertebral Column, Seismic Bumps and Bank Marketing). They employed the dual data sampling technique implemented in WEKA to all the datasets. They conducted the experiments based on Sensitivity, Specificity, G-means, and Time-efficiency. They concluded that SVM outperforms the other classifiers.

More recently, [GS17] was the first work that dealt with the issue of imbalanced SB datasets where normal bidding behavior is more significant that SB behavior. Three classifiers were deployed, Naive Bayes, Decision Trees, and Neural Networks, to test a hybrid data sampling technique (SMOTE and Random Under-sampling). The authors utilized Recall, F-measure, MCC and AUC measurements to obtain a better view of the learning performance of each classifier before and after data sampling. They demonstrated that the accuracy significantly improves after sampling across all
the metrics and classifiers. Decision Trees outperformed the other classifiers.

Lastly, [JKE18] conducted a comparison of seven meta-classifiers on four liver toxicity datasets that have different imbalanced class ratios. The considered meta-classifiers are Bagging, Under-instanced stratified bagging, Cost-sensitive classifier, MetaCost, Threshold Selection, SMOTE and ClassBalancer. Random Forest is the only baseline classifier that has been applied to all the re-balancing techniques. The authors found out that the best performing sampling techniques for the case study are Stratified bagging, MetaCost and CostSensitiveClassifier. Also, Random Forest and Bagging as stand-alone classifiers were unable to handle the imbalance issue.

### 4.3 Sampling SB Data

As presented in Table 4.1, the resulting labeled SB dataset is imbalanced with a ratio of normal to suspicious instances equal to 8:1. This ratio means that the instance count in the normal class is much higher than in the fraud class. With imbalanced data, the prediction outcomes are not reliable due to several reasons [GS17]: 1) classifiers presume that instances are distributed equally between the two classes; 2) the decision boundary of classifiers is biased towards the majority class. In this case, the minority class, which is usually the primary target for investigation, will not be identified correctly; 3) the classification performance will be reduced, especially for the minority class.

<table>
<thead>
<tr>
<th></th>
<th>Imbalanced Dataset</th>
<th>Over-Sampled Dataset</th>
<th>Under-Sampled Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Instances</td>
<td>5646</td>
<td>5646</td>
<td>675</td>
</tr>
<tr>
<td>Suspicious Instances</td>
<td>675</td>
<td>5617</td>
<td>675</td>
</tr>
<tr>
<td>Total</td>
<td>6321</td>
<td>11263</td>
<td>1350</td>
</tr>
<tr>
<td>Ratio</td>
<td>≈ 8:1</td>
<td>≈ 1:1</td>
<td>1:1</td>
</tr>
</tbody>
</table>

One strategy to address the imbalanced learning problem is to employ data sam-
pling to rebalance the class distribution. There are two main sampling methods, oversampling the minority class and/or under-sampling the majority class. We choose the Synthetic Minority Over-sampling TECnhique (SMOTE) [CBHK02,FGHC18], which is the most widely utilized over-sampling method. SMOTE synthetically generates new instances in the minority class by using the feature space instead of the data space. The new synthetic data is generated based on the KNN rules rather than over-sampling data by duplication. Technically, each instance in the minority class is plotted in the feature space, and its K nearest neighbors are determined. Next, the difference between the instance and its neighbor is computed and then multiplied by a random number between 0 and 1. Finally, the resulting value is added between the instance and its neighbor.

Regarding the under-sampling technique, we select SpreadSubSample (built in WEKA) that randomly produces a subset of the dataset (see Appendix B). Thus, the whole dataset is required to fit in the memory. The user can specify the desired distribution ratio between the majority and minority class. Instances from both classes are chosen randomly. In our case study, we define the ratio of 1:1 to re-balance equally the two categories. In Table 4.1, we also present the class distribution of the two re-sampled SB training datasets.

4.4 Instance-Incremental Classification

In typical learning, called eager learning, the model generalization is carried out with all the training instances. An eager learning method utilizes the same approximation to the target function. The main idea is to develop a unique global model that is reliable for classifying unseen instances. In contrast, with lazy learning, also called just-in-time learning, the generalization is conducted each time an instance is presented to the classifier [SW11]. A lazy learning method builds different approximations to
the target function for each new instance. The lazy classifiers that we select in this paper are Locally Weighted Learning, KStar instance-based and K-Nearest Neighbours. In addition, we consider the Hoeffding Tree classifier since it is also based on instance-incremental learning, but it does not belong to the lazy classifiers family.

As mentioned in [HR08], there are two implementation strategies for the classification models, which are batch and incremental. With the former strategy, the models are built based on a whole batch of instances previously loaded in the memory. In the second strategy, the models are refined by learning instance by instance. So, the model updates itself when a new instance becomes available. The benefits of incremental classification are threefold [Bif16]: 1) consumes a limited amount of memory since the dataset is not entirely loaded into the memory; 2) performs the prediction of the new instance in a little amount of time; and 3) has the ability to predict at any point. [Bif16] claims that incremental classification is very fast and considered to be more efficient.

In our research, we first tried the batch strategy, but when we utilized SMOTE, the baseline classifiers were all disabled because of a large number of multi-features instances (11263 in total). However, the instance-based classifiers efficiently processed the over-sampled SB dataset. This way, we can conduct a fair comparison of the two sampling techniques. For all the experiments, we employ 10-fold cross-validation to avoid overfitting the learned models.

To obtain further insights into the results given by each classifier, we consider seven quality measurements given below:

1. True Positive Rate (TPR) represents the sensitivity of the classifier in identifying a given class.

\[
TPR = \frac{TP}{TP + FN}
\]
2. False Positive Rate (FPR) denotes the percentage of instances of a class that have been incorrectly identified.

\[
FPR = \frac{FP}{FP + TN}
\]

3. Precision reflects the ability of a classifier to precisely identify instances of a class.

\[
Precision = \frac{TP}{TP + FP}
\]

4. F-Score (F1) is the harmonic mean of Precision and Recall.

\[
F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}
\]

5. Matthews Correlation Coefficient (MCC) is a correlation coefficient between the predicted and observed instances. It can be used for imbalanced data.

\[
MCC = \frac{[(TP \cdot TN) - (FP \cdot FN)]}{\left[(TP + FP)(TP + FN)(TN + FP)(TN + FN)\right]^{\frac{1}{2}}}
\]

6. Receiver operating characteristic (ROC) is used to plot the values of TPR in the function of FPR for different cut-off points. Points on the ROC curve describe a sensitivity/specificity pair corresponding to a decision threshold.

7. Precision-Recall curve (PRC) is used to plot precision values for corresponding recall values for all potential cut-offs for a test.
4.5 Classifying SB Data

To determine which sampling technique is the most suitable for our SB training dataset, we have to expose the desired instance-based classifiers to both datasets (see Appendix B). The possible settings that are available for each classifier are presented in Table 4.2. Based on our experiments, we show in bold the ones we have employed, since they provide the best outcomes.

Table 4.2: Instance-Incremental Classifiers Settings

<table>
<thead>
<tr>
<th>Instance Classifier</th>
<th>Underlying Classifier</th>
<th>Nearest Neighbor Search Algorithm</th>
<th>Distance Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWL</td>
<td>Linear Regression, Naive Bayes, SVM, Decision Trees (like Decision Stump Tree)</td>
<td>Ball Tree, Cover Tree, Filtered Neighbor Search, KD Tree and LNN Search</td>
<td>Manhattan, Filtered, Euclidean, Chebyshev and Minkowski</td>
</tr>
<tr>
<td>KStar</td>
<td>KStar</td>
<td>NA</td>
<td>Entropy-based distance</td>
</tr>
<tr>
<td>KNN</td>
<td>IBK</td>
<td>Ball Tree, Cover Tree, Filtered Neighbor Search, KD Tree and LNN Search</td>
<td>Manhattan, Filtered, Euclidean, Chebyshev and Minkowski</td>
</tr>
<tr>
<td>HT</td>
<td>VFDT: Leaf prediction using Naive Bayes Adaptive, Naive Bayes and Majority Class</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

4.5.1 Locally Weighted Learning

The first instance-incremental algorithm examined in this study is the Locally Weighted Learning (LWL) that employs an approximated local approach to assign weights to instances [AMS97]. This classifier predicts the instance label according to its neighbouring data points. Each neighbouring instance becomes a weighting factor that
reflects a positive or negative impact on classifying the new instance; the closest
instances are given higher weights than the faraway ones [Eng12]. The standard ap-
proach of LWL is called Memory-based Locally Weighted Regression in which the
batch setting is used. The other approach is the Locally Weighted Projection Re-
gression, which is an incremental learning method. The idea here is to keep storing
each learned model for the future prediction instead of discarding each model after
providing a prediction as done by the former approach [Eng12].

LWL belongs to the lazy classifiers family. In WEKA, the user is able to specify
the underlying classifier such as Linear Regression, Decision Stump Tree and Naive
Bayes. Additionally, the user can select the nearest neighbour search algorithm, such
as Ball Tree, Linear Nearest Neighbor (LNN) and Cover Tree. In our case, we consider
the Decision Stump Tree as the base classifier for prediction and LNN algorithm to
measure the distance between the $k^{th}$ neighbour and the new instance. In this search
algorithm, the user can choose the distance function, such as Manhattan distance,
Filtered distance, Euclidean distance and Chebyshev distance. In our study, we select
Euclidean distance.

Table 4.3 shows that the LWL classifier slightly provides better results with over-
sampling than under-sampling. With over-sampling, the results are very satisfactory
with TPRs equal to 97.1% and 100% for the normal and suspicious class respectively.
The total number of incorrectly classified normal instances is 165, and we have only
two wrongly predicted suspicious instances. On the other hand, 96.7% of normal
instances and 99.7% of suspicious instances were correctly predicted with under-
sampling.

Also, there are 22 (3.3%) of normal instances and 2 (0.3%) of suspicious instances
that were misclassified as presented in Table 4.4. The F-Score and MCC show a
minor difference between over- and under-sampling where the gap between the two
techniques is only 0.3% for F-score and 0.6% for MCC. Lastly, out of 11263 instances, 167 (1.48%) were misclassified when using over-sampling. In contrast, 24 of the 1350 instances were classified incorrectly with 1.78% overall error rate. As shown in Figure 4.1, the classification performances of LWL on both sampled datasets are highly accurate. Yet, the ROC and PRC confirm the slight superiority of over-sampling over under-sampling technique.

![Figure 4.1: ROC of LWL on over- and under-sampled dataset](image)

Table 4.3: LWL Performance on Sampled SB Datasets

<table>
<thead>
<tr>
<th>Class</th>
<th>Over-Sampling</th>
<th>Under-Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Suspicious</td>
</tr>
<tr>
<td>TPR</td>
<td>0.971</td>
<td>1</td>
</tr>
<tr>
<td>FPR</td>
<td>0.000</td>
<td>0.029</td>
</tr>
<tr>
<td>Precision</td>
<td>1.000</td>
<td>0.971</td>
</tr>
<tr>
<td>F1</td>
<td>0.985</td>
<td>0.985</td>
</tr>
<tr>
<td>MCC</td>
<td>0.971</td>
<td>0.971</td>
</tr>
<tr>
<td>ROC</td>
<td>0.996</td>
<td>0.996</td>
</tr>
<tr>
<td>PRC</td>
<td>0.997</td>
<td>0.992</td>
</tr>
</tbody>
</table>
Table 4.4: Data Sampling Confusion Matrix Produced by LWL

<table>
<thead>
<tr>
<th></th>
<th>Over-Sampling</th>
<th></th>
<th>Under-Sampling</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Classified</td>
<td>Normal</td>
<td>Suspicious</td>
<td>Normal</td>
</tr>
<tr>
<td>True</td>
<td>5481</td>
<td>5615</td>
<td>653</td>
<td>673</td>
</tr>
<tr>
<td>False</td>
<td>165</td>
<td>2</td>
<td>22</td>
<td>2</td>
</tr>
</tbody>
</table>

4.5.2 KStar Instance-Based

KStar classifier is another lazy learning method, which is an instance-based learner too. The entropy distance is used to determine the similarity between data points based on the probability of transforming one data point into another and a random selection between all possible transformations [CT95]. The entropy distance of KStar is defined as:

\[ K^*(x|y) = -\log_2 P^*(x|y) \]

where \( P^* \) is a probability function of all paths from instance \( x \) to instance \( y \). The pros of utilizing this distance measurement is that it can deal with real, symbolic and missing values of features [CT95].

With this classifier, WEKA enables users to define the most appropriate setting for handling their datasets. For instance, the user can select one of the four options to instruct how the classifier deal with missing values: 1) ignoring instances with missing values; 2) treating missing values as maximally different; 3) normalizing over the features; 4) averaging the column entropy curve. The last option is the default one that we consider in our experiment. Also, we keep the global blend parameter as 20% since it provides good results in our study and seemingly runs well for most datasets as claimed in [CT95]. This specific parameter value is used as the default
number in both WEKA and by the developer of KStar as well.

Table 4.5: KStar Performance on Sampled SB Datasets

<table>
<thead>
<tr>
<th>Class</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.978</td>
<td>0.998</td>
</tr>
<tr>
<td>FPR</td>
<td>0.002</td>
<td>0.022</td>
</tr>
<tr>
<td>Precision</td>
<td>0.998</td>
<td>0.978</td>
</tr>
<tr>
<td>F1</td>
<td>0.988</td>
<td>0.988</td>
</tr>
<tr>
<td>MCC</td>
<td>0.976</td>
<td>0.976</td>
</tr>
<tr>
<td>ROC</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>PRC</td>
<td>0.999</td>
<td>0.999</td>
</tr>
</tbody>
</table>

Table 4.6: Data Sampling Confusion Matrix Produced by KStar

<table>
<thead>
<tr>
<th>Classified</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
</table>
| Over-Sampling
| True      | 5520   | 5605       |
| False     | 126    | 12         |
| Under-Sampling
| True      | 653    | 661        |
| False     | 22     | 14         |

Table 4.5 shows that KStar performs better with the over-sampled dataset rather than with the under-sampled one. The TPRs of normal and suspicious instances are 97.8% and 99.8% respectively with over-sampling. Only 2.2% of normal instances and 0.02% of suspicious instances were erroneously identified where the count of normal instances is 126 and the count of suspicious instances is 12 as shows in Table 4.6. The results exhibited in the same table show that, with under-sampling, there are 96.7% of
normal instances and 97.9% of suspicious instances were correctly predicted. So, the incorrectly predicted instances are 3.3% for normal instances and 2.1% for suspicious instances. The total number of misclassified instances on the under-sampled dataset is 36 out of 1350 (See Table 4.6). The F-Score on over-sampling reached 98.8%, while on under-sampling it was 97.3% with about 1.5% difference. MCC shows that KStar worked better with over-sampling than under-sampling by 2.9%. While the overall error rate with using under-sampling is 2.67%, only 1.23% with over-sampling technique. Figure 4.2, presents the classification performances of KStar on both sampled datasets which are high. Also, the average ROC and PRC results, given in Table 4.5, confirm the slight superiority of over-sampling over under-sampling technique.

4.5.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is one of the most common non-parametric classifiers that have been successfully utilized in knowledge discovery and machine learning [GDCH12]. With KNN, an instance is given a prediction according to the majority voting of its $K^{th}$ neighbors, where the $K^{th}$ data points are determined by the similarity distance measurement. This classifier is a member of the lazy learning family, which is also an instance-based learner.
KNN is implemented by the IBK algorithm in WEKA. Based on the cross-validation, IBK algorithm can determine the optimal value of $K$. Also, the user is able to choose between five options of the nearest neighbors search algorithms, which are Ball Tree, Cover Tree, Filtered Neighbor Search, KD Tree and LNN Search (default). For each of nearest neighbors search algorithms, the user can choose the distance function, such as the Euclidean distance (default in all the options). Moreover, with the IBk setting, we can define the desired number of $KNN$ and distance weighting. We use with the default setting, LNN search algorithm based on Euclidean distance, no distance weighting option, and the number of $KNN = 1$, since they produce the best outcome for our sampled SB datasets.

Table 4.7: KNN Performance on Sampled SB Datasets

<table>
<thead>
<tr>
<th>Class</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.983</td>
<td>0.995</td>
</tr>
<tr>
<td>FPR</td>
<td>0.005</td>
<td>0.017</td>
</tr>
<tr>
<td>Precision</td>
<td>0.995</td>
<td>0.983</td>
</tr>
<tr>
<td>F1</td>
<td>0.989</td>
<td>0.989</td>
</tr>
<tr>
<td>MCC</td>
<td>0.978</td>
<td>0.978</td>
</tr>
<tr>
<td>ROC</td>
<td>0.997</td>
<td>0.997</td>
</tr>
<tr>
<td>PRC</td>
<td>0.997</td>
<td>0.993</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.970</td>
<td>0.967</td>
</tr>
<tr>
<td>FPR</td>
<td>0.033</td>
<td>0.030</td>
</tr>
<tr>
<td>Precision</td>
<td>0.968</td>
<td>0.970</td>
</tr>
<tr>
<td>F1</td>
<td>0.969</td>
<td>0.969</td>
</tr>
<tr>
<td>MCC</td>
<td>0.938</td>
<td>0.938</td>
</tr>
<tr>
<td>ROC</td>
<td>0.986</td>
<td>0.986</td>
</tr>
<tr>
<td>PRC</td>
<td>0.985</td>
<td>0.981</td>
</tr>
</tbody>
</table>

Also, KNN classifier performs better with over-sampling than with under-sampling, as shown in Table 4.7. The classification results obtained using over-sampling are high, where the TPR is 98.3% for the normal instances and 99.5% for the suspicious instances. Thus, 1.7% of normal instances are inaccurately classified, while, the per-
Table 4.8: Data Sampling Confusion Matrix Produced by KNN

<table>
<thead>
<tr>
<th>Classified</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>5549</td>
<td>5588</td>
</tr>
<tr>
<td>False</td>
<td>97</td>
<td>29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classified</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>655</td>
<td>653</td>
</tr>
<tr>
<td>False</td>
<td>20</td>
<td>22</td>
</tr>
</tbody>
</table>

The percentage of misclassified suspicious instances is only 0.05%. Under-sampling TPRs outputs demonstrate 97% of the normal instances, and 96.7% of the suspicious instances were accurately predicted. Therefore, there are 3% of the normal instances, and 3.3% of the suspicious instances were incorrectly predicted. The average of F-Score on over-sampling is 98.9%, while on under-sampling is 96.9%. The average of the MCC measurement shows a considerable difference between over- and under-sampling techniques, where KNN works better with over-sampling than under-sampling by 4%. There are 126 misclassified instances out of 11263, so the overall error rate is around 1.12%. On the other side, the overall error rate with using under-sampling hits 3.11%, as demonstrated in Table 4.8. Again, the superiority of over-sampling technique over under-sampling is given in Figure 4.3, where the ROC is 99.7% for the over-sampled dataset and 98.6% for the under-sampled one.

Figure 4.3: ROC of KNN on over- and under-sampled dataset
4.5.4 Hoeffding Tree

Hoeffding Tree (HT), also called VFDT, is a decision tree based on the instance-incremental learner approach. It is able to learn from a large amount of data stream. With this classifier, a simple instance is sufficient enough to select the best dividing attribute by using the Hoeffding bound. The latter provides a certain level of confidence on the optimal instance to split the tree. Consequently, the model is built on the quantified number of instances determined by Hoeffding bound [DH00].

According to the training datasets, users are able to set the most suitable options for the VFDT classifier implemented in WEKA. For example, the user can set the number of instances that a leaf should observe between the splitting attempts where the default number is 200 instances. Also, in case of two or more instances having similar measurement outcomes, there is a need to break this tie so that other instances can decide about the best instance to choose. Thus, WEKA enables users to specify the breaking tie threshold (default threshold is 0.05). For the leaf prediction strategy, three options are available: 1) Naive Bayes Adaptive (default); 2) Naive Bayes; 3) Majority Class. The default minimum weight fraction for information gain splitting is 0.01; and users are allowed to change it. The given number of splitting confidence is 1.0E-7, which is the acceptable error in the splitting decision. Finally, two splitting criterion that users can choose are: information gain splitting (default) and Gini splitting. We examine different options of settings starting with the default ones on both sampled SB datasets. All the experiment attempts did not provide better results than the default settings. Consequently, the results below are based on the default settings.

Unlike previously discussed classifiers, HT behaves better with under-sampling than over-sampling in the overall results (Table 4.9). The classifier provides remarkable results when applying under-sampling where the Recall percentages are high on both normal and suspicious instances, 96.7% and 99.7% respectively. Thus, the
proportions of the miss-classified normal and suspicious instances are just 3.3% and 0.3% respectively. When applying over-sampling, the produced Recall results are 98.5% and 93.7% on the normal and suspicious instances sequentially. Therefore, the FPRs are 1.5% for the normal class and 6.3% for the suspect class. Interestingly, the TPR when using over-sampling is higher than when using under-sampling on the normal instances and vice versa on the suspicious ones. Since the minority class is the suspicious class, we consider under-sampling better in this case. The F-measurement on over-sampling reached 96.1%, while on under-sampling it achieved 98.2%. The MCC metric shows a considerable difference between under- and over-sampling techniques; HT performs better with under-sampling than with over-sampling by 4.2%. As presented in the confusion matrix of Table 4.10, 24 instances were miss-classified out of 1350 instances in under-sampling, i.e. 1.78% of overall error rate. In the over-sampling confusion matrix, the overall error rate is 3.89%, where the total number of incorrectly classified instances is 438. As given in Figure 4.4, the classification per-
Table 4.10: Data Sampling Confusion Matrix Produced by HoeffdingTree

<table>
<thead>
<tr>
<th>Classified</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>5560</td>
<td>5265</td>
</tr>
<tr>
<td>False</td>
<td>86</td>
<td>352</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classified</th>
<th>Normal</th>
<th>Suspicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>653</td>
<td>673</td>
</tr>
<tr>
<td>False</td>
<td>22</td>
<td>2</td>
</tr>
</tbody>
</table>

Performances of LWL on both sampled datasets are highly considerable to be used for SB detection model where, generally, the over-sampling performed better. However, the suspicious class results for PRC confirm the slight superiority of under-sampling.

Figure 4.4: ROC of HT on over- and under-sampled dataset

4.6 Discussion

As we can observe, across all the experiments, the four incremental classifiers produced very satisfactory results on the seven performance metrics for both normal and suspicious classes. This type of learning provides a high capability for classifying SB data with both over- and under-sampling techniques, as shown in Table 4.5. According to the results, we highlight the following findings:

1. Predicting with under-sampling is not always the preferred technique even
though it provides a better outcome with the HT classifier.

2. For the identification of suspicious instances, LWL combined with over-sampling is the most performing. We may mention that KStar and KNN combined with over-sampling, and HT combined with under-sampling attained the second best True Positive Rate.

3. Regarding normal instances, HT and KNN combined with over-sampling returned the best accuracy. Also, KStar combined with over-sampling is the second best.

Figure 4.5: Instance-incremental classifiers performance over the sampled datasets
All the results above indicate that over-sampling is better than under-sampling for our SB training dataset.

4.7 Conclusion

SB training datasets of online auctions are highly imbalanced. Imbalanced data have proved to deteriorate the accuracy of the classification models, especially for the minority class. This is a serious concern in fraud detection problems where the minority class, representing the fraud class, has the highest misclassification cost. To the best of our knowledge, this is the first time, the effect of imbalanced data is investigated with incremental classification algorithms. In our empirical study, we have examined the performance of several instance-based classifiers combined with oversampling and under-sampling techniques. Both sampling techniques provided high prediction results for both normal and suspicious classes across all the performance metrics. Over-sampling slightly outperforms under-sampling for all the selected learning classifiers. Even though the examined classification measurements showed that HT combined with over-sampling provided excellent results for the normal instances, the produced outcomes on the suspicious instances were exceedingly better when using under-sampling.

Our next work consists of examining the impact of cost-sensitive classification using instant-incremental classifiers to address the imbalanced learning problem of our SB dataset and then comparing their predictive performance with the data sampling performance.
Chapter 5

CSL-based on

Instance-Incremental Classification

Abstract: One of the most common form of cyber-crimes in e-auctions is shill bidding (SB) since this behaviour is difficult to detect, as it looks similar to normal bidding behaviour. To address this challenge, we first generate high-quality SB data from actual auctions and effectively label the multi-dimensional data. A serious issue that arose regarding the labeled SB dataset is that it is imbalanced. So, in this study, we examine the feasibility of the Cost-Sensitive Learning (CSL) to the imbalanced fraud data. Then, we compare the performance of CSL and data sampling, both produced through instance-based classification algorithms. Instance-based learning is appropriate for our fraud detection problem since this type of classification is useful for data streaming, such as bidding transactions. According to the experimental results, instance-based classification returns a high accuracy for both techniques. However, data sampling slightly outperforms CSL regarding the fraudulent class across all the classifiers.
5.1 Introduction

5.1.1 Research Scope

E-auctions are rated one of the most profitable e-commerce applications. In the last two decades, we witnessed a significant increase in auctioning products of different varieties. Nevertheless, e-auctions still remain vulnerable to cyber-crimes [Org16] due to three main reasons: the high anonymity of users, the low cost of auction services, and the high flexibility in terms of bidding. Auctions are an attractive environment for dishonest moneymakers who commit three kinds of fraud: pre-auction (ex. misrepresentation of products), post-auction (ex. non-delivery of products) and in-auction (ex. shill bidding and bid shielding) fraud. Our primary interest is on shill bidding (SB), the most common form of auction fraud but the most difficult to detect since a shill bidder can mimic normal bidding behaviour [FXV12]. SB is conducted by the product seller and/or colluding bidders who place multiple bids through phony accounts with the sole purpose of increasing the payoff for the seller. This type of fraud leaves no visible evidence, unlike pre- and post-auction fraud. Indeed, buyers remain unaware that they have been cheated and overcharged. Several empirical studies have shown the presence of SB activities across different commercial auction sites [FXV12,SW17].

Additionally, lawsuits have been filed against dishonest sellers and collaborators who committed SB [AS18]. The fraud resulted in substantial financial losses for the winning buyers. Research on SB fraud is limited, as it is difficult to generate SB data due to the following challenges: 1) identifying pertinent SB strategies; 2) determining robust metrics to measure the SB patterns, 3) scraping data from commercial auction websites, 4) preprocessing the auction data, and 5) measuring the SB metrics against the extracted auction data. In our previous work [AS18], we developed a high-quality (unlabelled) SB dataset based on a large collection of auctions crawled from eBay,
the most popular auction platform.

While labelling a SB dataset with multi-dimensional features is a delicate operation, it is however critical for building reliable fraud classification models. For this reason, in [AS19a] we introduced a two-stage approach for properly labelling SB datasets based on a robust hierarchical clustering technique. Our new approach returned remarkable results, as shown in the experiments. The labelled SB dataset that we obtained is imbalanced, as the training datasets of any fraud-detection applications. In the auction context, the count of normal bidders far outnumbers the count of shill bidders. An imbalanced dataset is a serious concern for the classification algorithms because, on the one hand, the predictive performance is deficient, as shown in [GS17], while on the other hand, the classifiers’ decision boundaries are biased towards the majority class. Since the classifiers receive little information about the rare class, it is difficult to produce meaningful predictions for this class. This is unacceptable because, in fraud detection problems, the minority class, representing the fraudulent instances, is the primary target for investigation, as they have the highest impact. These instances are associated with the highest misclassification cost. Two main methods are capable of handling the imbalanced learning problem [YRR+14,LFMTH12]. The first method handles the skewed class distribution at the data level by re-sampling the majority and/or minority class. The second method uses Cost-Sensitive Classification (CSL), which assigns weights to the classes; for example, high weight to the fraud class.

5.1.2 Contributions

In this work, we are particularly interested in CSL, which considers the cost of misclassifying instances to provide robust prediction models by setting specific weights for classes or instances. The goal of this research is to examine the feasibility of the CSL approach for our imbalanced SB training dataset and to build an optimal fraud
classification model. Additionally, we compare the CSL accuracy with our previous experimental results on data sampling. We employ several instance-based classifiers: Locally Weighted Learning, KStar instance-based, K- Nearest Neighbours and Hoeffding Tree. Instance-based learning is a promising approach for the classification of data stream and problems that are dynamic and complex [ZGZG11]. Auctions are a typical example of the real-world data stream. Indeed, everyday, thousands of auctions happen on auction sites, such as eBay, and auction data can be sent continuously to the fraud classifier in order to detect potential SB activities. Consequently, instance-based classification is appropriate for our particular fraud detection problem.

There are numerous studies that tackled the issue of imbalanced learning but very few for classifying SB data. Some studies found out that data-level approaches outperform algorithm-level approaches [CCV14, MVCR+16], but others are the opposite [JKE18, MVCR+16]. There is no clear winner since all depend on the training datasets. Thus, our ultimate goal is to investigate which approach is the most suitable for our SB training dataset. We have two main contributions to this present work. We first compare the CSL performances provided by the different instance-incremental SB classifiers. It is worthwhile to explore the performance of each classifier and compare it with other classifiers. Previously in [AS19b], we conducted several experiments regarding instance-incremental learning that we applied to over- and under-sampled SB datasets. Hence, we can compare the CSL approaches against the data sampling approaches.

The structure of the paper is as follows. Section 2 describes notable studies on CSL applications to imbalanced data. Section 3 explains how bidding fraud data have been produced from commercial auctions. Section 4 describes the systematic labelling method performed on the generated SB data. Section 5 gives an overview of CSL and setups and explains how to customize it for our fraud detection problem. Section 6 discusses and compares the predictive performance of the CSL-based fraud
classifiers. Section 7 compares the two approaches, data sampling and CSL, with a focus on the suspicious class i.e. fraudulent bidders. Finally, Section 8 summarizes the findings of this study and highlights potential directions for future research.

5.2 Related Work

CSL has been approached from various angles, such as integrating it with classification algorithms to obtain reliable prediction models, comparing it with traditional data sampling methods, or combining it with other techniques to deal with the skewed class distribution. While reviewing previous studies, we focus on their motivations in applying CSL, their classification algorithms, types of datasets and evaluation metrics, and the observed contributions.

Starting with [BVdP09], the imbalanced class distribution in churn prediction is investigated to find the best solution for handling this issue. Only two performance measurements are considered: AUC and Lift. The study examined the classifiers’ performance increase on data sampling, boosting and weighted Random Forests compared to other standard modelling techniques. There are six real-life proprietary European churn modelling datasets have different churn rates. The applied classifiers are random forest and logistic regression. The contribution of this work is that the under-sampling is capable of improving the prediction accuracy, as well as confirming the finding mentioned in [WP03], that is, there is no general answer regarding which technique for handling imbalanced data is the best, as the answer depends on the applied classifiers and the case-dependent. Also, the Weighted Random Forests combined with CSL significantly outperforms the standard Random Forests. However, data sampling produced the best performance.

In [TZCK09], the authors applied a modified SVM classifier to handle the imbalanced class distribution adequately. The suggested SVM models are: 1) information
granules (granular computing represents information in the form of some aggregates) based learning utilized for SVM (GSVM-RU); 2) CSL for SVM (SVM-WEIGHT); 3) applying SVM to the over-sampled dataset; 4) running SVM on the under-sampled dataset. The performances of the suggested models are compared using various measurement metrics, including Precision, Recall, G-mean, F-measure, and ROC. The experiments are conducted over seven highly imbalanced datasets: Oil, Mammography, Satimage, Abalone, Yeast, Yeast. The imbalance ratios are 4.38%, 2.32%, 9.73%, 0.77%, 5.72%, 3.44%, 4.14%, respectively. It is worth mentioning that the source and detailed information about each dataset are missing. The researchers conclude that GSVM-RU achieved optimal performance. Also, SVM-WEIGHT is not as efficient as GSVM-RU, yet it can be the best choice if a dataset is not large.

Another study, presented in [TNGST10], involved experiments using two suggested methods, using both data sampling and CSL to deal with the issue of imbalanced class distribution. The first method combines CSL with data sampling. Initially, the datasets are re-balanced by applying sampling techniques, such as SMOTE, TLINK, RUS, and ROS. Then, the SVM classifier is used to the rebalanced datasets. Next, the produced outcomes are entered into a sigmoid function to compute the posterior probabilities. Finally, the Bayes risk criterion is applied to the proposed method in order to pursue the best model with minimum predicted costs. The second method treats the cost matrix as a hyper-parameter locally optimizes it, and then trains the model. The experiments involved 18 imbalanced datasets from UCI, where the imbalanced ratio ranged from 1.77 (lowest) to 64.03 (highest). G-mean is the only quality metric considered in this study. The researchers deduced that the first method successfully achieves the misclassification cost reduction in some cases only, while the second method’s performance is highly considerable.

Moreover, the research described in [LFMTH12] analyzed how data level proposals and algorithm proposals perform in terms of handling skewed datasets. The data
level is represented by over-sampling and its variant Edited Nearest Neighbor (ENN) rule (SMOTE and SMOTE+ENN). On the other hand, CSL represents the algorithm level. These techniques are applied as follows: 1) datasets are re-balanced using over-sampling, 2) CSL is applied to the original class distribution of the datasets, and 3) hybrid approaches that combine the first and second methods. The classifiers considered in this study are SVM, Fuzzy Hybrid Genetics-Based Machine Learning (FH-GBML), decision tree C4.5, and KNN, which are all applied to the three mentioned handling imbalanced techniques. The authors used 66 benchmark binary imbalanced datasets that have been made public on KEEL. The classification quality of each experiment is measured only by AUC. The three main observations obtained from this research are: 1) techniques for handling imbalanced datasets improve the overall classification quality; 2) the outcomes of each experiment indicate that there is no difference between the preprocessing methods and CSL, and they are both adequate for addressing the issue of imbalanced class distribution; and 3) hybrid models do not always perform better than standard approaches.

In [CZZ13], the authors suggested directly integrating the performance metrics (AUC and G-mean) into the CSL+SVM to enhance the classification quality by concurrently optimizing the best pair of feature subset, the essential parameters, and the misclassification cost. The proposed method is called MOCSSVM, which is applied to ten benchmark datasets from UCI, and the minimum ratio is 1:4, while the maximum ratio reached is 1:39. The performance of MOCSSVM is compared with different models, such as basic SVM with and without the feature selection, CSL+SVM with grid search, and CSL+SVM. As the authors stated, the MOCSSVM competed with those from the state-of-the-art models and showed promising results.

[KWS14] also proposed an ensemble of CSL decision trees to solve the imbalanced learning problem. An evolutionary algorithm is applied to select classifiers concurrently and assigns weights to the members for the fusion process. The experiment is
based on six benchmark datasets collected from the KEEL repository. The imbalance ratios are set manually to 1:10, 1:25, and 1:50. Then, the classifier outcomes are compared for each rate. The performance quality is examined by measuring the distance from the point on the ROC to the point of the optimal performance. The study shows that merging CSL decision trees with the random subspace in feature space improves recognition of the minority class, and as the authors claim, the ensemble of CSL decision trees provides promising results.

[MVCR+16] investigated a real-life problem: predicting as soon as possible the student dropout courses. The authors proposed using Interpretable Classification Rule Mining (ICRM) to identify which students require help. The dataset contains 419 students who enrolled in the Academic Unit Preparatoria at the Autonomous University of Zacatecas in Mexico. While 362 students continued courses, the rest of the students dropped out (13.6%). The evaluation metrics considered in this work are: accuracy, true positive rate, true negative rate, and geometric mean (GM). Our main focus is on experiment 3, where the authors compared different classifiers combined with different techniques for handling imbalanced class distribution, using the proposed approach. The applied classifiers are C45+SMOTE, SVM+SMOTE, C45+CSL, SVM+CSL, and hierarchical genetic fuzzy system based on Genetic Programming-SMOTE (GP-SMOTE). As stated in the article, the overall geometric mean (GM) shows better results at the early stages when applying the rebalancing and CSL techniques. However, CSL outperformed data sampling at the early stages, while data sampling outperformed CSL at the later stages.

Recently, [KHB+18] proposed a CSL+deep neural network to address the skewed class distribution (CoSen Classifier). The proposed model can automatically learn the optimal features that represent both the majority and minority classes. The parameters of the neural network and the class-dependent costs are optimized during the learning process. The experiment is conducted on six image classification datasets:
Melanoma Detection (1,300 instances), Coral Classification (2,055 instances), Object Classification (9,144 instances), Scene Classification (15,620 instances), Handwritten Digit Classification (70,000 instances), and Image Classification (60,000 instances). G-mean and F-score provide a measurement of performance in this study. The authors’ conclusions are: 1) the proposed method CSL+ deep convolutional neural network (CoSen CNN) outperforms the baseline method; 2) the CoSen Classifier is superior when compared with data sampling techniques; 3) CoSen is useful for both binary and multi-label classifications. It is worth mentioning that [ZL10] noted that many studies have proven that the CSL approach is not useful for multi-classification problems.

5.3 CSL for SB Classification

5.3.1 An Overview on CSL

CSL is popular method to handle the issue of imbalanced datasets (usually containing binary classes). CSL considers the costs of the misclassified instances (false positives and false negatives) in different classes (normal and suspicious classes in our fraud problem) [Dom99]. As it is known, fraud datasets are imbalanced, and in this situation, the classifier neglects the variable costs and treats the misclassification costs all the same. This is unacceptable in fraud detection applications where the misclassification errors possess different costs. Also, the produced classification model will be unreliable due to a high probability of misclassifying instances from the fraud class.

As stated in [LFMT12], three tactics have been suggested to address the implementation of CSL. The first technique involves re-scaling the training data, and this technique is considered to be the most common. In this approach, the distribution of the classes of the training dataset is re-sampled based on the cost matrix via applying over- or under-sampling, giving weights to instances, or by changing
the decision threshold. When using data sampling, there are possible disadvantages, such as eliminating valuable instances in the under-sampled dataset and overfitting in the over-sampled dataset [Gan12]. On the other hand, weights are given to the instances for each class, according to their misclassification cost (FP, or FN). Then, the weighted instances are provided to the cost-blind algorithm to predict the unseen instances. The second technique involves approaches that modify the learning process to implement a CSL classifier, such as decision tree induction, where different building-tree strategies are applied to minimize the misclassification cost. The variable cost is used to select the splitting feature for the data and to decide whether a subtree needs to be pruned. The third technique involves Bayes decision approaches, where instances lie in the class that has the minimum predicted cost.

It is mentioned in [ZL10] that studies that approach the misclassification cost problem can be divided into two strategies: 1) instance-dependent cost, where a dependent misclassification cost is assigned to each instance, and 2) class-dependent cost, where to each class a misclassification cost is assigned. Specifically, in WEKA, the CSL is developed based on these described strategies, where re-weighting instances using the mean of the total cost given to each class is the first way, and the second way is that the class is predicted using the minimum expected cost. One of the main difficulties of applying CSL is specifying the optimal weights for the cost matrix since these weights cannot be obtained from the datasets [KWS14].

5.3.2 CSL Setup

The labelled SB dataset has imbalanced binary classes, where the majority class is the normal class (+), and the minority class is the suspicious class (−). Let \( C(s, n) \) be the prediction cost of an instance in class \( s \) (suspicious) predicted to be in class
From the cost matrix presented in Table 5.1, the notation of $C(-, +)$ is the cost of a suspicious instance that predicted as normal, and $C(+, -)$ notation is the vice-versa case.

**Table 5.1: Cost Matrix in CSL**

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td>Normal</td>
<td>$C(+, +) = 0$</td>
<td>$C(-, +) = 1$</td>
</tr>
<tr>
<td>Suspicious</td>
<td>$C(+, -) = 2, 3, and 5$</td>
<td>$C(-, -) = 0$</td>
</tr>
</tbody>
</table>

The default CSL cost in WEKA is a $1 \times 1$ matrix set to 0. In our SB dataset, there are normal and suspicious instances; thus, we change the number of classes to 2, and we resize the cost matrix into $2 \times 2$. We provide weights in the cost matrix as follows: 0.0 for true positive and true negative instances, and 1.0 for false positive instances. Regarding the False Negative (FN) instances, as claimed in [ZL10, KWS14], to determine the cost of incorrect predictions, we need to ask the experts in the investigated domain. We analyzed several research papers on CSL for the fraud detection field, and found that there is no specific way to set the misclassification cost. For example, in [LFMTH12], the misclassification cost is set to the data imbalance ratio, whereas, in [LKZ10, LMZ14], the cost is set manually to different ratios such as 2, 3, 5, 10, 15, and 20. As presented in Table 5.1, by trial and error learning, we set the punishment as 2, 3, and 5 in case FPs are produced, since our primary focus is detecting fraudulent bidders. Then, we select the ratio that provides the best classification performance. The minimal expected cost parameter is set to false and random seed to 1.

In our work, we examine the feasibility of CSL approach to our SB dataset. Each experiment was conducted with 10-fold cross-validation to avoid over-fitting. Also, for each experiment, different instance-incremental classifiers were selected as the base classifiers in the CSL configuration, which are Locally Weighted Learning (LWL), K-Star, K-Nearest Neighbors (KNN), and Hoeffding Tree (HT). Note that the con-
figurations of each selected classifier can be modified as shown in Table 4.2. It is worth mentioning that we have examined different parameters’ values for each selected instance-based classifier and considered only the ones that produced the optimal classification quality.

To obtain further insights into the performance returned by each classifier, we consider seven quality measurements: True Positive Rate (TPR), False Positive Rate (FPR), Precision, F-Score (F1), Matthews Correlation Coefficient (MCC), Receiver Operating Characteristic (ROC) Area, Precision/Recall (PRC) Area, and the total cost which is defined as:

\[
TotalCost = FP \times C(+, -) + FN \times C(-, +)
\]

where FP denotes the number of false positives and FN false negatives.

5.4 CSL and Instance-based Classification

To determine which approach is the most suitable for handing the class imbalance of the SB dataset, we expose the same instance-based classifiers for both techniques, data sampling and CSL. The possible settings for each classifier are exposed in Table 4.2, and in bold, we show the ones we have used in the experiments.

5.4.1 Cost-Sensitive LWL

Tables 5.2, and 5.3 show that the best CSL+LWL performance is achieved at cost 3. As illustrated in Table 5.2, CSL+LWL classifier achieves remarkable results on the normal and suspicious classes. It returns 97.2% of TPR for the normal class, and almost 99.6% for the suspicious class. The FPR (or fall-out) shows that the incorrectly-classified instances in the suspicious class is only 0.4%, whereas about 2.8% of normal instances are misclassified. Precision for the two classes exposes a
Table 5.2: Performance of CSL+LWL where N and S denote normal and suspicious respectively

<table>
<thead>
<tr>
<th>$C_{FP}$</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>TPR</td>
<td>0.972</td>
<td>0.996</td>
<td>0.972</td>
</tr>
<tr>
<td>FPR</td>
<td>0.004</td>
<td>0.028</td>
<td>0.004</td>
</tr>
<tr>
<td>Precision</td>
<td>0.999</td>
<td>0.811</td>
<td>0.999</td>
</tr>
<tr>
<td>F1</td>
<td>0.986</td>
<td>0.894</td>
<td>0.986</td>
</tr>
<tr>
<td>MCC</td>
<td>0.885</td>
<td>0.885</td>
<td>0.886</td>
</tr>
<tr>
<td>ROC</td>
<td>0.991</td>
<td>0.991</td>
<td>0.991</td>
</tr>
<tr>
<td>PRC</td>
<td>0.998</td>
<td>0.938</td>
<td>0.998</td>
</tr>
<tr>
<td>Total Cost</td>
<td>163</td>
<td>165</td>
<td>176</td>
</tr>
</tbody>
</table>

Table 5.3: Confusion Matrix of CSL+LWL

<table>
<thead>
<tr>
<th>Total Cost</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>True</td>
<td>5489</td>
<td>672</td>
<td>5490</td>
</tr>
<tr>
<td>False</td>
<td>3</td>
<td>157</td>
<td>3</td>
</tr>
</tbody>
</table>

difference of about 19% between the two classes, which is a great difference, where the normal class is about 100%, and the suspicious class is about 81.2%. Also, there are 156 ($\approx 2.8\%$) normal instances and only 3 ($\approx 0.4\%$) suspicious instances that were misclassified, as presented in Table 5.3. The results of F-Score are very high for the normal class (98.6%) and reasonably high for the suspicious class (89.4%). However, the classification quality given by the MCC score is around 88.6%, which indicates that the CSL+LWL classifier performed reasonably well. Also, the performance of ROC area and PRC area are above 99%, except for the suspicious class where PRC area decreased by about 4.5%.

5.4.2 Cost-Sensitive KStar

CSL+KStar reached the best classification performance at cost 5, as presented in Tables 5.4, and 5.5. In general, the KStar classifier performed better than the LWL classifier, as shown in Table 5.4. CSL+KStar achieves high results on the normal
Table 5.4: Performance of CSL+KStar

<table>
<thead>
<tr>
<th>$C_{FP}$</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>TPR</td>
<td>0.988</td>
<td>0.827</td>
<td>0.983</td>
</tr>
<tr>
<td>FPR</td>
<td>0.173</td>
<td>0.012</td>
<td>0.105</td>
</tr>
<tr>
<td>Precision</td>
<td>0.979</td>
<td>0.890</td>
<td>0.987</td>
</tr>
<tr>
<td>F1</td>
<td>0.984</td>
<td>0.857</td>
<td>0.985</td>
</tr>
<tr>
<td>MCC</td>
<td>0.841</td>
<td>0.841</td>
<td>0.864</td>
</tr>
<tr>
<td>ROC</td>
<td>0.991</td>
<td>0.991</td>
<td>0.991</td>
</tr>
<tr>
<td>PRC</td>
<td>0.999</td>
<td>0.909</td>
<td>0.999</td>
</tr>
<tr>
<td>Total Cost</td>
<td>303</td>
<td>309</td>
<td>276</td>
</tr>
</tbody>
</table>

Table 5.5: Confusion Matrix of CSL+KStar

<table>
<thead>
<tr>
<th>Total Cost</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>True</td>
<td>5577</td>
<td>558</td>
<td>5550</td>
</tr>
<tr>
<td>False</td>
<td>117</td>
<td>69</td>
<td>71</td>
</tr>
</tbody>
</table>

class, where 97.7% of TPR for the normal class, and 95.7% for the suspicious class. The incorrectly classified instances in the suspicious class is 4.3% and only 2.3% for the normal class. Precision for the normal class is 99.5% and 83.1% for the suspicious class.

The confusion matrix in Table 5.5 demonstrates that only 131 (2.3%) normal instances and 29 (4.3%) suspicious instances were misclassified. The F-Score for the normal class is very high (≈ 98.6%), and it is relatively high for the suspicious class (≈ 89%). Furthermore, the classification quality given by the MCC score is about 87.8%, which denotes that the CSL+KStar classification is satisfactory. In general, the results for the ROC area and PRC area are above 99% for both classes. However, with regards to the suspicious class, the percentage of the PRC area dropped by 10%.

5.4.3 Cost-Sensitive KNN

As illustrated in tables 5.6, and 5.7, CSL+KNN produces the optimal performance at cost 3. Table 5.6 shows that CSL+KNN provides substantial results on the normal
and suspicious classes, where TPR is 98% and 91.6% respectively. The FPR shows
that the incorrectly-classified instances for the suspicious class is 8.4%, whereas only
2% of normal instances are misclassified. Precision for the normal class reached 99%,
and only 84.4% for the suspicious class.

There are 114 (≈ 2%) normal instances and 57 (≈ 8.4%) suspicious instances that
were misclassified, as presented in Table 5.7. The results of F-Score are very high
for both classes, where the normal class is 98.5%, and the suspicious class is 87.8%.
The classification quality provided by the MCC score is 86.4%. The outcomes for the
ROC area and PRC area results are around 98%, except with suspicious class, the
percentage of PRC area is decreased by 11%.

<table>
<thead>
<tr>
<th>$C_{FP}$</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>TPR</td>
<td>0.981</td>
<td>0.890</td>
<td><strong>0.980</strong></td>
</tr>
<tr>
<td>FPR</td>
<td>0.110</td>
<td>0.019</td>
<td><strong>0.084</strong></td>
</tr>
<tr>
<td>Precision</td>
<td>0.987</td>
<td>0.851</td>
<td><strong>0.990</strong></td>
</tr>
<tr>
<td>F1</td>
<td>0.984</td>
<td>0.870</td>
<td><strong>0.985</strong></td>
</tr>
<tr>
<td>MCC</td>
<td>0.855</td>
<td>0.855</td>
<td><strong>0.864</strong></td>
</tr>
<tr>
<td>ROC</td>
<td>0.980</td>
<td>0.980</td>
<td><strong>0.980</strong></td>
</tr>
<tr>
<td>PRC</td>
<td>0.997</td>
<td>0.884</td>
<td><strong>0.997</strong></td>
</tr>
<tr>
<td>Total Cost</td>
<td>253</td>
<td>285</td>
<td>391</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total Cost</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>True</td>
<td>5541</td>
<td>601</td>
<td>5532</td>
</tr>
<tr>
<td>False</td>
<td>74</td>
<td>105</td>
<td><strong>57</strong></td>
</tr>
</tbody>
</table>

### 5.4.4 Cost-Sensitive HT

As given in Tables 5.8, and 5.9, The best CSL+HT performance is achieved at cost
5. CSL+HT classifier produces highly accurate results on the normal and suspicious
Table 5.8: Performance of CSL+HT

<table>
<thead>
<tr>
<th>$C_{FP}$</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>TPR</td>
<td>0.974</td>
<td>0.975</td>
<td>0.969</td>
</tr>
<tr>
<td>FPR</td>
<td>0.025</td>
<td>0.026</td>
<td>0.003</td>
</tr>
<tr>
<td>Precision</td>
<td>0.997</td>
<td>0.819</td>
<td>1.000</td>
</tr>
<tr>
<td>F1</td>
<td>0.985</td>
<td>0.890</td>
<td>0.984</td>
</tr>
<tr>
<td>MCC</td>
<td>0.880</td>
<td>0.880</td>
<td>0.877</td>
</tr>
<tr>
<td>ROC</td>
<td>0.991</td>
<td>0.991</td>
<td>0.991</td>
</tr>
<tr>
<td>PRC</td>
<td>0.999</td>
<td>0.897</td>
<td>0.999</td>
</tr>
<tr>
<td>Total Cost</td>
<td>179</td>
<td>179</td>
<td>181</td>
</tr>
</tbody>
</table>

Table 5.9: Confusion Matrix of CSL+HT

<table>
<thead>
<tr>
<th>Total Cost</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified</td>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>True</td>
<td>5501</td>
<td>658</td>
<td>5473</td>
</tr>
<tr>
<td>False</td>
<td>17</td>
<td>145</td>
<td>2</td>
</tr>
</tbody>
</table>

classes, where TPR is 97% and 99.7%, respectively. The FPR illustrates that the incorrectly-classified instances in the suspicious class is around 0.3%, whereas the classification error is only 3% for the normal instance, as shown in Table 5.8. Precision outcomes for both classes are 100% for the normal class and only about 80% for the suspicious class.

There are 171 ($\approx 3\%$) normal instances and only 2 ($\approx 0.3\%$) suspicious instances that were misclassified, as presented in Table 5.9. The F-Score shows very high outcomes for the normal class (98.4%), while the score is 88.6% for the suspicious class. The classification quality provided by MCC score is 87.8%, which indicates that the performance of the HT classifier performed very well. In general, the ROC area and PRC area area around 99% for both classes, yet the percentage of PRC area decreased by around 9% for the suspicious class.
5.4.5 Discussion

The classification quality of all of the examined classifiers using CSL is quite high. In our decision-making process, the best CSL+classifier performance is selected based on the following strategy:

1. We first consider the classifiers with the less FPs and FNs (see Table 5.10).

2. If the results of the first point are similar between classifiers, then we consider the MCC and F1 metrics since they both take into account the accuracy of Precision and Recall (see Table 5.11).

3. If the previous points fail, then we compare the total cost of the classifiers (see Table 5.11).

<table>
<thead>
<tr>
<th>Table 5.10: Confusion Matrix of CSL+Classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWL</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>5490</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5532</td>
</tr>
<tr>
<td>57</td>
</tr>
<tr>
<td>KNN</td>
</tr>
<tr>
<td>5515</td>
</tr>
<tr>
<td>29</td>
</tr>
<tr>
<td>5475</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.11: CSL+Classifiers Performances on Suspicious Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSL+</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>F1</td>
</tr>
<tr>
<td>MCC</td>
</tr>
<tr>
<td>Total Cost</td>
</tr>
</tbody>
</table>

We can summarize the classifiers’ outcomes into four key points:

- Although HT provides the lowest FPs, LWL produces the highest accuracy for the suspicious class, across all the quality metrics. Besides, LWL has the lowest total cost and has only 3 FP instances, which indicates that this classifier is highly recommended for SB detection.
• The second-best classification outcomes were returned by the HT model, and therefore, it is also suitable for detecting SB. For instance, its classification quality is over 88% for the suspicious class as provided by the F1 metric. Also, HT possesses the second lowest total cost (181), and the best classifier for detecting the suspicious instances, where it reached 673 instances out of 675, as presented in Tables 5.10, and 5.11.

• Across all the quality metrics illustrated in Tables 5.10, and 5.11, the KStar is the third best classifier.

• Even though KNN returned the worst outcomes among all the experiments, still its accuracy results are acceptable.

5.5 Data Sampling vs. CSL

5.5.1 Comparison

In our previous work [AS19b], we examined the feasibility of over- and under-sampling techniques for tackling the imbalanced SB dataset. We conducted several experiments involving the same instance-incremental classifiers and compared the outcomes. The findings are as follows:

1. Both data sampling techniques presented high-performance results for both normal and suspicious classes. Though, over-sampling slightly outperformed under-sampling for all the selected learning classifiers.

2. For the classification of suspicious instances, LWL+over-sampling is the best performing. KStar and KNN combined with over-sampling, and HT with under-sampling attained the second best approaches.
3. HT and KNN classifiers combined with over-sampling gave the best accuracy for the normal class.

In this section, we compare the outcomes of the two examined techniques (data level and algorithmic level) for handling imbalanced data based on the instance-incremental classifiers. Here, we mainly concentrate on the suspicious class, since this class is the target for investigation.

Table 5.12: LWL: Data Sampling vs. CSL on Suspicious Class

<table>
<thead>
<tr>
<th>Technique</th>
<th>Over</th>
<th>Under</th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>1</td>
<td>0.997</td>
<td>0.996</td>
</tr>
<tr>
<td>FPR</td>
<td>0</td>
<td>0.003</td>
<td>0.004</td>
</tr>
<tr>
<td>Precision</td>
<td>0.971</td>
<td>0.968</td>
<td>0.812</td>
</tr>
<tr>
<td>F1</td>
<td>0.985</td>
<td>0.982</td>
<td>0.894</td>
</tr>
<tr>
<td>MCC</td>
<td>0.971</td>
<td>0.965</td>
<td>0.886</td>
</tr>
<tr>
<td>ROC Area</td>
<td>0.996</td>
<td>0.995</td>
<td>0.991</td>
</tr>
<tr>
<td>PRC Area</td>
<td>0.992</td>
<td>0.994</td>
<td>0.946</td>
</tr>
</tbody>
</table>

In general, the LWL classifier provides the highest prediction values across all the experiments involving different techniques for handling imbalanced data, as shown in Table 5.12. The TPR outcome for combining LWL with over-sampling is 100%, and it is over 99% for both under-sampling and CSL.

In Figure 5.1, we see that the LWL performance on data sampling is slightly better than on CSL, and the best classification is obtained by combining LWL with over-sampling. Precision score of LWL+over-sampling is 97.1%, which is decreased by 1.7%, 16.1% for under-sampling and CSL techniques respectively. F1 rating for the over-sampled dataset is higher than the score for the under-sampled dataset by 0.3% and higher than the score for CSL by around 10%. Furthermore, MCC shows that the accuracy of LWL combined with over-sampling is superior to the accuracy of under-sampling, by 0.6%, and by about 9% compared to CSL accuracy.

KStar performed very well with data sampling as well as CSL. Using the over-sampled dataset, the TPR reached 99.8%, and there is only around a 2% and a 4%
Table 5.13: KStar: Data Sampling vs. CSL on Suspicious Class

<table>
<thead>
<tr>
<th>Technique</th>
<th>Over</th>
<th>Under</th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.998</td>
<td>0.979</td>
<td>0.957</td>
</tr>
<tr>
<td>FPR</td>
<td>0.002</td>
<td>0.021</td>
<td>0.043</td>
</tr>
<tr>
<td>Precision</td>
<td>0.978</td>
<td>0.968</td>
<td>0.831</td>
</tr>
<tr>
<td>F1</td>
<td>0.988</td>
<td>0.973</td>
<td>0.890</td>
</tr>
<tr>
<td>MCC</td>
<td>0.976</td>
<td>0.947</td>
<td>0.878</td>
</tr>
<tr>
<td>ROC Area</td>
<td>0.999</td>
<td>0.990</td>
<td>0.991</td>
</tr>
<tr>
<td>PRC Area</td>
<td>0.999</td>
<td>0.985</td>
<td>0.898</td>
</tr>
</tbody>
</table>

difference when using the under-sampled dataset and CSL, respectively (Table 5.13).

Figure 5.2 demonstrates that the CSL+KStar classifier slightly under-performs, compared to data sampling. Again, the best outcomes from this classifier are achieved with over-sampling, where the Precision score of 97.8%. Furthermore, Precision shows that the other techniques perform relatively well, as the score declined only by 1% for the under-sampled dataset and about 14.7% for CSL. The F1 score for the over-sampled dataset is 98.8%, which is higher than the scores achieved by 1.5% when using under-sampling and by 9.8% when using CSL. The MCC returned excellent results for both the re-sampled datasets. However, with regards to CSL, this rate
Figure 5.2: KStar Performance on imbalanced data techniques

drops by at least 7.4% as compared to the MCC rate for under-sampling.

Table 5.14: KNN: Data Sampling vs. CSL on Suspicious Class

<table>
<thead>
<tr>
<th>Technique</th>
<th>Over</th>
<th>Under</th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.995</td>
<td>0.967</td>
<td>0.916</td>
</tr>
<tr>
<td>FPR</td>
<td>0.005</td>
<td>0.033</td>
<td>0.084</td>
</tr>
<tr>
<td>Precision</td>
<td>0.983</td>
<td>0.970</td>
<td>0.844</td>
</tr>
<tr>
<td>F1</td>
<td>0.989</td>
<td>0.969</td>
<td>0.878</td>
</tr>
<tr>
<td>MCC</td>
<td>0.978</td>
<td>0.938</td>
<td>0.864</td>
</tr>
<tr>
<td>ROC Area</td>
<td>0.997</td>
<td>0.986</td>
<td>0.980</td>
</tr>
<tr>
<td>PRC Area</td>
<td>0.995</td>
<td>0.981</td>
<td>0.884</td>
</tr>
</tbody>
</table>

From the KNN results, we can observe that both data sampling techniques outperform CSL, as illustrated in Table 5.14. Moreover, the best performance is given for the over-sampled SB dataset, where the TPR is 99.5%, whereas in the under-sampled dataset it is 96.7%. The TPR result for CSL is only 91.4%. So, the difference between the techniques is over 5% in the favour of data sampling.

As it is shown in Figure 5.3, similarly, the highest classification quality is produced by using the over-sampling technique, where the precision rate is 98.3%, which is higher than the precision rates for under-sampling and CSL by 1.3% and 14.1%,
respectively. The F1 score is remarkably high on all over- and under-sampled datasets (98.9% and 96.9%, respectively). However, the F1 rate dropped by over 10% when this classifier combined with CSL (87.7%), as compared with the over-sampling technique. The MCC rate also provides great results for all of the techniques, as the rate is 97.8% and 93.8% for over- and under sampling techniques. Yet, MCC is decreased by about 7.4% as compared with the under-sampling technique.

Unlike other classifiers, the experimental outcomes show that HT behaves better on the under-sampled dataset compared to the other techniques, as shown in Figure 5.4. The TPR reflects the outstanding performance of HT+under-sampling, as
well as CSL+HT techniques, where both reached 99.7%. The TPR accuracy on the over-sampled dataset dropped by exactly 6% as compared with the under-sampling technique. Nevertheless, F1, and MCC, prove that HT+under-sampling performance is considered the second-best choice for this classifier, as shown in Table 5.15.

![HT performance over the handling imbalanced data techniques](image)

**Figure 5.4: HT performance over the handling imbalanced data techniques**

The precision score for the over-sampled dataset is higher than the score for the under-sampled dataset due to the excellent performance of the classifier on the normal class (the scores are 98.4% and 96.8%, respectively). However, this score is dropped by at least 17% on CSL+HT. Since the suspicious class is our primary target, we focus on the classifier performance on this class, where the best outcomes are obtained using the under-sampled technique. The F1 score for the under-sampled dataset reaches 98.2%, which is higher than the scores of the over-sampled dataset and CSL by 2.2% and 10.4%, respectively. Finally, the MCC measurement proves that applying HT+under-sampled dataset is the best choice, since the score is 96.5%, while the performance on the over-sampled dataset decreases by 4.2%. When using CSL, the MCC score hits only 87.8%, which is about 9% less than the score for the under-sampled dataset.
5.5.2 Discussion

We can summarize the findings of all of the experiments into six key points:

1. With the help of the techniques for handling imbalanced data, instance-incremental learning provided remarkable classification capabilities for the SB dataset.

2. Data sampling produced outstanding results when compared to CSL across all experiments. Nevertheless, there is a possibility of overfitting in the case of over-sampling, and critical information loss in the case of under-sampling.

3. LWL performed very well on both data sampling and CSL, and the performance metrics proved that this classifier combined with the over-sampling is the best choice for detecting SB.

4. The accuracy of KNN and KStar classifiers was relatively similar; they produced high results on data sampling (F1 above 96%), and reasonably positive outcomes when using CSL (F1 ≈ 89%). The difference between them is that KStar performed slightly better on both data sampling and CSL.

5. In general, HT provides the second-best classification performance across all the examined techniques.

6. All the examined handling imbalanced techniques returned a satisfactory classification performance but with a slight superiority of the data sampling over CSL.

5.6 Conclusion

As in the case of any fraud detection applications, the SB dataset obtained from online auctions is imbalanced. Thus, we have to tackle this issue before implementing fraud classification models. One of the conventional techniques to handle this problem is
CSL, where a penalty weight is assigned to a classifier whenever it provides a false positive or a false negative. In this study, we examined the accuracy of cost-sensitive instance-incremental classifiers. Then we compared the performance outcomes of CSL with the outcomes provided by data sampling using the same classifiers.

The experiments on CSL demonstrated that all classifiers produced acceptable performance for identifying normal and suspicious instances, and LWL provided the best classification accuracy overall (RPC = 94.6%). The KNN and KStar classifiers performed very similarly on both classes, and their classification quality is good with a minor superiority for KStar. For example, RPC score of CSL+KStar is 89.9%, which is decreased by only 1.5% for CSL+KNN. The second-best classifier performance is achieved by CSL+HT where the RPC reached 90.8%. Although CSL produced high results across all classifiers, the outcomes generated by the data sampling techniques are still superior, as presented by MCC metric. All the classifiers combined with data sampling produced over 92%, whereas the best MCC achieved with CSL is 88.6%. For example, LWL+data-sampling gave about 97% as an average of MCC, which is decreased by around 9% when it comes to CSL.
Chapter 6

Conclusion and Future Work

6.1 Summary

Over the past three decades, electronic commerce has witnessed a significant increase in the trade of goods and services. This kind of business has created an attractive environment for fraudsters who can perform various types of fraudulent activities to illegally raise money. One such activity is SB, which is prevalent across many auction sites and, unfortunately, this type of fraud is challenging to detect due to its similarity to normal bidding behaviour as well as the difficulty of systematically defining it. Furthermore, genuine bidders are not aware of this fraud until it is too late. Thus, this type of fraud has become one of the most critical fraud perpetrated on auctions. Implementing a SB detection model is very difficult and requires a deep understanding of the bidding behaviour. Moreover, the lack of SB datasets makes the implementation of SB detection and classification systems troublesome. In order to build an efficient SB detection model, we, first, created an SB dataset from data collected from real auctions of commercial sites that were most likely infected by malicious moneymakers. Thus, the initial phase in this study was scraping a large number of eBay auctions of a popular product. The raw eBay auction data collected
had a lot of distortions that needed to be cleaned. After preprocessing the raw auction data, we produced a high-quality SB dataset based on the most reliable SB strategies. One of this work’s achieved goals is to share the preprocessed auction dataset and the SB training (unlabelled) dataset, thereby allowing researchers to experiment with using different ML techniques.

Classifying each instance in the training dataset is another critical and sensitive step in ML, as the quality of a model’s predictions depends on this step. This is another challenging task faced in this research because the obtained high-quality SB training dataset is not labelled. Therefore, one of the objectives of this work is to address the above-mentioned problem by providing a labelling approach for the SB pattern dataset to be used in SB detecting models based on ML. The hierarchical clustering techniques exhibit excellent results for defining and grouping similar data instances. We employed CURE algorithm to distinguish between normal and suspicious bidders behavior. The procedure of the systematic labelling technique is summarized as follows:

1. Partition the high-quality SB training dataset into five subsets based on the bidding durations.

2. Determine the optimal number of clusters of each subset using the most reliable technique for this purpose.

3. Pass the optimal number of clusters along with targeted subset and other parameters to CURE.

4. Finally, examine each cluster by comparing its mean to the decision line.

The experiment demonstrated that the employed labelling system returns excellent outcomes.

Class imbalance is a complicated issue in ML. Many researchers have suggested alternative approaches, whether from the data level or the algorithm level, to address
this problem. Some researchers approve the former approaches, while others prefer
the latter one, and others approve both. The generated high-quality labelled SB
dataset is imbalanced, just like any real-world fraud data, where the number of normal
instances is much higher than the number of suspicious instances. To overcome the
problem of imbalanced SB datasets, in this study, we investigated over- and under-
sampling techniques, which are data level approaches, through several instance-based
classification algorithms. Thousands of auctions are established on eBay daily, and
auction bid transactions may be sent continuously to the optimal fraud classifier in
order to detect potential SB activities. Thus, instance-based classification is suitable
for our particular fraud detection problem, since instance-based learning is capable of
learning from data streaming. Based on the experimental outcomes, the incremental
classification quality is high for both over- and under-sampled SB datasets. Moreover,
the results show that over-sampling slightly outperforms under-sampling for both
normal and suspicious classes across all experiments.

The class of imbalanced data also investigated using algorithmic level techniques
to gain further insights about the SB dataset. So, this study also covers and ex-
amines the feasibility of applying the CSL technique to the imbalanced dataset, as
CSL has been suggested in many studies. The CSL results are compared with the
data sampling outcomes that were previously conducted in our research. The same
instance-incremental classifiers are set as the base line classifiers for the CSL ap-
proach. According to the experimental results, incremental classification returns a
high performance with both data and algorithmic level techniques. CSL shows highly
predictive performance results across all experiments. However, both data sampling
techniques slightly outperform CSL for the suspicious class across all classifiers.
6.2 Future Work

In this section, we would like to provide an overview of the future contributions in ML generally, and SB fraud detection in particular.

- Preprocessing raw datasets is a challenging and important step that consumes a lot of time and effort, but it must be addressed. Therefore, it is important to develop software that helps data scientists to track what has been done to the data, which is one of our planned future projects. Preprocessing raw data can contain different types of tasks and queries. So, by tracking the modifications that have been done in the dataset, we can save time and efforts for preprocessing future raw data. Also, we will automate the data preprocessing to address the issue of hard-loaded work.

- The developed SB dataset consists of eight different SB features. So, identifying and defining more bidder and seller behavioural patterns is another work we plan to do in the future. The feasibility of the SB dataset will thereby be enhanced, making it more generalized for use in various SB detection models.

- Our future work will concentrate on deriving a complete online auction fraud detection system that instantly discovers shill bidders at the last stage of an auction i.e. just before processing the payment of the products. We suggest an hybrid approach that functions as follows in real-word scenarios:
  - We employ one-class SVM to detect outliers. The classifier will be trained on the normal SB instances only. Thus, any unseen instances classified as outliers will represent fraud.
  - We use CSL+LWL to classify unseen SB instances since this combination is proven in our experiments to be capable of producing high-quality classifications.
– Then, we compare the classification results of the two learned models. When there is a conflict in the prediction of an instance, then we send this instance for further investigation.

• In this work, we focused mainly on utilizing instance-incremental classification methods and examine the feasibility of data sampling and CSL, which yielded excellent results, in terms of non-linearly classifying the imbalanced SB dataset. In the future, we plan to test the performance of other powerful classification techniques, such as ensemble learning, and incremental-decremental learning.

• One of the future research questions that we need to answer is as follows: when we have an imbalanced dataset (with a ratio of, as an example, 1:10) can one-class SVM handle it, if 70% of normal instances are trained and the other 30% are used for testing (note: 10% of tested instances are suspicious) without utilizing handling imbalanced techniques?
REFERENCES


[CZZ13] Peng Cao, Dazhe Zhao, and Osmar R. Zaïane. An optimized cost-sensitive SVM for imbalanced data learning. In *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD*


Alberto Fernández, Salvador García, Francisco Herrera, and Nitesh V. Chawla. Smote for learning from imbalanced data: Progress and chal-


Appendix A

SB Metrics in MSSQL & Python

Bidder Tendency

```sql
-- Step 1: Find the bidder tendency per auction
CREATE VIEW BIDDER_TENDENCY_Per_Auc
AS
(
    SELECT
        Auction_id,
        BidderID,
        COUNT(BidderID) AS NO_Tendency_Per_Auc1
    FROM Const_SB_Attributes1
    GROUP BY BidderID, Auction_id
)

-- Step 2: Find the bidder tendency for all auctions
CREATE VIEW BIDDER_TENDENCY_For_All
AS
(
    SELECT BidderID, COUNT(BidderID) AS BIDDER_TENDENCY_For_All1
    FROM Const_SB_Attributes1
    GROUP BY BidderID
)

-- Step 3: Find the bidder tendency shell pattern.
CREATE VIEW BIDDER_TENDENCY
AS
(
    SELECT 
        BIDDER_TENDENCY_Per_Auc.Auction_id AS A_ID,
        BIDDER_TENDENCY_Per_Auc.BidderID AS B_ID,
        BIDDER_TENDENCY_Per_Auc.NO_Tendency_Per_Auc1,
        BIDDER_TENDENCY_For_All.BidderID,
        BIDDER_TENDENCY_For_All.BIDDER_TENDENCY_For_All1,
        IIF(BIDDER_TENDENCY_For_All.BIDDER_TENDENCY_For_All1 > 1,
            BIDDER_TENDENCY_Per_Auc.NO_Tendency_Per_Auc1 * 1.0 / 
            BIDDER_TENDENCY_For_All.BIDDER_TENDENCY_For_All1, 0) AS BIDDER_TENDENCY1
    FROM BIDDER_TENDENCY_Per_Auc
    INNER JOIN BIDDER_TENDENCY_For_All
    ON BIDDER_TENDENCY_Per_Auc.BidderID = BIDDER_TENDENCY_For_All.BidderID
)

-- Step 4: update Bidder_Tendency attribute in Auction_SB_Patterns table.
UPDATE Const_auc_SBP
SET Bidder_Tendency = BIDDER_TENDENCY.BIDDER_TENDENCY1
FROM BIDDER_TENDENCY
WHERE Const_auc_SBP.Bidder_ID = BIDDER_TENDENCY.B_ID and Const_auc_SBP.Auction_ID = BIDDER_TENDENCY.A_ID;
SELECT * FROM [dbo].[Const_auc_SBP];
```
Early Bidding

-- Step 1: Findig Early_Bidding pattern in each auction.
CREATE VIEW EARLY_BIDDING
AS

(SELECT record_id,
Auction_id,
BidderID,
1 - ((SBDsec - BTsec) * 1.0 / ADsec) AS Early_Bidding1
FROM Const_SB_Attributes1
)

SELECT Auction_id, BidderID, min(Early_Bidding1) FROM EARLY_BIDDING
where Auction_id = 600
group by Auction_id, BidderID

select Auction_id, Bidder_ID, Early_Bidding
from Const_auc_original_SBP
where Auction_ID = 600

--Step 2: update Early_Bidding attribute in Auction_SB_Patterns table.
update Const_auc_SBP
set Early_Bidding = EARLY_BIDDING1
from (SELECT record_id, Auction_id, BidderID,
1 - ((SBDsec - BTsec) * 1.0 / ADsec) AS Early_Bidding1
FROM Const_SB_Attributes1
) EARLY_BIDDING1
where Const_auc_SBP.Bidder_ID = EARLY_BIDDING1.BidderID
and Const_auc_SBP.Auction_ID = EARLY_BIDDING1.Auction_id;

SELECT * From Const_auc_SBP;
Bidding Ratio

--Step 1: Finding the total bids submitted by each bidder in each auction
CREATE VIEW total_Bdr_bides_per_A AS

( SELECT Auction_id as A_ID,  
  BidderID as B_IB,  
  count(BidderID) as total_Bdr_bides_perA1  
  FROM Const_SB_Attributes1  
  GROUP BY Auction_id,BidderID)

--Step 2: Find the total number of bids in each auction
CREATE VIEW total_bides_A AS

( select A_ID as A_ID1, sum(total_Bdr_bides_perA1) as total_bides_A1  
  FROM total_Bdr_bides_perA  
  GROUP BY A_ID)

--Step 3: Find Bidding Ratio
CREATE VIEW BR AS

( select A_ID as auc_ID, B_IB as bdr_ID,  
  total_Bdr_bides_perA1*1.0 / total_bides_A1 as BR1  
  FROM total_Bdr_bides_per_A  
  INNER JOIN total_bides_A ON A_ID = A_ID1)

--Step 4: Updating Bidding Ratio on Auction_SB_Patterns
UPDATE Const_auc_SBP
  SET Bidding_Ratio = BR.BR1
  FROM BR  
  WHERE Auction_ID = BR.auc_ID and Bidder_ID = BR.bdr_ID;

SELECT * FROM Const_auc_SBP
Last Bidding

--Step 1: Finding the last bid of each bidder in each auction
Create view last_B_bid_in_A
as
(
    select Auction_id as auc_id,
    BidderID as bdr_id,
    AEDsec as End_Time,
    ADsec as A_duration,
    min(BTsec) as last_B_bid_in_A1
    from Const_SB_Attributes1
    group by Auction_id, BidderID, AEDsec, ADsec
)

--select Auction_id, BidderID, AEDsec, ADsec, BTsec, BidAmount, Winningbid
from Const_SB_Attributes1
where Auction_id = 5
order by BTsec

--Step 2: compute the last bidding pattern
Create view last_bid
as
(
    select auc_id,
    bdr_id,
    IIF(((last_B_bid_in_A1 - End_Time)^1.0 / A_duration) > 0.0001,
        ((last_B_bid_in_A1 - End_Time)^1.0 / A_duration), 0) as last_bid1
    from last_B_bid_in_A
)

select * from last_bid

--Step 3: Updating Last_Bidding attribute on Auction_SB_Patterns
update Const_auc_SB
set Last_Bidding = last_bid.last_bid1
from last_bid
where Const_auc_SB.Auction_ID = last_bid.auc_id and Const_auc_SB.Bidder_ID = last_bid.bdr_id;

select * from Const_auc_SB
Auction Starting Price

-- Step 1: Compute Starting_Price_Average pattern
CREATE VIEW Avg_Start_Price AS

    (select Auction_ID as A_ID,
       IIF( (SELECT (AVG(StartingPrice) * 0.5) FROM Const_SB_Attributes1) > (StartingPrice),
            (1 - ((StartingPrice * 1.0) / (SELECT AVG(StartingPrice) FROM Const_SB_Attributes1))), 0)
       as Avg_Start_Price1
       from Const_SB_Attributes1)

    select Avg_Start_Price1 from Avg_Start_Price

-- Step 2: Update the Starting_Price_Average in the Auction_SB_Patterns table
update Const_auc_SBP
    set Starting_Price_Average = Avg_Start_Price1
    from Avg_Start_Price
    where Auction_ID = A_ID;

Select * from Const_auc_SBP

Successive Outbidding

sucBidding = int (1)

# for bid in auctionlist:
for i in range(len(auctionlist)):
    if i+1 < len(auctionlist):
        if auctionlist[i] == auctionlist[i-1] and auctionlist[i] == auctionlist[i-2] and auctionlist[i] == auctionlist[i-3]:
            sob = 1
            print (auctionlist[i],',',sob)
        elif auctionlist[i] == auctionlist[i-1] and auctionlist[i] == auctionlist[i-2]:
            sob = 0.5
            print (auctionlist[i],',',sob)
        else:
            sob = 0
            print (auctionlist[i],',',sob)
        break
    if auctionlist[i] == auctionlist[i+1]:
        sucBidding += 1
    else:
        if sucBidding >= 3:
            for j = 0 to i
                sob = 1
                print (auctionlist[i],',',sob)
            elif sucBidding > 2:
                sob = 0.5
                print (auctionlist[i],',',sob)
            else:
                sob = 0
                print (auctionlist[i],',',sob)
        sucBidding = 1
Winning Ratio

-- Step 1: Removing duplicates number of bidder won auctions
Create view Remove_duplicate_Winning_B as
{
    select Auction_id as Auc_id, Bidder_id as bdr_id, Winning_bid, Bid_amount,
    count (Bidder_id) as Winning_B,
    IIF(count (Bidder_id) > 1,1,1) as Remove_duplicate1
    from cln_Auc_Attributes
    where Winning_bid = Bid_amount
    group by Auction_id, Bidder_id, Winning_bid, Bid_amount
}

-- Step 2: computing the number of bidder won auctions
Create view No_B_won_A as
{
    select bdr_id, Remove_duplicate1, count (Remove_duplicate_Winning_B, Remove_duplicate1)
    as No_B_won_A1
    from Remove_duplicate_Winning_B
    group by bdr_id, Remove_duplicate1
}

-- Step 3: computing the number of bidder joined auctions
Create view No_B Joined_A as
{
    select Bidder_id as b_id, count(Bidder_id) as No_B_joined_A1
    from cln_Auc_Attributes
    group by Bidder_id
}

-- step 4: compute Winning Ratio pattern
Create view WR as
{
    select b_id as Bdr_ID, 1 - (No_B_won_A1 * 1.0 / No_B_joined_A1) as WR1
    from No_B_joined_A
    inner join No_B_won_A on No_B_joined_A.b_id = No_B_won_A.bdr_id
}

-- Step 5: Update Winning_Ratio pattern in the Auction_SB_Patterns
Update 1: To set 1 for non winners
update Auction_SB_Patterns
    set Winning_Ratio = 1
Update 2: To set weight for winners
update Auction_SB_Patterns
    set Winning_Ratio = WR1.WR1
from WR
where Auction_SB_Patterns.Bidder_ID = WR.Bdr_ID;
Auction Bids

-- Step 1: Finding the number of bids in each auction
CREATE VIEW number_of_bids AS

( SELECT
    distinct Auction_id as A_ID,
    count(BidderID) as NO_bids
FROM Const_SB_Attributes1
GROUP BY A_ID
)

select * from number_of_bids

-- Step 2: Finding the avg number of bids in all auctions
CREATE VIEW avg_number_of_bids AS

( SELECT
    A_ID as A_ID1,
    (select sum(NO_bids) / (count(A_ID)) as avg_bids from number_of_bids) as avg_bids1
FROM number_of_bids
)

-- Step 3: Comparing and computing the bids in each auction with avg bids all auction
CREATE VIEW AUCTION_BIDS AS

( select A_ID as A_id1,
    NO_bids as NO_bids1,
    avg_bids1 as avg_bids2,
    IIF(avg_bids1 < NO_bids,
        1 - (avg_bids1 * 1.0 / NO_bids), 0) as AUCTION_BIDS1
FROM number_of_bids
inner join avg_number_of_bids on A_ID = A_ID1
)

-- Step 4: set the results in SB table
update Const_auc_SB
    set Auction_Bids = AUCTION_BIDS.AUCTION_BIDS1
from AUCTION_BIDS
    where Const_auc_SB.Auction_ID = AUCTION_BIDS.A_id1;
## Appendix B

### Tools and Libraries

Table B.1: List of computers, tools, and libraries

<table>
<thead>
<tr>
<th>devises, tools, and services</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octoparse service</td>
<td>To professionally scrap auctions data from ebay for three months.</td>
</tr>
<tr>
<td>Windows 10 on Dell machine that equipped with Intel (R) Xeon (R) CPU E5-1620 V4 @ 3.50HGs 3.50HGs</td>
<td>To process the large extracted data since it has multi-CPUs. Note that the imac equipped with 2.8 GHz Intel Core i7, and memory 12 GB 1067 MHz DDR3 could not handle the received data properly.</td>
</tr>
<tr>
<td>Microsoft server management studio (SSMS 2017)</td>
<td>To preprocess the raw auctions data. To calculate the statistics. To compute the SB patterns. Note that, Microsoft excel was not capable to handle the raw dataset.</td>
</tr>
<tr>
<td>Anaconda Navigator (1.7.0)</td>
<td>It is a graphical user interface tool that enables you to launch applications (such as python in our case) and run conda packages, and environments without the need to use line commands.</td>
</tr>
<tr>
<td>iMac (10.13.6 (17G6030) with 2.8 GHz Intel Core i7 CPU and 12 GB 1067 MHz DDR3 of memory)</td>
<td>To set up Anaconda Navigator environment for Python to implement SB labelling strategy</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Python</td>
<td>Used to implement the clustering, and to determine the optimal number of clusters techniques.</td>
</tr>
<tr>
<td>WEKA (3-9-2-oracle-jvm)</td>
<td>It is an environment desktop that collects machine learning algorithms for data mining assignments. Also, It includes tools for data preprocessing, classification, clustering, regression, association rules mining, and visualization.</td>
</tr>
<tr>
<td>import silhouette_samples, silhouette_score</td>
<td>To determine the optimal number of clusters.</td>
</tr>
<tr>
<td>from sklearn.cluster import KMeans</td>
<td>K-mean clustering is required by silhouette to produce the silhouette’s score. Also, to be used for the clusters visualization.</td>
</tr>
<tr>
<td>from pyclustering.cluster.cure import cure</td>
<td>To perform CURE clustering technique.</td>
</tr>
<tr>
<td>import seaborn</td>
<td>It is Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.</td>
</tr>
<tr>
<td>Class SMOTE</td>
<td>To over-sample the minority class.</td>
</tr>
<tr>
<td>Class SpreadSubsample</td>
<td>To under-sample the majority class.</td>
</tr>
<tr>
<td>Class CostSensitiveClassifier</td>
<td>A metaclassifier that makes its base classifier cost-sensitive.</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Randomize</td>
<td>To randomly shuffle the order of instances.</td>
</tr>
<tr>
<td>NumericToNominal</td>
<td>A filter for turning numeric attributes into nominal ones.</td>
</tr>
<tr>
<td>LWL Classifier</td>
<td>To perform Classification based on LWL</td>
</tr>
<tr>
<td>K* Classifier</td>
<td>To perform Classification based on K*</td>
</tr>
<tr>
<td>IBK Classifier</td>
<td>To perform Classification based on KNN</td>
</tr>
<tr>
<td>Hoeffding Tree</td>
<td>To perform Classification based on Hoeffding Tree</td>
</tr>
</tbody>
</table>