Dissertation

Classification of Biomedical Data with Class Imbalance

クラスインバランスがある生物医学データの分類

Graduate School of

Natural Science & Technology

Kanazawa University

Division of Electrical Engineering and Computer Science

Student ID No. 1824042010

Name : Kunti Robiatul Mahmudah

Chief Advisor : Professor Kenji Satou

Date of Submission : June, 2021

Abstract

Classification of Biomedical Data with Class Imbalance

In the machine learning task, pre-processing is a crucial step before feeding the data into the model. It improves the quality of the data and performances of the model. One of the pre-processing task is dealing with class imbalance. This problem can decrease the performance or give the model bias results. A frequently used method to resolve this problem is through oversampling. In this study, we employ some oversampling methods and features extraction methods as the pre-processing task. In the case of Chronic obstructive pulmonary disease (COPD), we proposed methods that utilize gene expression data from microarrays to predict the presence or absence of COPD. The proposed method assists in determining better treatments to lower the fatality rates. In this study, microarray data of the small airway epithelium cells obtained from 135 samples of 23 smokers with COPD (9 GOLD stage I, 12 GOLD stage II, and 2 GOLD stage III), 59 healthy smokers, and 53 healthy non-smokers were selected from GEO dataset. Machine learning and regression algorithms performed in this study included Random Forest, Support Vector Machine, Naïve Bayes, Gradient Boosting Machines, Elastic Net Regression, and Multiclass Logistic Regression. After removing imbalanced data effect using SMOTE, classification algorithms of elastic net regression and multiclass logistic regression achieved high AUC score and the other metrics which outperformed the other classifiers. In the case of binary features data, by converting binary features into numerical ones using feature extraction methods prior to oversampling can fully display their potential in improving the classifier's performances. Although the comprehensive experiment in this study was done using benchmark datasets and real medical datasets, it was observed that a converted dataset consists of numerical features is better for oversampling methods. In addition, it is confirmed that features extraction and oversampling are synergistically contributing to the improvement of classi-fication performance.

keywords: binary features, class imbalance, imbalanced data, feature extraction, oversampling, dimensionality reduction, COPD, biomedical data, classification

Acknowledgment

First and foremost, praises and thanks to Allah SWT, for His showers of blessings throughout my research work to complete my Ph.D degree successfully.

I would like to express my deep and sincere gratitude to my research supervisor, Professor Kenji Satou and other professor in Bioinformatics laboratory, Graduate School of Natural Science and Technology, Kanazawa University, for giving me the opportunity to do research and continually supporting and providing invaluable guidance throughout this research.

I am very much thankful to my husband, Norma Sidik Risdianto, and my son, Haidar Nurshiddiq Ibrahim for their love, understanding, prayers, encouragements, and continuing support to complete this research work.

I am extremely grateful to my parents, and my siblings for their love, prayers, support, and sacrifices for preparing me for my future.

Special thanks goes to my lab-mates for the discussions and inspirations to overcome any obstacles that I faced.

In addition, a thank you for Japanese government (MEXT) and Rotary Yoneyama Foundation for providing me with scholarship so I can finish my Ph.D without worrying about my financial condition.

Contents

Abstract ii

Acknowledgment iv

Contents v

1 Introduction 1

1.1 Class Imbalance . . . 1
1.2 Binary Features Data . . . 2
1.3 Objectives . . . 2
1.4 Contributions . . . 3
1.5 Thesis organization . . . 3

2 Machine Learning 5

2.1 Definition . . . 5
2.2 Pre-processing data . . . 6
2.2.1 Data Cleaning . . . 6
2.2.2 Data Transformation . . . 7
2.2.3 Dimensionality Reduction . . . 7
2.2.4 Handling Missing Value . . . 8
2.2.5 Dealing with Class Imbalance . . . 9
2.3 Machine Learning Model . . . 9
2.3.1 Supervised Learning . . . 9
2.3.2 Unsupervised Learning . . . 13
2.3.3 Semi-supervised Learning . . . 14
2.3.4 Reinforcement Learning . . . 15
2.4 Model Evaluation . . . 15

2.5 Performance Metrics . . . 16
2.5.1 Threshold Metrics . . . 16
2.5.2 Ranking Methods and Metrics . . . 19
2.5.3 Probabilistic Metrics . . . 20
2.6 Frameworks . . . 21

3 Dealing with Class Imbalance and Features Extraction Methods 22
3.1 Algorithm Level . . . 22
3.2 Data Level . . . 23
3.2.1 Down-sampling . . . 23
3.2.2 Up-sampling . . . 23
3.2.3 Under-sampling . . . 23
3.2.4 Over-sampling . . . 25
3.3 Features Extraction . . . 30
3.3.1 Principal Component Analysis (PCA) . . . 30
3.3.2 Independent Component Analysis (ICA) . . . 31
3.3.3 T-Distributed Stochastic Neighbor Embedding (tSNE) . . . 31
3.3.4 Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) . . . 31

4 Machine Learning Algorithms for Predicting Chronic Obstructive Pulmonary Disease (COPD) from Gene Expression Data with Class Imbalance 33
4.1 Introduction . . . 33
4.2 Material and Methods . . . 35
4.3 Machine Learning Methods . . . 36
4.4 Evaluation Metrics of Predictive Models . . . 36
4.5 Evaluation of Resampling Methods . . . 38
4.6 Experiment and Result . . . 39
4.7 Comparison of Machine Learning Algorithm and Regression Analysis . . . 40
4.8 Comparison of Resampling Methods . . . 41
4.9 Conclusion . . . 42

5 Classification of Imbalanced Data Represented as Binary Features 43
5.1 Introduction . . . 43
5.2 Material and Methods . . . 45

5.3 Oversampling and Features Extraction Method 46
5.4 Classification Methods 47
5.5 Evaluation Metrics 48
5.6 Result and Discussion 48
5.7 Conclusion 54

Conclusion **55**

List of Figures

1.1 An example of Imbalance Data 1

2.1 K-NN classifier with different value of k. 10
2.2 An example of decision tree. 12
2.3 An example of random forest. 13
2.4 Optimal hyperplane. 14
2.5 5-fold cross validation. 15
2.6 (a) comparison of ROC; (b) comparison of precision-recall curve (PRC) . . 20
2.7 Machine Learning Framework by Guo. 21

4.1 Flowchart of this study. 34
4.2 Mean difference (MD) plot displays log2 fold change versus average log2 expression values for all the 54,675 probes. Highlighted genes are significantly differentially expressed in COPD compared to healthy non-smoker (red = upregulated, blue = downregulated) 39

5.1 The flowchart of the proposed approach. 44

List of Tables

2.1 Confusion matrix of binary class case. 16

4.1 An illustrative example of the confusion matrix for a 3-class classification test. 37
4.2 Accuracy and AUC for different models. 40
4.3 Average sensitivity and specificity for different models. 40
4.4 Multiclass LR with different resampling method. 42

5.1 Multiclass LR with different resampling method. 46
5.2 Performance metrics of the best combination compared to base model. . . 49
5.3 Summary of accuracy improvements about SPECT dataset. 51
5.4 Summary of accuracy improvements about analcatdata_fraud dataset. . . 51
5.5 Summary of accuracy improvements about MRSA pathogenicity dataset. . 52
5.6 Summary of accuracy improvements about C. difficile pathogenicity dataset. 52
5.7 Summary of F1 score improvements about SPECT dataset. 52
5.8 Summary of F1 score improvements about analcatdata_fraud dataset. . . . 53
5.9 Summary of F1 score improvements about MRSA pathogenicity dataset. . 53
5.10 Summary of F1 score improvements about MRSA drug resistance dataset. 53
5.11 Summary of F1 score improvements about C. difficile pathogenicity dataset. 54

Chapter 1

Introduction

1.1 Class Imbalance

Imbalance datasets exist in many real-world data. Class imbalance happen when the number of a class is far less than that in the other one as can be seen in Figure 1.1 of PCA plot of *dis* dataset retrieved from [1]. The target class is usually the minority class or the class which has samples far less than the other one and a sample in this class is called as positive sample while a sample from the other one is called as negative sample. This problem can lead classifier to bias toward the majority class because it will most likely to predict a positive sample as negative sample. Therefore, a method to deal with class imbalance should be done first before providing the data as an input to the classifiers to improve detection of minority sample or the performance metrics.

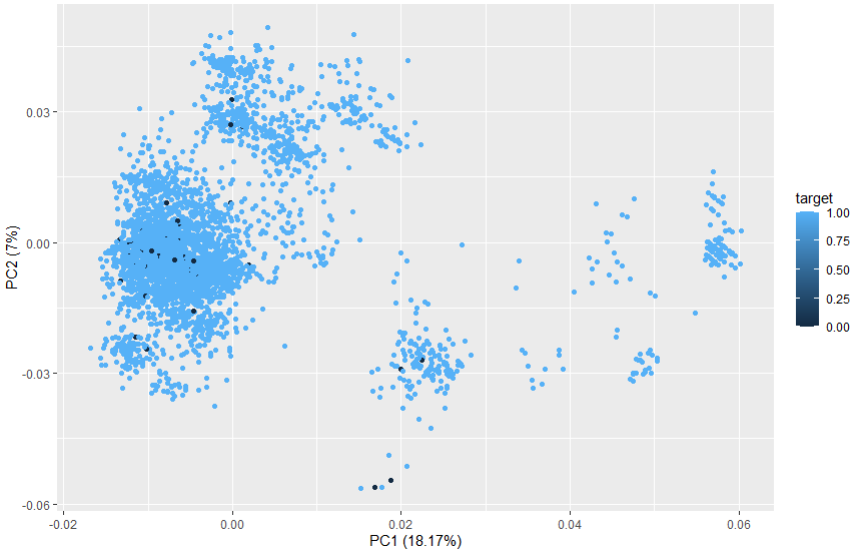

Figure 1.1: An example of Imbalance Data

Research on the class imbalance issue is cruial in the topic of in machine learning. These two factors outline why class imbalance problem should be handling prior to classification. The problem of class imbalance is pervasive across a wide range of important fields in the data mining community and when confronted with the problem of class imbalance, most common classification learning methods were shown to be insufficiently effective. Many available methods exist to deal with class imbalance at either in algorithm level or data level. This thesis shows methods of the data level on the classification of imbalanced data such as SMOTE, ADASYN, adaptive neighbor SMOTE, borderline-SMOTE, safe-level SMOTE, relocating-safe-level SMOTE, and DBSMOTE.

1.2 Binary Features Data

In the field of machine learning, it is important to understand the characteristics of input data and select the methods that are most suitable for achieving high performance in the machine learning task (regression, classification, clustering etc.). As mentioned in the previous section of methods for dealing with class imbalance, many of the methods are specifically developed to overcome the problem in numerical data. In contrast, some types of data are represented as binary value that can take only one of two values (0/1, T/F, M/F, etc.). Such a binary feature is located on the border of numerical and categorical values. It can be treated as numerical value; however, the domain of value is quite poor and no difference from categorical value. Therefore, for a dataset consists of binary features, direct application of oversampling methods is not a good idea since such methods rely on numerically represented values for synthesizing new minority samples.

1.3 Objectives

Typically, classification is conducted on a dataset that consists of numerical features and target classes. For instance, a grayscale image is usually represented as a matrix of integers varying 0-255 and it enables to apply various classification algorithms to image classification tasks. However, datasets represented as binary features are not so special and their amount is not neg-ligible. On the other hand, oversampling algorithms such as SMOTE and its variation are often used if the dataset for classification is imbalanced. However, since SMOTE and its variant synthesize new minority samples based on the original samples, the diversity of the samples synthesized from binary features is highly limited due to the poor representation of original features. To solve this problem, a

preprocessing approach is studied. By converting binary features into numerical ones using feature extraction methods, succeeding oversampling methods can fully display their potential. In this research, we study the method to handling class imbalance in binary features dataset by combining feature extraction and oversampling methods as the preprocessing task.

1.4 Contributions

Machine Learning is one of the frequently used systems to do classification and clustering in biomedical data. Many studies of classifications such as drug discovery, disease detection, and genes selection have been explored by many researches. This study contributes in:

- Proposed a method to apply machine learning in classification of biomedical data with class imbalance.
- Proposed a method in classification of binary attributes data with class imbalance
- improved evaluation performances in microarray data classification with class imbalance.

1.5 Thesis organization

This thesis consist of 6 chapters:

- **Chapter 1** will briefly introduce the background of class imbalance and dimensional reduction task. The objectives, contribution, and the organization of this research will be presented in this chapter.
- **Chapter 2** will present some background knowledge about machine learning that we used in this research.
- **Chapter 3** will introduce about class imbalance problem. Then it will show some previous works of techniques to resolve this problem. It will also some basic explanation of features extractions, one of dimensional reduction methods, that were utilized in this research.

- **Chapter 4** will explain about the published article entitled "Machine Learning Algorithms for Predicting Chronic Obstructive Pulmonary Disease (COPD) from Gene Expression Data with Class Imbalance".
- **Chapter 5** will explain about the published article entitled "Classification of Imbalanced Data Represented as Binary Features".
- **Chapter 6** will present the summary of the accomplished works in our research. Other than that, it will also present some ideas for future works to improve performances of our proposed methods.

Chapter 2

Machine Learning

In this chapter, I will present some background knowledge about machine learning. Then I will describe about machine learning techniques we used in this research. And finally, I will show some examples in implementing machine learning models in microarray dataset.

2.1 Definition

People have utilized data for decades and have adjusted systems to the data patterns changes. However, when the volume and variability of data increases greatly and grows beyond our ability to make sense of it and write those rules manually, we will increasingly rely to automated systems that can learn from either the data or the changes in the data. Machine learning is part of artificial intelligent (AI) which study computer algorithms and gives it the ability to automatically learn and improve through experience without being programmed periodically. Machine learning covers a whole process from data collection to prediction to make use and get insight of the data. It includes a task consisting of applying statistical and mathematical approaches to have machines learnt from data.It is a tool and technology that we can utilize to answer questions with data. The data type can be tabular data, text, images, videos, and anything that we can extract some knowledge of. This capabilities of machine learning can be applied to a diverse number of fields. We can see many examples of machine learning like on recommendation system of Netflix, Amazon, and other e-commerces that suggest things for users or Ads that pops up on Google and looks at user's preferences. Apple's siri also an example of technology that convert voice's acoustic pattern into probability distribution by utilizing machine learning and deep learning.

In machine learning, the data is used to inform the building and fine tuning of a

predictive model which can subsequently be used to make predictions on previously unlabeled data and answer our questions. As more data is obtained, the built model can be modified over time and new predictive models deployed and getting improved since the model can adjust automatically to the data.

In the life science domain data, from drug discovery, genes and disease identification, medical imaging diagnosis, outbreak prediction, and many more, machine learning has provided computer systems completely new capabilities. In machine learning, the purpose of training is to develop an accurate model that almost all of the time answers our questions accurately. To train our model, we need to gather data to train it. Therefore, we need to understand the steps in machine learning model, so we can develop a model to get insight from our data.

2.2 Pre-processing data

Data pre-processing is the procedure for preparing raw data for use in a machine learning model. It's the first and essential stage in building a machine learning model. A real-world data sometimes contains noise, missing values, data imbalance, or is not in the correct format, making it unsuitable for use in machine learning models. Data pre-processing is an essential step to clean up data and make it suitable for machine learning models resulting in the improvement of the accuracy and efficiency of the model. This includes data preparation and data wrangling. In general, pre-processing data can be as the process of data cleaning, data integration, data transformation, dimensionality reduction, handling missing value, dealing with class imbalance, outliers removal, and data validation. The most common data structure in wrangling or preparing a data is data frame which based on rows and columns as in a spreadsheet.

2.2.1 Data Cleaning

Data cleaning is a process to detect and repair errors, duplication, corrupted data, or incomplete data in the dataset in order to create a reliable dataset and improve the quality of the training data. This process is important since when collecting and combining data from multiple source, there is possibility that we create duplicate, mislabelled, incomplete, or improperly formatted data.

2.2.2 Data Transformation

When collecting data, different type of variables with a significant gap or range of that variables may occur. The collected data cannot be used directly for performing the analysis process. Insisting on using the original scale of data in the analysis can lead to error or bias and the algorithm may put more weight on the larger range variables. For that reason, transform data which can be normalizing or standardizing is required to obtain maximum data quality that is ready for analysis. Generally saying, data transformation is a process of transforming or converting data from one format to another format that required to a destination system.

2.2.3 Dimensionality Reduction

Dimensionality refers to the number of features or variables of dataset. Dimensionality reduction is the technique of reducing the number of random variables considered by generating a set of statistically important variables or by projecting the dataset into a lower-dimensional subspace that captures the "gist" of the data. In a real-world data, it is common to have thousands of features in a dataset. Not all the features have equal importance or significance, in fact, sometimes only several features are informative and needed to be included in analysis. In a very high dimension dataset, it is beneficial to reduce the dimension of dataset to reduce computation cost, avoid over-fitting, visualize the dataset more easily, improve the interpretation of the parameters by removing multi-collinearity, and remove irrelevant features from the data. Dimensionality reduction technique is divided into feature selection and feature extraction.

1. **Feature Extraction**

   Features ectraction is the method to reduce the dimension of the data by projecting the dataset into a lower-dimensional subspace that captures the "gist" of the data. Feature extraction can be both linear and non linear. The most common linear methods of features extraction used in machine learning are Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA). As for non-linear features extraction methods, t-distributed stochastic neighbor embedding (t-SNE) and uniform manifold approximation and projection (UMAP) are commonly used in the machine learning algorithms. The main disadvantage of dimensionality reduction using feature extraction is that the generated features can not be interpreted by us directly as the original features.

2. **Feature Selection**

   Features selection is a method to remove less or non-significant features for the data, so that the remain features are the features that can contribute most in the machine learning. Feature selection type can be divided as:

   - embedded : feature importance using random forest, Lasso regularization, etc.
   - wrapper: forward, backward, stepwise selection, etc.
   - filter: chi-square test, fisher's score, correlation coefficient, variance threshold, mean absolute difference (MAD), etc.
   - hybrid: combination of the different approaches mentioned above to get the best possible feature subset.

   The benefit of feature selection compared to feature reduction is that we can maintain our ability to interpret our model and output.

2.2.4 Handling Missing Value

Most of machine learning algorithms can not deal with missing values both in the target and features. That is why, it is crucial to handle it first before go with machine learning model. Delete the sample which has missing value in some features is the simplest way, but sometimes the sample is an important sample that can not be delete by the researchers with certain reason. Moreover, deleting that sample can lead to bias in the task. There are some common ways to handling missing value, three common ways are:

1. **Deletion method**

   This include delete a sample that has certain portion of missing value in the features/variables. For example, if a data has at least 80 percent information of all the variables, then it should be keep. Otherwise, the data will be deleted.

2. **Data imputation with average or median**

   A common ways in data imputation are average imputation where the missing value in a cell is inputted with the average value of that variable and median imputation where the missing value is inputted with the median of that variable.

3. **Data imputation with a model**

   Another way to handle missing value is using regression or k-nearest neighbour to predict the null value using the dataset.

2.2.5 Dealing with Class Imbalance

Class imbalance data occurs when the number of a class of a data is significantly smaller than the number of the other classes. In real-world dataset, the occurrence of class imbalance is frequent for example in medical such as disease-related data. The target class is usually the minority class and is the main focus for the analysis. Most of machine learning algorithms can not deal with class imbalance directly. Therefore, a method to deal with class imbalance should be done first before feeding the data as an input to the classifies to improve detection of minority sample or the performance metrics. There exist many methods to deal with class imbalance such as:

- collecting more data
- changing performance metric
- using penalize algorithms
- using resampling technique

The detail explanation of the resampling methods to deal with this problem is presented on Chapter 3 as it becomes our main interest in this work.

2.3 Machine Learning Model

Machine learning consist of four types of techniques:

2.3.1 Supervised Learning

In supervised learning, the data that is used to feed a model is a labelled data. There are two types of supervised learning techniques: regression and classification. In the regression, the built model aims at forecasting the output value based on historical data, for example we want to predict the price of a house according to variables of area size, location, and the distance from the house to a certain place. In regression, if $X = \{x_1, x_2, ..., x_i\}$ is a set of independent variables and Y is target or dependent variable, then Y should be continuous value. Algorithms such as linear regression, random forest, decision tree, k-nearest neighbors can be used for regression.

In the classification, the built model aims at separates the data into different classes, for example in deciding whether a received email is spam or not. Target of dependent variable Y in classification is categorical or discrete value. Algorithms such as logistic

regression, random forest, decision tree, support vector machine, and k-nearest neighbors can be used for classification although some of them also can be used for regression. These are some supervised learning for the classification task that commonly used in the machine learning.

K-nearest neighbor

K-nearest neighbor (k-NN) is a basic classification method which does not require any training. From statistics perspective, k-NN is a non-parametric classification method which means it needs no underlying assumptions about the data or its distribution pattern in general. In the binary classification, the value of k is usually set to an odd positive integer, because if it is set to be an even number, there is a possibility that the number of positive and negative samples in the k-nearest neighbor is equal. This can lead to a tie in the decision, i.e., two class labels achieving the same score because k-NN algorithm take majority as the class label for the given test sample.

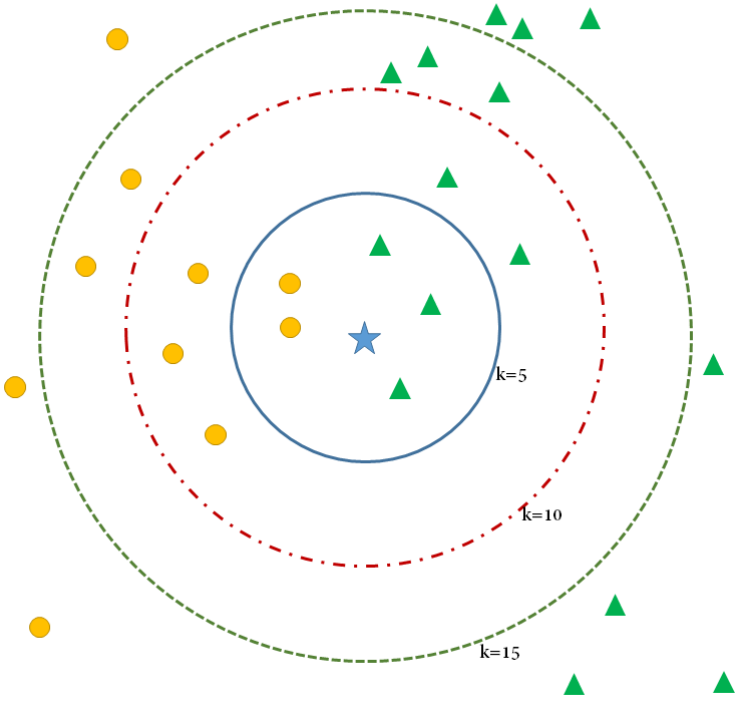

Figure 2.1: K-NN classifier with different value of k.

Figure 2.1 shows an example of k-NN classifier in determining the given test sample (marked with star) belongs to which class. If k is set to 5, the sample belongs to the triangle class since it is the majority in the 5 nearest neighbor of the star sample. However,

when it is set to 10, both triangle and circle class have 5 samples among its 10 nearest neighbors. The value k in k-NN also depends on the size of the training data. In the rule of thumb, k is set to be the square root of the number in the training data. In the imbalanced data case, samples of minority classes appear sparsely in the data space when there is uneven training data. When given a test sample, the computed k-nearest neighbors have a higher probability of samples from the majority classes. As a result, a given test data from minority classes are more likely to be classified wrongly. However, k-NN can be used in combination with sampling techniques such as over-sampling or undersampling to improve the classifier performance [2] . They reported that for the performance metric of sensitivity, k-NN gave the higher result than SVM and logistic regression (LR) when combined with oversampling technique.

Decision Tree C4.5

C4.5 is a famous algorithm of decision tree classification, which can be used to make a decision based on a certain set of data either univariate or multivariate. This classifier is flowchart-like tree structure where attributes or features are represented by internal node, decision rule is represented by branch, and the outcome is represented by leaf node. Each leaf note is assigned to the class that has majority value. Certain training algorithms are applied to a training dataset to automatically create the decision tree. Fiqure 2.2 shows an example of decision tree.

C4.5 is a type of decision tree which employs gain ration as a splitting criterion. C4.5 stops its splitting process when the number of samples to be split falls below a specified threshold. The least reliable branches are pruned using error-based pruning. In the case of class imbalance, decision trees may need to generate a lot of tests to differentiate between minority and majority classes. The split process may be stopped before the branches for forecasting minority classes are recognized in some learning procedures. Other learning procedures may prune the branches for forecasting minority classes as they are prone to overfitting. The reason for this is that accurately predicting only a few number of samples from minority classes yields insufficient success to considerably minimize the error rate, compared to the error rate generated by overfitting. Because the pruning in decision tree is mostly based on forecasting error, certain branches that predict small classes are likely to be deleted, and the new leaf node will be labeled with a dominating class.

Compared to another type of decision tree such as ID3, C5.0, and CART, C4.5 is said to be less affected by class imbalance and performs noticeably better than MLP (multi-layer perceptron), SVM, k-NN, and Naïve Bayes on medium datasets with high ratio of

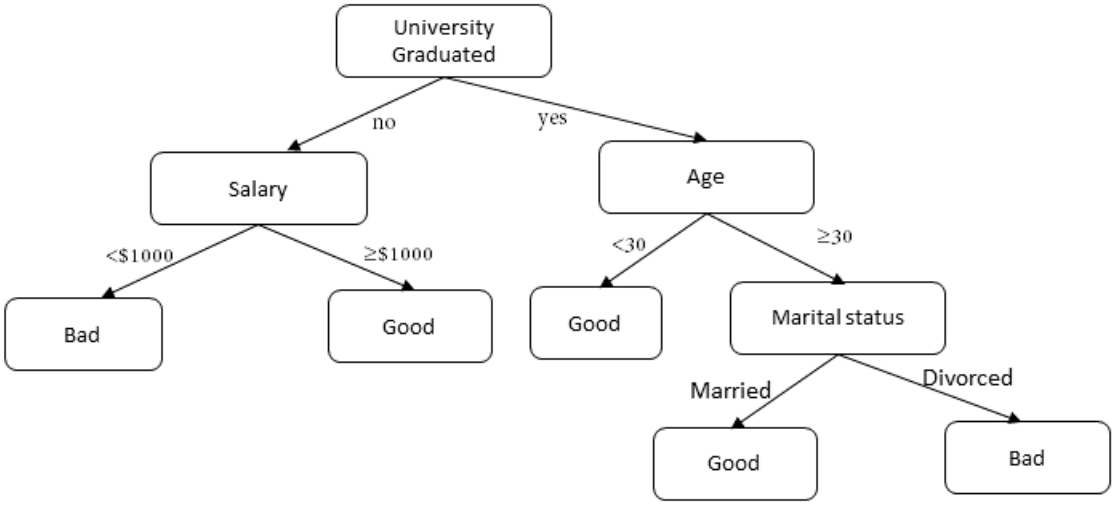

Figure 2.2: An example of decision tree.

class imbalance [3].

Random Forest (RF)

RF is also based on tree structure, however it utilizes many decision trees and random sampling. In random forest, the built forest is an ensemble of decision trees, which are commonly trained using the "bagging" method. The basic idea of "bagging" method is that combining several learning models improves the overall output. The final decision is taken from the majority of the trees which is chosen randomly by random forest.

From Figure 2.3, we can see that the majority voting of four decision trees is "Bad", which becomes the final prediction of the given sample. Besides for classification, random forest can be used for regression. Random forest is said to be a classifier that prone to class imbalance existence, but the inclusion of data sampling improves the classification performance of random forest classifier [4].

Support vector machine (SVM)

Support vector machine (SVM) becomes one of the most popular algorithms which achieves high performance in various kinds of two-class classification. In the case of binary class,

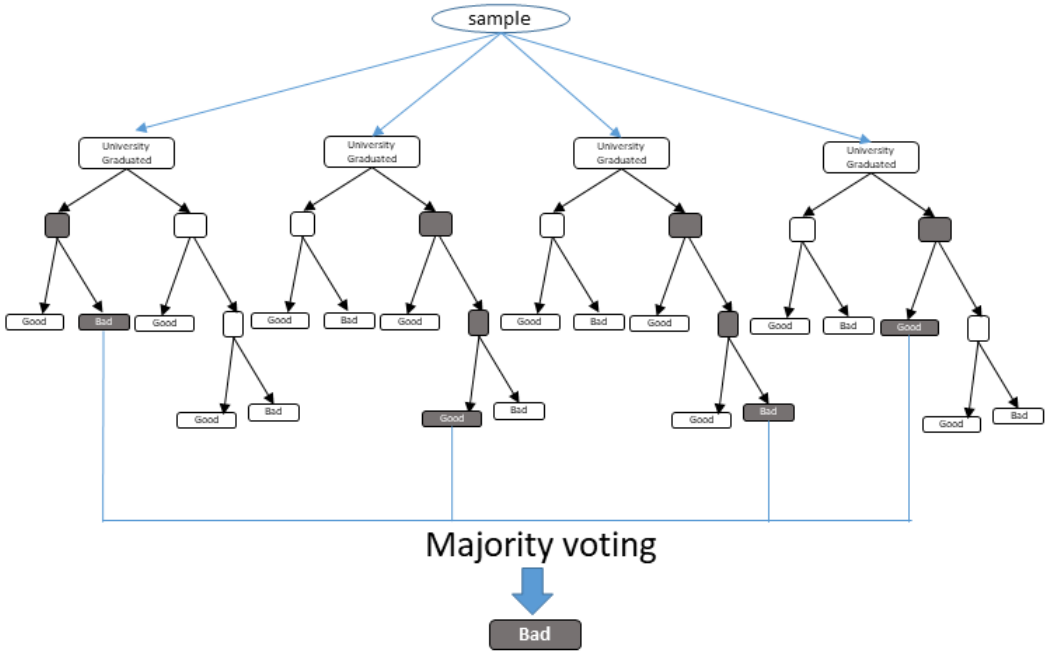

Figure 2.3: An example of random forest.

to accurately separate two classes, a SVM tries to find an optimal splitting line called as "hyperplane". The goal in SVM classification is to find the optimal hyperplane through training the SVM algorithm to the training dataset. A hyperplane with the greatest distance to the nearest training data points results in a good separation of classes as shown on Figure 4.

Figure 2.5 presents a given dataset that linearly separable. A hyperplane showed as straight-line function can be drawn to separate the samples into class "circle" and "triangle". SVM also can be used for both linearly or non-linearly separable data by set the kernel to be used in the algorithm. In the case of imbalanced data, compared to other classifiers, SVM is believed to be less vulnerable. This is because the class boundaries are generated using only a few support vectors so that the class sizes may not have a significant impact on the class boundary [5].

2.3.2 Unsupervised Learning

Contradiction with supervised learning, unsupervised learning utilizes a dataset without the right answers. It means that we have just the data itself without labels. This technique aims at uncovering hidden structure. The algorithm is developed to find cluster of the data which means it tries to find similarities or difference and group the data according to their similarities. Unlike classification, the output labels are not known beforehand instead, the

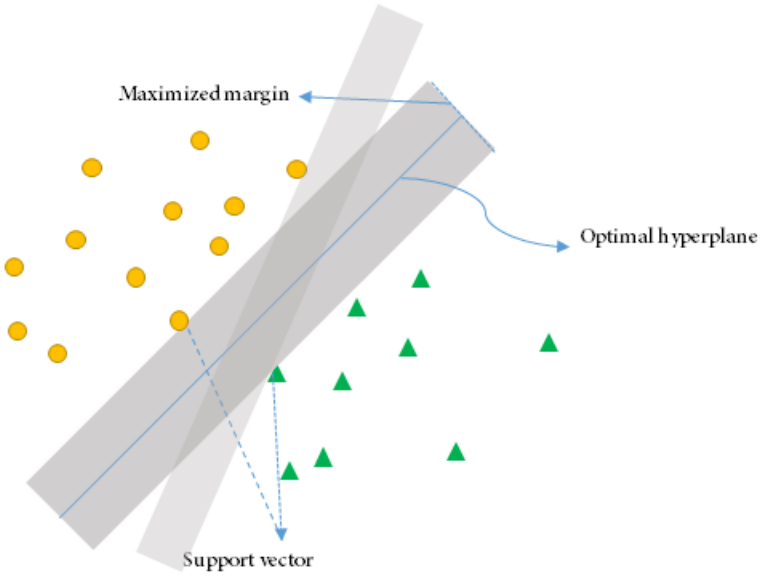

Figure 2.4: Optimal hyperplane.

algorithm creates the label itself. Google news is one example that relies heavily on these techniques. From the given news, it tries to decided whether it is criminal, market, or sport news based on the text data. Some common algorithms used in clustering task are k-means clustering and hierarchical clustering.

2.3.3 Semi-supervised Learning

Semi-supervised learning is used when the given dataset is partially labelled and left the remains unlabelled. This type of learning applied both supervised and unsupervised learning. This technique is often used when labelling massive amounts of data is time-consuming and costly. In this process, unsupervised learning is used to discover and learn the structure of the data, then the output of this learning is feed into supervised learning as training data and use the supervised learning model to predict on new unseen data. As an example of medical case, having clinicians manually label the data of medical imaging to detect cancer is an extremely expensive task. Furthermore, those clinicians may have more important priorities to respond to. So, we can see that the clinicians has labelled part of the dataset while leaving the remains unlabeled.

2.3.4 Reinforcement Learning

In the reinforcement learning, the algorithm learns from a series of reinforcement and feedback where it will be given rewards for a positive result and it will be given punishments for a negative result. This type of learning commonly used in a model built for video game. When starting the game, the model of the game has no idea how to play, then from every movement, the model gets a reward when it wins and receives punishment when it loses. The model then can learn from the feedback, and becomes able to identify which movements are good or bad.

2.4 Model Evaluation

After a model is built with training data, to determine the generalization accuracy of a model on future unseen data, the model's performance must be evaluated. Using test data, methods for evaluating a model's performance are divided into holdout and cross-validation.

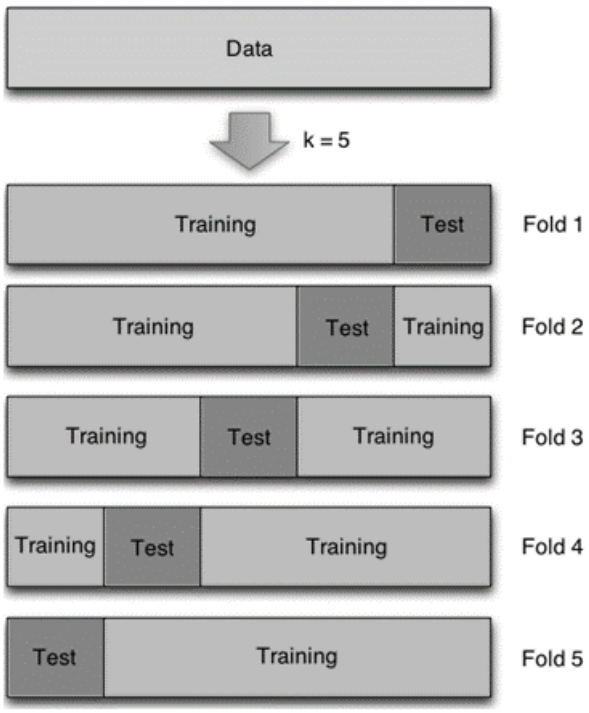

Figure 2.5: 5-fold cross validation.

In holdout method, the dataset is randomly divided into test data and training data. Sometimes, other than that two, the dataset also is divided into validation data. The

Table 2.1: Confusion matrix of binary class case.

	Predicted as positive	**Predicted as negative**
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

proportion of training and test data usually is set to 80 percent and 20 percent but it can be different based on the analyst preference. As for validation set, it is a subset of the dataset used to evaluate the model's performance throughout the training phase. It acts as a testing ground for fine-tuning model parameters and selecting the best performing model. A validation set isn't required for all modeling algorithms.

Cross-validation method used to partition the dataset into training set that is used to train the model, and a test set that is used to evaluate the model. There are some type of cross-validation such as k-fold cross validation, repeated k-fold cross validation, and leave one out cross validation. The first type is the most commonly used in machine learning. The k value in k-fold cross validation usually is set to 3, 5, or 10. When performing 5-fold cross validation, the data is divided into 5 equal set. The 4 part of the data is used as training set and the 1 part is used as test set of the model. This process is repeated for each of these splits and the accuracy is averaged of all these 5 trials. This process ensure that every data point is only used once in a test set and 4 times in a training set.The process of cross validation is shown in the the Figure 2.5.

2.5 Performance Metrics

There are three families of machine learning's performance metrics as we explain in this section.

2.5.1 Threshold Metrics

All the metrics in this family is based on confusion matrix. The input of this matrix is recorded from the number of samples of each class that were either correctly predicted or missclassified in each class. The entries for confusion matrix can be seen in the following table.

The TP indicates the number of samples that correctly predicted as positive when the actual value is positive. The FN value indicates the number of samples that wrongly predicted as negative when the actual value is positive. Some threshold metrics such

as accuracy, sensitivity, specificity, precision, F1-score, G-mean, and Cohen's kappa are derived from the value in the above table.

Accuracy

Accuracy is the most basic evaluation metric in machine learning. It defines the ratio of the number of samples that correctly predicted/classified in both positive and negative class versus the total number of the samples. This metric can be calculated as follow:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Accuracy only works well when there are an equal number of samples in each class. But when it comes to class imbalance situation, other metrics are advisedly used along together with accuracy. It happens because in the imbalance case, the model tend to classify a given sample as the majority class but still achieving high accuracy. When dealing with a rare disease, the cost of failing to diagnose a sick person's sickness is far more than the cost of subjecting a healthy individual to further testing. In this case, another metric such as sensitivity will give better interpretation than accuracy.

Sensitivity/Recall and specificity

The sensitivity metric is drawn from the positive samples. It shows the ratio of positive samples that correctly predicted as positive. In the opposite, specificity is drawn from the negative samples. It shows the ratio of negative samples that correctly predicted as negative. These two metrics are usually used in the medical domain data such as disease-related data, drug discovery, etc. Those metrics can be calculated as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.3}$$

Unlike accuracy that measure how often a classifier correctly predict the positive and negative samples, sensitivity and specificity also measure the same case, but in separate way or in each class.

Precision

Precision is the ratio between the TP and all the 'positive' values, TP and FP. It is drawn from the positive prediction value which means it is the measure of sample that correctly classified as positive out of all the positive prediction. Precision can be calculated as:

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

When it comes to reducing the amount of false positives, this measure is crucial since this metric put more focus on the positive class.

F1-score

F1-score is the Harmonic mean of the precision and recall/sensitivity. This score can be an indication of a good precision as well as a good recall. It can be calculated as:

$$F1 - score = 2 * \frac{precision * recal}{precision + recall} \tag{2.5}$$

High precision but low recall offers you an incredibly accurate result, but it also misses a big number of difficult-to-classify cases. The higher the F1 Score, the better our model's performance. If we want to weigh the contribution of each component as desired, we can change the value formula above as:

$$F_\alpha = \frac{(1 + \alpha)[precision * recall]}{[\alpha * precision] + recall}, \alpha \in R^+ \tag{2.6}$$

Usually, α can be 1,2, or 0.5 depends on the desired weigh. When $\alpha = 1$, as in $F1 - score$, it means that the weigh of recall and precision are equal. Basically, we can use $\alpha = 1$ if false negatives and false positives are equally important, $\alpha = 2$ if false negatives is more important, and $\alpha = 0.5$ if false positives is more important.

There are other types of threshold metrics such as Geometric Mean (G-mean), Fleiss' kappa, Cohen's Kappa, Macro Average Accuracy (MAA), or combinations of threshold metrics such as mean-class-weighted accuracy, optimized precision, adjusted G-mean, and index of balanced accuracy.

2.5.2 Ranking Methods and Metrics

Receiver Operating Characteristic (ROC) and AUC

ROC plots TP rate or sensitivity in the y-axis and FP rate or 1-specificity in the x-axis at every possible thresholds so that it is also called as probability curve. The closer the ROC curve to the top left corner, the better the classifier performance in predicting the class label. FPR or 1-specificity can be calculated from confusion matrix value as follow:

$$FPR = \frac{FP}{FP + TN} \tag{2.7}$$

ROC curve is considered as one of the evaluation metrics which well suited to class imbalance case since ROC separates performance of each class i.e sensitivity and specificity into two different measures. It can also be extended to problems with three or more classes by utilizing one versus all classes.

AUC is abbreviation of Area under the ROC Curve. AUC calculated the entire area under a two-dimensional ROC curve. The best classifier on average is said to have the highest AUC. Mathematically, using the Wilcoxon's rank sum statistic, AUC can be calculated as:

$$AUC(f) = \frac{\sum_{i=1}^{|T_p|} (R_i - i)}{|T_p|\,|T_n|} \tag{2.8}$$

whereR_i is the rank of ith example in T_p fiven by classifier f and $|T_p| \subset T$ and $|T_n| \subset T$ are the subsets of positive and negative examples in test set T respectively.

Precision and Recall Curve

As mentioned in the previous section, precision represents the percentage of data that is truly positive among the data classified as positive by the classification model in other words, it represents the certainty of positive judgment where hinger is better. While recall or sensitivity represents the percentage of actual positive data correctly classified as positive The Precision-Recall (PR) curve is a frequently used graphical tool for evaluating scoring function performance in terms of their ability to differentiate between two class. Therefore, the PRC becomes a more appropriate metric for imbalanced datasets than the ROC curve since it represents the connection between precision and recall (sensitivity).

In the comparison of ROC and PRC, the ROC curve has both the viewpoint that Positive can be judged as Positive and Negative can be judged as Negative. However,the PRC focuses only on the fact that Positive can be judged as Positive. Therefore, it is better to use the ROC curve that looks at the balance of both as a performance index of

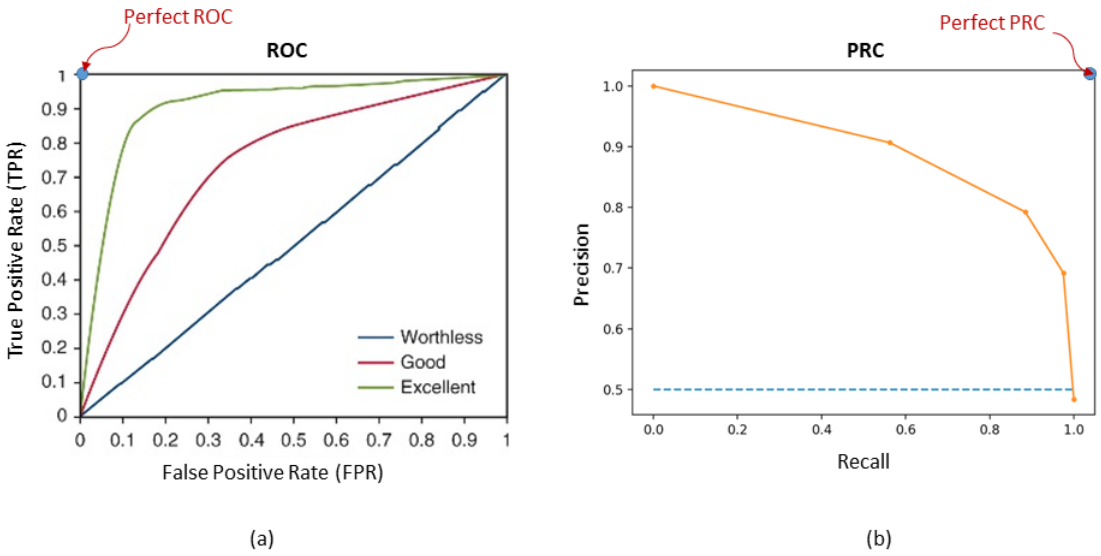

Figure 2.6: (a) comparison of ROC; (b) comparison of precision-recall curve (PRC)

the classifier, but if there are overwhelmingly many Negatives, it is accurate to judge the majority of Negatives as Negatives. And use PRC to see if you are only interested in the results of the Positive class as this metric sees whether a small number of Positives can be judged properly. The ideal state for PRC is where the curve sticks to the upper right. While ideal ROC is where the curve sticks to the upper left as can be seen in the Figure 2.6 below.

2.5.3 Probabilistic Metrics

The purpose of probabilistic metrics is to measure the uncertainty in a classifier's predictions. These are appropriate for issues where we're less concerned with correct vs. incorrect class predictions and more concerned with the model's prediction uncertainty and punishing predictions that are wrong yet extremely confident. Some widely used metrics in this family include root-mean-squared error, log loss (cross entropy),and brier score. It examines the performance of a classification model with a probability value between 0 and 1 as the prediction input. As the predicted probability diverges from the actual label, log loss rises. Brier score focused on the positive class or minority class in the case of class imbalance, which makes it is appropriate for imbalanced classification. A perfect classifier has root-mean-squared error and brier score of 0.0 and log loss of 0.0 with the worst is being infinitely positive.

2.6 Frameworks

The general framework to approaching the process of machine learning as described by Guo can be seen as these 7-steps:

1. Data collection : it starts with defining the problem to be solved or question to be asked with and gathering a dataset.
2. Data preparation : preparing and wrangling the data to make it suitable for the model.
3. Choose model : defining and choosing an appropriate model for the data.
4. Train model : developing model that can be improved so it gives better result than baseline model.
5. Evaluate model : choosing performance metrics and deciding on evaluation protocol such as cross validation.
6. Parameter tuning : model tweaking and regularization and hyperparameters tuning
7. Predicting

The framework of machine learning algorithm can be seen in the Figure 2.7.

Figure 2.7: Machine Learning Framework by Guo.

Chapter 3

Dealing with Class Imbalance and Features Extraction Methods

Imbalance datasets exist in many real-world data. Class imbalance happen when the number of a class is far less than that in the other one. The target class is usually the minority class or the class which has samples far less than the other one and a sample in this class is called as positive sample while a sample from the other one is called as negative sample. This problem can lead classifier to bias toward the majority class because it will most likely to predict a positive sample as negative sample. Therefore, a method to deal with class imbalance should be done first before feeding the data as an input to the classifiers to improve detection of minority sample or the performance metrics. There exist many methods to deal with class imbalance at either in algorithm level or data level.

3.1 Algorithm Level

At algorithm level, a method is proposed to modify the algorithm either change or add another line of algorithm in order to solve the imbalance data issue. Ensemble learning and cost-sensitive learning are some examples of algorithm-level methods. Bagging and boosting are classic ensemble learning methods that have shown to be effective in dealing with class imbalanced problems. This method includes modifying the already available classification algorithms to make them suitable for imbalanced data sets. In cost sensitive learning, a large misclassification cost is assigned to defective examples while a small misclassification cost is assigned to non-defective instances.

3.2 Data Level

At data level, there are some available form of re-sampling methods that were proposed. This method consists of over-sampling and under-sampling to re-balance the original dataset. The under-sampling method is done by reducing samples of the majority class whilst over-sampling method is done by adding samples of minority to the original dataset so that the new dataset become nearly balanced.

3.2.1 Down-sampling

Downsampling is a technique for reducing the number of training samples that belongs to the majority class. The majority class is subsampled to achieve basically the same number of sample as the minority class. For example, if we have a training data with 900 of it are negative samples and the remain 100 are positive sample. Down-sampling would randomly sample the first class so that we will have 100 samples of each positive and negative class. The main drawback of downsampling if that we tend to lose a lot of important information when we remove the data.

3.2.2 Up-sampling

Up-sampling is a technique to increase the number of training samples of the minority class by randomly duplicating the minority class. Using this method, the numbers of both classes are (nearly) the same. As the same example of downsampling, after upsampling is done, in the final dataset, each positive and negative class will have a training data of 900 samples. This balancing technique keeps the model from bias toward the majority class. The relationship (border line) between the class labels is also unaffected. Furthermore, because of the added samples, the upsampling method introduces bias into the system.

3.2.3 Under-sampling

Undersampling the majority class is one of the most popular and simplest approaches to handle imbalanced data. In order to effectively balance the class distribution, undersampling methods eliminate samples from the training dataset that belong to the majority class, such as lowering the skew from a 1:1000 to a 1:100, 1:10, 1:3, or even a 1:1 class distribution. There are some methods of undersampling, and in this chapter we will take a look only on four methods.

Random Undersampling

The simplest or basic undersampling method is by random undersampling. In this method, we can simply choose n samples at random from the majority class, where n is the number of samples for the minority class in the training data. Despite simple and effective, this method has the drawback of removing samples without regard for how important or significant they may be in defining the decision border between the classes. We potentially lose important data in this process. The wiser method for undersampling are Near Miss and Condensed Nearest Neighbor undersampling.

Near Miss

A near-miss method can contribute in the balancing of an imbalanced dataset. This undersampling algorithms is a good technique to balance the data that done by examining the class distribution and removing samples from the majority class at based on their distance to other samples in the minority class. When two samples of different classes are relatively near to each other, this method will remove the sample of the majority class to balance the distribution.

There are three versions of this near miss method.

1. Version 1: The data is balanced by selecting the sample of majority class with the minimum average distance to the three nearest samples of minority class.

2. Version 2: The data is balanced by selecting the sample of majority class with the minimum average distance to the three furthest samples of minority class

3. Version 2: The data is balanced by selecting the sample of majority class with the minimum average distance to the each samples of minority class.

Condensed Nearest Neighbor (CNN)

The aim of CNN undersampling is to select a subset of the training set T let's say U, such that every point in T has a neighbor in U who belongs to the same class. According to More, A (2016), U can be chosen through this steps::

1. Choose a random sample from T and set $U = p$.

2. Scan $T - U$ and add to U the first sample found which its nearest neighbor in U is from a different class.

3. Repeat step 2 until U is maximal.

Because CNN undersampling involves multiple runs over the training data, it becomes slower than other methods. Furthermore, the subset chosen might vary significantly due to the randomness related to the selection of samples at each iteration. The method of condensed nearest-neighbor (CNN) also selects samples at random. As a result redundant samples are retained and internal rather than boundary samples are occasionally retained.

Tomek Link

A pair of a positive sample and a negative sample is called a Tomek link if they are the closest neighbors to each other. This undersampling method is done by deleting all tomek links from the dataset. Another option is to simply delete only the majority class sample from a Tomek link. In practice, the Tomek Link method is commonly combined with other methods, such as the Condensed Nearest Neighbor Rule.

3.2.4 Over-sampling

On the contrary of undersampling methods, over sampling is used when the number of obtained data is insufficient. To balance the classes, oversampling approaches increase the number of objects in the minority class. There are some popular oversampling methods which most of them is developed based on SMOTE. In this study, seven methods of oversampling are explained.

SMOTE

SMOTE is known as the pioneer method for oversampling an imbalance dataset. The method use k-nearest neighbors in creating new synthetic samples to balance the class distribution of the dataset [6]. Each positive sample is pairing with its nearest neighbors then along a line connecting the samples with one of the selected nearest neighbors, synthetic sample is generated. Mathematically, the process can be described as follow. A new synthetic samples s' is computed by $s^{'=s+gap(ls-s)}$ where $gap \in [0, 1]$ and ls is a selected neighbor from k-positive nearest neighbors of point s. Then, the process is repeated until the number of positive and negative samples become balanced. In dealing with class imbalance data by creating new synthetic samples, SMOTE is the pioneer methods where many methods in this issue were developed in order to improve classification performances based on this technique.

ADASYN

[7] introduced another technique to deal with class imbalance called ADASYN. The main idea of this method is to create synthetic samples based on the level of difficulty in learning the samples of minority class. The procedure to calculate new synthetic sample with ADASYN is as follows. Training datasets with m samples $\{x_i, y_i\}$, where $i = 1, \ldots, m$, x_i is data sample in R dimensional feature pace $\mathbf{X}$, and $y_i \in Y = 1, \ldots, C$ where C is class label. The data is divided into m_{min} and m_{maj}, where m_{min}+$m_{maj} = m$ and $m_{min} \leq m_{maj}$. m_{min} and m_{maj} are the number of minority class and the number of majority class respectively. First, determine the value of $\beta \in [0, 1]$, with β is parameter to determine the level of balance. Then calculate the desired number of new synthetic samples to be generated:

$$G = \left(m_{maj} - m_{min}\right) \beta \tag{3.1}$$

If $\beta = 1$, then the dataset will become perfectly balanced. Second, determine k-nearest neighbors for each x_i in minority class to calculate r_i value as the ratio of majority samples in k nearest neighbors of x_i. r_i is defined as:

$$r_i = \Delta_i / K, \quad i = 1, .., m_{min} \tag{3.2}$$

where Δ_i is the number in K nearest neighbors of x_i that belongs to m_{maj}. From the above formula, it can be seen that $r_i \in [0, 1]$. r_i shows the difficulty of learning, where the higher the value of r_i, the more difficult the classification algorithm to learn the dataset. Third, with $S = \sum_{i=1}^{i=m_{min}} r_i$, normalize r_i by:

$$\widehat{r_i} = r_i / S \tag{3.3}$$

so that $\sum_{i=1}^{i=m_{min}} \widehat{r_i} = 1$ shows that $\widehat{r_i}$ is a density distribution. Fourth, for each x_i in minority class, calculate g_i for the number of synthetic samples to be generated by:

$$g_i = \widehat{r_i} \times G \tag{3.4}$$

Finally, a synthetic sample can be generated by:

$$s_i = x_i + (x_{ji} - x_i) \lambda \tag{3.5}$$

where x_{ji} is randomly selected positive sample from k-nearest neighbors of x_i and $\lambda \in [0, 1]$. This process is repeated until all the positive samples is used in generating a

set of synthetic sample.

Borderline SMOTE

There also exist some oversampling techniques developed based on SMOTE. One of them is Borderline SMOTE introduced by [8]. This technique developed based on the SMOTE by putting focus more on the borderline of each class. It highlighted the problem that a sample located on or nearby the borderline tend to be misclassified by classification methods than that located far from the borderline. Therefore, a sample located on or near the borderline will give more contribution to classification. From that reason, this technique tries to strengthen the borderline minority samples by generating synthetic samples along this region only. The procedure to synthesize new samples with this technique can be described as follows. First, find the borderline of minority samples. It defines three region of positive instance by considering the number of negative instance on k-nearest neighbors of each positive sample. Suppose that S is the training set that divided into Mj and Mn as the majority and minority class respectively, and $Mj = \{y_1, y_2, \ldots, y_{n\_maj}\}$, $Mn = \{x_1, x_2, \ldots, x_{n\_min}\}$ where n_maj and n_min are the number of majority class samples and the number of minority class samples in k-nearest neighbors of $x_i \in minority$. The region of x_i can be described as noise if the number of $n_{maj} = k$. If $0 \leq n_{maj} \leq k$ the region is described as safe, while if $k/2 \leq n_{maj} \leq k$ then the region is described as easily misclassified and put x_i into DANGER set so that $DANGER \subseteq Mn$ and $DANGER = \{x'_1, x'_2, \ldots, x'_{d\_min}\}$, $0 \leq d\_min \leq n\_min$. Second, determine the desired number $n \times d_{num}$, $n \in [1, k]$ of synthetic minority samples to be generated. For each $x'_i \in Mn$, select n nearest neighbors from its k-nearest neighbors in Mn, so that we have $diff_j$, $j = \{1, 2, \ldots, n\}$ as the difference between x'_i and its nearest neighbors from Mn. Therefore, the synthetic samples can be generated by $syn_j = x'_i + r_j \times diff_j$. This idea of Borderline SMOTE in selecting certain region to generate synthetic samples inspired other SMOTE-based techniques in oversampling methods such as safe-level SMOTE and relocating safe level SMOTE.

Safe-level SMOTE

[9] introduced another improvement of SMOTE, namely Safe-level SMOTE. This technique highlighted the drawback of SMOTE that SMOTE naively ignore nearby majority instances in synthesizing the minority samples along a joining line of a minority samples and its selected nearest neighbors. This new generated synthetic sample by SMOTE cause a classification model tends to create larger and less specific region resulting in overgen-

eralization. Therefore, safe-level SMOTE will determine whether a sample in minority class is safe to use in generating synthetic samples or not. In other word, Safe-Level SMOTE is designed to generate synthetic samples around selected positive samples that considered as safe. The procedure of this technique can be described as follows. First, the process is done by calculating k-nearest neighbors of each positive sample $x_i \in minority$, then safe-level value is calculated by counting the number of positive instance in its k-nearest neighbors. x_i is considered as noise and will be excluded from the next step if there is no positive instance in k-nearest neighbors of x_i. For the rest of positive instance $x_i \in minority$, $\hat{x}_i$ is a randomly selected positive sample of its k-nearest neighbors. Then define safe level ratio sl_R as:

$$sl_R = sl_{x_i}/sl_{\hat{x}_i} \tag{3.6}$$

where sl_{x_i} and $sl_{\hat{x}_i}$ are the number of positive samples in k-nearest neighbors of x_i and the number of positive instance in k-nearest neighbors of $\hat{x}_i$ respectively. By considering the above values, a synthetic sample s is generated according to these criteria: 1) if $sl_R = \infty$ and $sl_{\hat{x}_i} = 0$, a synthetic sample will be generated in range $[x_i, x_i]$; 2) if $sl_R = 1$ and $sl_{\hat{x}_i} \neq 0$, a synthetic sample will be generated in range $[x_i, \hat{x}_i]$; 3) if $sl_R > 1$ and $sl_{\hat{x}_i} \neq 0$, a synthetic sample will be generated in range $[x_i, (x_i - \hat{x}_i)/sl_R]$; and 4) if $sl_R < 1$ and $sl_{\hat{x}_i} \neq 0$, a synthetic sample will be generated in range $[\hat{x}_i - sl_R \bullet (x_i - \hat{x}_i), \hat{x}_i]$. After completing the process, a set of synthetic samples is combined to original dataset to obtain a new, nearly balanced dataset.

Relocating Safe-level SMOTE

This method is extended work of Safe-Level SMOTE which synthesize new sample in a safe region by avoiding majority samples. It highlights the fact that safe-level SMOTE ignored the possibility that some synthetic samples are generated closer to negative samples than to positive samples. It contradicts with the procedure in generating synthetic sample of safe-level SMOTE where it tries to avoid negative samples in generating synthetic samples. Therefore, [10] introduced an additional algorithm to relocate generated synthetic samples if it locates around negative samples. After the process in generating synthetic sample s with Safe-level SMOTE is done, then the distance of s to x_i dan $\hat{x}_i$ are calculated. This two values are compared to the distance between s and its closest negative samples. If the distance is less than the closest distance either to x_i or $\hat{x}_i$, the synthetic sample is relocated closer to that point.

Density-Based SMOTE

Another variation of SMOTE-based oversampling technique is Density-based minority over-sampling technique or DBSMOTE. It is an integration of DBSCAN and SMOTE. DBSCAN or Density-Based Spatial Clustering of Application with Noise is proposed by [11] aiming at discovering clusters of arbitrary shape. DBSMOTE is proposed by [12] to create synthetic samples of an arbitrarily shaped cluster found by DBSCAN. Inspired by Borderline-SMOTE in maintaining detection rate of majority class, DBSMOTE focused the work on over-lapping region. However, this technique also highlights the drawbacks of Borderline-SMOTE which fails in maintaining the detection rate of negative samples while improving that on positive samples. To resolve this drawback, DBSMOTE developed different approach to precisely oversampling both in the over-lapping and the safe region to improve positive sample detection while maintaining detection rate on the negative samples. In this technique, by Definition 1, underlying weighted directional graph term is introduced.

Definition 1

A directly density-reachable graph of a cluster C discovered by DBSCAN is denoted by G(C) = (V, E), where V is a set of nodes represented as instances in C and E is a set of edges where $E = \{(v_1, v_2) \in V \times V |$ an instance v_1 is directly density-reachable instance v_2 w.r.t. Eps and MinPts or vice versa$\}$. Let $w : E \to R$ be a weight function where $w(v_1, v_2)$ is set as a distance nodes $v_1, v_2 \in V$.

The algorithm DBSMOTE synthesizes a minority instance along the shortest path retrieved in a directly density-reachable graph from each sample to the pseudo-centroid cluster. Define S as original dataset which then is parted to M_{min} and M_{maj} for minority class and majority class respectively. DBSCAN starts generating $C_1, C_2, \ldots, C_m$ of disjoint clusters to detect a set of noise samples N which eroded away in the next step from the minority class M_{min}. From generated disjoint clusters, DBSMOTE then generated $C'_1, C'_2, \ldots, C'_m$ set of synthetic samples which added to the original dataset S to create an oversampled dataset S'.

Adaptive Neighbor SMOTE

While methods mentioned above focused on where to generate synthetic samples, Adaptive Neighbor Smote (ANS) put its focused on how many neighbors needed to synthesize a new sample [13]. In other word, ANS concentrates on deciding the appropriate value for parameter k in k-nearest neighbors of each positive sample that is needed in SMOTE and SMOTE-based algorithms to synthesize a new sample. The parameter k is selected

according to density level of each positive sample's region. Other than that, this technique tries to utilize noise sample which usually excluded from the process in generating synthetic samples by other techniques. The procedure began with detecting noise sample(s) and excluding it from the entire dataset. Then, choose the maximum distance value that connects a positive sample $x_i \in minority$, so that it has at least one nearest neighbor of positive instance. Then, k is defined as the total number of positive instance in under the previously chosen distance. SMOTE is then performed for each x_i with different k depends on its density. By utilizing the value of k, the area of the generated synthetic samples will be more spread out inside the dense area and not sparsely distributed as in SMOTE.

3.3 Features Extraction

Feature extraction is a process meant to reduce the number of variables in a high-dimensional dataset by creating new variables from the existing variables without losing the information of the original dataset. This process is done for some purposes such as to reduce the problem arising from high dimension of a dataset, to increase computational efficiency, or to visualize the dataset either in 2D or 3D.

3.3.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is one of feature extraction methods that commonly used to reduce the dimension of a large dataset. PCA is a statistical technique of features extraction that uses orthogonal transformation to transform a set of data of possibly correlated variables into a set of values of linearly uncorrelated variables. [14]. Generally, the procedure of this method can be described as follows. First, the range of the original variables is standardized to make all the variables contribute equally to the analysis. After the standardization is done, compute the covariance matrix to identify the correlation of each variable to the others. Then eigenvectors and eigenvalues of the matrix are computed to identify the principal components. Once it is done, the eigenvectors are ordered by their eigenvalues in descending order. Eigenvalues which are less significant are excluded and the rest eigenvalues form Feature vector. Finally, the result of feature extraction with PCA is obtained by:

$$final\_dataset = Feature\_vectors^T * Standardized\_dataset^T$$

3.3.2 Independent Component Analysis (ICA)

Another linear methods of feature extraction using component analysis is Independent Component Analysis (ICA). ICA is an important method in signal-based analysis such as EEG signal to help separating normal and abnormal signals. The general framework of ICA was introduced in 1980s by some researcher which aims at extract hidden factors of a dataset by transforming the variables to a new set of variables that is maximally independent. What distinguished ICA from PCA is that PCA assumes that signals are subject to multivariate Gaussian distribution and uses orthogonal bases to decompose signals. There exist some methods to perform ICA for reducing dimensionality of dataset. [15] describes one way to perform ICA through three steps dimensionality reduction. First is to remove the average of dataset. The second step is to remove the correlation between components using the covariance matrix. And the last step is to and maximize non-Gaussian components using kurtosis or negentropy method to measure the non-Gaussian's level.

3.3.3 T-Distributed Stochastic Neighbor Embedding (tSNE)

Other than linear dimensionality reduction techniques, there are some available techniques that are nonlinear. One of those is t-distributed stochastic neighbor embedding (tSNE) first introduced by [16]. It reduces the dimensionality of a dataset by giving each data point a location in 2D or 3D dimensional map. This technique aims at identifying the relevant pattern of a dataset while maintain its local structure [13]. For each point of a datasets, tSNE models the probability distribution of other points which are closest to it. One of the most im-portant parameters to be set when using t-SNE is perplexity, which is the expected num-ber of nearest neighbors each point has. The performance of tSNE is fairly robust under different settings of this parameter. Generally, the perplexity is set depend on the size of the dataset. The default value of perplexity in some packages is set to 30 for a dataset whose variables more than 30.

3.3.4 Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP)

Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) is another nonlinear dimensionality reduction which builds a mathematical theory to justify the graph-based approach [17]. It is developed based on ideas from topological data analysis and manifold learning techniques, which assumes that the data is uniformly

distributed on the manifold. To make this assumption true, it defines a Riemannian met-ric on the manifold. Compared to tSNE, UMAP provides much faster computational run-ning time.

Chapter 4

Machine Learning Algorithms for Predicting Chronic Obstructive Pulmonary Disease (COPD) from Gene Expression Data with Class Imbalance

4.1 Introduction

Chronic obstructive pulmonary disease (COPD) is a progressive inflammatory lung disease that reduces lung airflow and has a substantial impact on patients' everyday life. COPD has become one of the most important risk factors for lung cancer [18]. According to the World Health Organization, COPD was responsible for 5% of all deaths worldwide in 2015, and by 2020, there will be 4.7 million of the 68 million deaths worldwide caused by COPD [19]. When COPD has caused significant lung damage, it has just been detected some time later which means it is too late. COPD is difficult to identify early on since symptoms don't develop until the lungs have been severely damaged. Opportunities for early detection of COPD can be improved with the helps of existing computational technology, development of machine learning algorithms, and better access to health and disease-related data. [20] emphasized the need of using machine learning algorithms in the development of Clinical Decision Support Systems to classify the various phases of COPD in patients. [21] used machine learning methods to identify 38 genes linked to the pathophysiology of COPD and ILD (interstitial lung disease). The genes that has been identified can help in the development of improving COPD and ILD treatments. Gene expression data is extensively used in disease research, which can identify components

of the genome that are significantly changed. It helps us understand which biological processes are affected [22]. However, because the number of experiments or samples is less than the number of genes or probes that are typically employed as attributes or features, gene expression data analysis and management becomes complex and challenging task. Furthermore, platform differences that cause batch effects, varied experimental circumstances, and a lack of standardization in experimental annotation pose a significant obstacle.

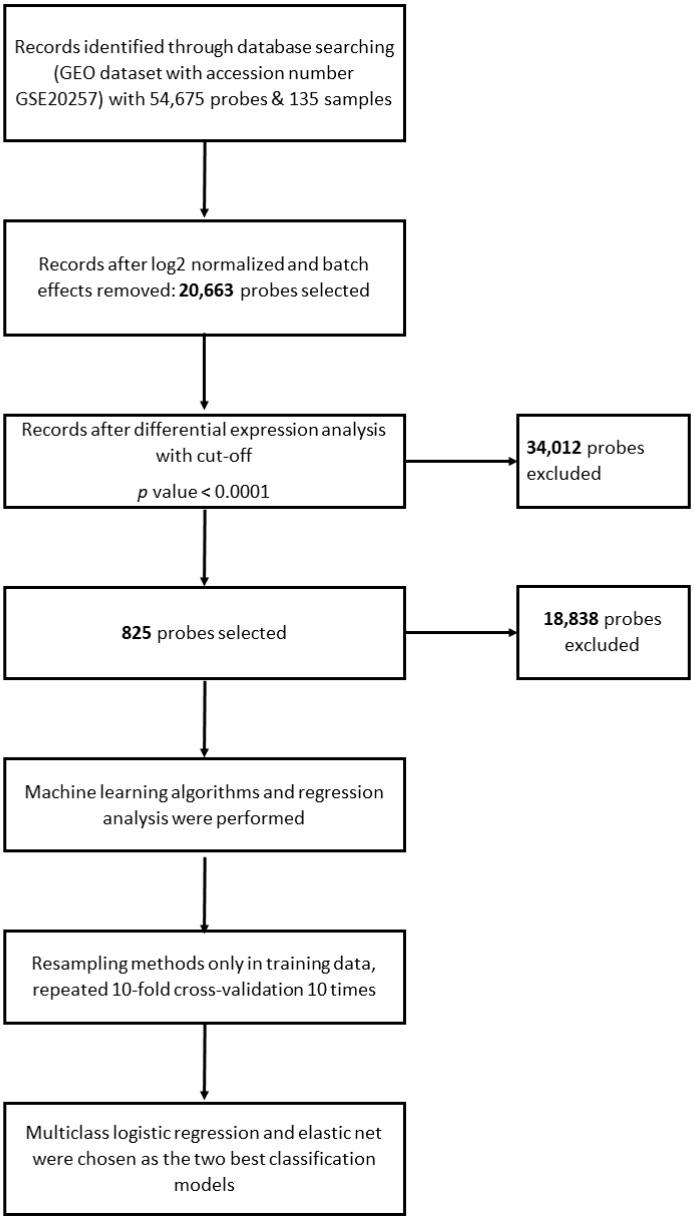

Figure 4.1: Flowchart of this study.

The presence of class imbalance, i.e., the number of data represented in one class is

smaller than in other classes, makes this task considerably more difficult. The minority class is usually the target class, but a classifiers will most likely to underperform in this category while bias towards the majority. One of the methods to deal with class imbalance is by resampling the original dataset either by oversampling or undersampling [6]. The goal of this work was to use machine learning techniques to solve the problem of class imbalance through the synthetic minority oversampling technique (SMOTE). We used the "caret" package to run additional resampling approaches for comparison. This figure above shows the flowchart for this study.

The rest of this document goes into our methods in details. The material and methods used in this study are described in section 2. It briefly discusses about data selection methods of improving model performance. We briefly discuss the machine learning and regression methods and approaches for this study, and the evaluation metrics that we use to evaluate the performance of our method proposed. The experiment and results are discussed in Section 3. Finally, section 4 brings this paper to a conclusion.

4.2 Material and Methods

We utilized microarray dataset of the small airway epithelium (SAE) downloaded from the Gene Expression Omnibus (GEO) database, https://www.ncbi.nlm.nih.gov/geo/, with accession number GSE20257. Bronchoscopy and brushing were used to acquire the airway epithelial cells, which were performed by Crystal Laboratory at Weill Cornell Medical College's Department of Genetic and Medicine. The data was first collected on June 27, 2011 and was most recently updated on March 25, 2019. Gene expressions are organized in GeneChip HG-U133 Plus2.0 arrays, which represent roughly 14,500 well-characterized human genes which can be used to investigate human biology and disease processes [23].

A total of 54,675 probes are included in the collection, which contains gene expression data from 135 human patients. There are 23 smokers with COPD (9 GOLD stage I, 12 GOLD stage II, and 2 GOLD stage III) among the 135 individuals, 59 healthy smokers, and 53 healthy nonsmokers. Bioconductor's "affy" and "biobased" R packages were used to log2 normalize the data and remove batch effects respectively. The "Limma" packages was then used to identify probes that were significantly different in healthy nonsmokers compared to COPD patients by utilizing differential expression analysis. The machine learning algorithms then employed the chosen probes. The probe selection aims to reduce the dataset's dimension, which is important for lowering the computational cost of modeling. Additionally, deleting unwanted, irrelevant, and duplicated attributes that

statistically do not contribute to the predictive model's accuracy and other evaluation metrics might improve the model's performance.

4.3 Machine Learning Methods

For the classification task, we used various machine learning models such as support vector machine (SVM), naïve bayes, random forest, gradient boosting machine (GBM), and regression models such as elastic net regression and multiclass logistic regression (LR). The R packages "caret," "e1017," "nnet," and "naivebayes" were used to implement all machine learning and regression algorithms.

Elastic net is a regularized regression model that employs an L1 and L2 linear combination penalty. The strengths of the other two regularized regression models, ridge and lasso regression, are combined in this Elastic net model. In this regression model, parameter has a value between 0 and 1. The goal of this regression model was to reduce the loss function. Multiclass logistic regression is a extension of binary logistic regression. This model is used to predict categorical response variables with more than two outcomes. The goal of this model is to capture the linear relationship between the response and independent variables.

4.4 Evaluation Metrics of Predictive Models

The mean accuracy, AUC, sensitivity, and specificity for each of the proposed classification models were analyzed to assess the model's performances. The ability of a model to distinguish positive and negative outcomes of a disease-related dataset is evaluated using sensitivity and specificity [24]. Confusion matrix is used to calculate various evaluation metrics such as accuracy, sensitivity, and precision as described in Chapter 2. This table below illustrates the nine potential outputs for the three classes 1, 2, and 3. It shows the elements of three-dimensional confusion matrix described [25].

In the table below, the columns represent the predicted classes, and the rows represent the actual classes. We then have the numbers of nine cases where TP_1 is the case for which the classifier predicted as class-1 and the sample were actually class-1, and E_{12} is a sample from class-1 that misclassified as class-2. Thus, the false negative in the class-1 (FN_1) is the sum of E_{12} and E_{13} ($FN_1 = E_{12} + E_{13}$) which indicates the sum of all samples that were actually class-1 but were misclassified as class-2 or class-3. Whereas the false positive in the class-1 (FP_1) is the sum of E_{21} and E_{31} ($FP_1 = E_{21} + E_{31}$) which indicates the sum

Table 4.1: An illustrative example of the confusion matrix for a 3-class classification test.

	Predicted as 1	**Predicted as 2**	**Predicted as 3**
Actual 1	$\mathbf{TP}_1$	E_{21}	E_{31}
Actual 2	E_{12}	$\mathbf{TP}_2$	E_{32}
Actual 3	E_{13}	E_{23}	$\mathbf{TP}_3$

of all sample that actually were not class-1 but were misclassified as class-1.

In the "caret" packages, the accuracy is defined as the overall accuracy using the predicted classes, while sensitivity and specificity are defined as the averages of the "one versus all" statistics. As described in Ballabio et al., (2018), the overall accuracy is computed as follows:

$$Acc = \frac{\sum_{i=1}^{l} TP_i}{n} \tag{4.1}$$

where TP_i is the number of true positive samples in class-i, and n is the total number of samples. Accuracy shows how accurate our classification model is able to predict the class labels given in the problem statement. In other word, the best selected model has the highest accuracy.

Sensitivity for multiclass classification is computed as follows:

$$Sn = \frac{\sum_{i=1}^{g} Sn_i}{g} \tag{4.2}$$

where Sn_i is sensitivity for class-i and g is the total number of classes. Sn_i can be calculated as follows:

$$Sn_i = \frac{TP_i}{TP_i + FN_i} \tag{4.3}$$

On the other hand, specificity for multiclass classification is computed as follows:

$$Sp = \frac{\sum_{i=1}^{g} Sp_i}{g} \tag{4.4}$$

where Sp_i is specificity for class-i. Sp_i can be calculated as follows:

$$Sp_i = \frac{\sum_{\substack{k=1 \\ i \neq k}}^{g} (n_k - E_{ik})}{n - n_i} \tag{4.5}$$

Sensitivity shows the ability of a model in correctly identifying positive data out of all actual positives data. In contrast, specificity shows the ability of a model in correctly

identifying negative data out of all actual negative data. The higher the sensitivity and specificity, the better the model in correctly identifying data that belong to a certain class as well as a data that do not belong to the class.

To calculate AUC score, we used multiclass.roc function from pROC packages which computed multiclass AUC as an average AUC defined by [26]. For multiple classes labelled as $0, 1, 2, \ldots, (c-1)$ with $c > 2$, the separability between class i and j or auc is defined as follows:

$$auc = \frac{\hat{A}(i|j) + \hat{A}(j|i)}{2} \tag{4.6}$$

where $\hat{A}(i|j)$ is the probability shows that if we draw a member of class j randomly, the estimated probability of j belongs to class i will be lower than if if we randomly draw a member of class i instead. This also applies to the reverse case. For multiclass case $\hat{A}(i|j) \neq \hat{A}(j|i)$.

$$AUC = \frac{2}{c(c-1)} \sum aucs \tag{4.7}$$

with aucs all the pairwise roc curves.

The best model is selected based on the highest value of the four evaluation metrics. The higher the AUC, the better the model in distinguishing a positive example from a negative one.

4.5 Evaluation of Resampling Methods

We applied the SMOTE algorithm in the experiment with two distinct CRAN packages, "DMwR" and "smotefamily." For comparison, we employed down-sampling and up-sampling methods. Up-sampling involves sampling a dataset randomly so that all classes have equal number as the majority class. Down-sampling, on the other hand, will sample a data set randomly so that all classes have equal number as the minority class.

We utilized repeated k-fold cross-validations to assess the performance of all the classifiers, as this is a standard approach for this purpose. By repeating the k-fold cross-validation method n times and providing the mean result of all folds from all runs, this evaluation technique enhances the performance of machine learning and regression algorithms. According to [27], repeated cross validation is an effective technique for reducing the complexity of regression and machine learning models.

4.6 Experiment and Result

During the data collecting process, gene expression data frequently contains unnecessary, irrelevant, and duplicated data. We eliminated unnecessary data in the first stage before conducting the classification model so that our proposed classification method would be more accurate. The raw data obtained from the GOE dataset was log2 normalized using the "biobased" R package, batch effects and unnecessary variation were eliminated using the "affy" package, and statistically compared or analysed for differential expression using the "Limma" tool.

Out of 54,675 probes, 20,663 were chosen after batch effects were removed in the data pre-processing. As shown in this figure below, we found 825 probes that were significantly different in COPD patients compared to healthy non-smokers using a p-value of 0.0001 . The dataset was split into two parts: a training set and a test set, each with a proportion of 80% and 20%, respectively. SMOTE was then used to resample the data only in the training data. With repeated 10-fold cross-validations 10 times, the oversampled data were included in machine learning and regression modeling techniques.

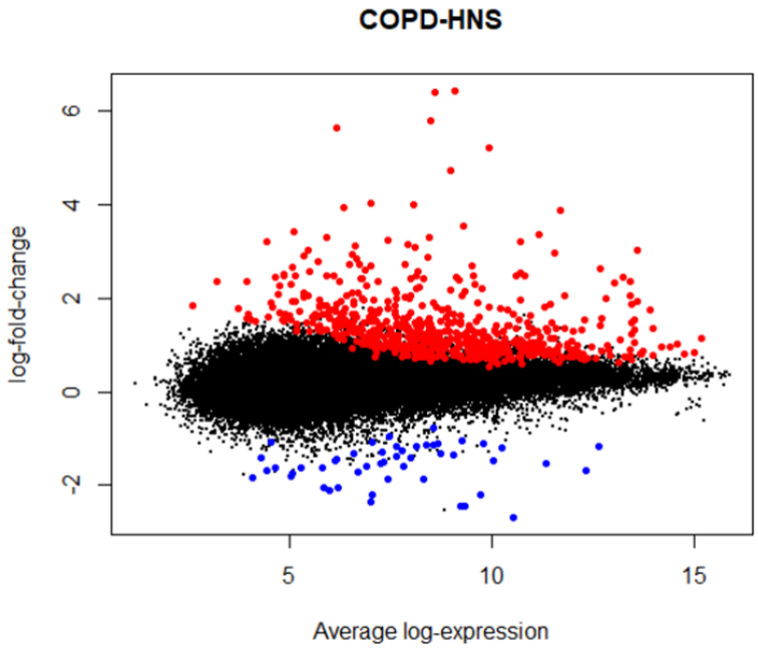

Figure 4.2: Mean difference (MD) plot displays log2 fold change versus average log2 expression values for all the 54,675 probes. Highlighted genes are significantly differentially expressed in COPD compared to healthy non-smoker (red = upregulated, blue = downregulated)

Table 4.2: Accuracy and AUC for different models.

Classifier	**Accuracy** (%)	**AUC** (%)
SVM	68	73
+SMOTE	68	85
Naïve Bayes	48	70
+SMOTE	64	76
Random Forest	48	60
+SMOTE	64	81
GBM	64	81
+SMOTE	56	70
Elastic Net	64	71
+SMOTE	76	89
Multiclass LR	72	82
+SMOTE	80	90

4.7 Comparison of Machine Learning Algorithm and Regression Analysis

This tabel below shows the comparison of the accuracy and AUC scores of SVM, naïve bayes, random forest, GBM, elastic net regression, and multiclass LR models with and without SMOTE to handle with class imbalance.

Table 4.3: Average sensitivity and specificity for different models.

Classifier	**Sensitivity**	**Specificity**
SVM	0.53	0.81
+SMOTE	0.70	0.82
Naïve Bayes	0.44	0.71
+SMOTE	0.67	0.80
Random Forest	0.37	0.69
+SMOTE	0.61	0.81
GBM	0.57	0.80
+SMOTE	0.50	0.76
Elastic Net	0.56	0.80
+SMOTE	0.76	0.87
Multiclass LR	0.67	0.84
+SMOTE	0.80	0.89

Except for GBM, all performance increased in the models with SMOTE compared to those without SMOTE after repeated 10-fold cross-validation 10 times. This showed that

SMOTE is effective in handling class imbalance. Multiclass LR with SMOTE achieves the best results, with an overall accuracy score of 80 percent and an AUC of 90 percent, respectively. As seen in the table below, this model has the highest sensitivity and specificity values of 0.80 and 0.89, respectively. The high sensitivity and specificity of the model signify that it can properly distinguish subjects who belong to a certain class as well as those who do not belong to the class.

From those table, it can be seen that Elastic net regression is the second-best model based on the evaluation metrics, with a slightly lower accuracy and AUC score (76 percent and 89 percent, respectively) than multiclass LR.

4.8 Comparison of Resampling Methods

In multiclass LR, we investigated some resampling methods to evaluate their effects on model performance. Two SMOTE functions from two independent packages were used. The difference between SMOTE in the "DMwR" and "smotefamily" packages is that "DMwR" employs a combination of SMOTE and majority class under-sampling, whilst "smotefamily" does not. In using "DMwR," we need to tune the two parameters, *perc_over* and *perc_under* until we achieve an appropriate sample size. We set perc over to 200 and perc under to 300 in this function. We also conducted up-sampling and down-sampling for comparison.

After splitting the dataset into training and validation data, 110 out of 135 data samples were used as training data, including 43 samples of healthy nonsmokers, 48 samples of healthy smokers, and 19 samples of COPD. The "caret" package's function of upSample randomly replicating the data and add those data into the original dataset so that each class has 48 samples, while downSample randomly select the data, so that each class has 19 samples.

The AUC values for multiclass LR of several resampling methods are shown in Table 4.4. The AUCs of SMOTE from "DMwR" and "smotefamily" are quite similar, with only a 0.8 percent difference. Since the results from both packages are insignificantly different, we may use one of the SMOTE functions from those packages. In comparison, using the upSample function to resample the dataset improved the AUC performance by 5%, whilst using the downSample function lowered the performance by 4.3 %. The AUC performance of the upSample function, however, is still lower to that of SMOTE, either by employing "DMwR" or "smotefamily" packages. In all four evaluation metrics, models trained with SMOTE outperformed models trained without it.

Table 4.4: Multiclass LR with different resampling method.

Resampling methods	**AUC of Multi-class LR (%)**
Without resampling	82.4
SMOTE from "DMwR"	90.1
SMOTE from "smotefamily"	89.3
upSample	87.4
downSample	78.1

4.9 Conclusion

We employed a microarray dataset to predict the existence of COPD in this study by first addressing the class imbalance. Previous research on this dataset has predicted the existence of COPD but neglecting the existence of class imbalance. Based on repeated 10-fold cv 10-times, our proposed model can predict the existence of COPD with an overall accuracy and AUC score of 80% and 90%, respectively. The results show that by addressing class imbalance in the pre-processing task can more correctly predict the existence of COPD.

Our proposed methods have higher sensitivity and specificity than those that do not take class imbalance into consideration in the pre-processing task. It indicates that the chosen model can accurately identify sample that belongs to a certain class as well as sample that does not belong to that class. The proposed technique in this study can be utilized to assist in the design of improved therapies to reduce COPD-related morbidity and mortality. To improve the performance, in the future study, we are considering using more recent and sophisticated resampling approaches.

Chapter 5

Classification of Imbalanced Data Represented as Binary Features

5.1 Introduction

Understanding the properties of input data and selecting the methods that are most suited for obtaining high performance in the machine learning job are critical in the field of machine learning (regression, classification, clustering etc.). Class imbalance is a typical problem that arises from real-world data. This occurs when the number of samples represented in one class is less than the number of samples represented in the other(s) class(es), as in a rare disease dataset. Typically, the minority class is more important and is designated as the target class.

As a result, a dataset with class imbalance is more vulnerable to missclassification in the minority class than in the majority class. This problem makes it difficult for classifiers to properly predict the minority class since they are more likely to predict minority class data as majority class data. To handle the problem of class imbalance, a number of methods have been developed. At the data level method, resampling is a frequently used solution for this problem. Undersampling the majority class or oversampling the minority class are examples of the method. Many existing method, however, were designed explicitly to overcome the problem in numerical data. Some kinds of data, on the other hand, are represented as binary values that can only have one of two values (0/1, T/F, M/F, etc.). A binary feature like this sits at the border of numerical and categorical values. It can be considered as a numerical value, but the domain of value is limited, and there is no difference between itself and categorical value. Direct application of oversampling methods to a dataset with binary features is not a good idea since such techniques rely

on numerically represented values for synthesizing new minority samples.

In this study, we studied on utilizing feature extraction to convert binary features into numerical features before oversampling. Classification performance was evaluated using a variety of datasets, classifiers, feature extraction approaches, and oversampling techniques. The effectiveness of the approach was validated through extensive experiments.

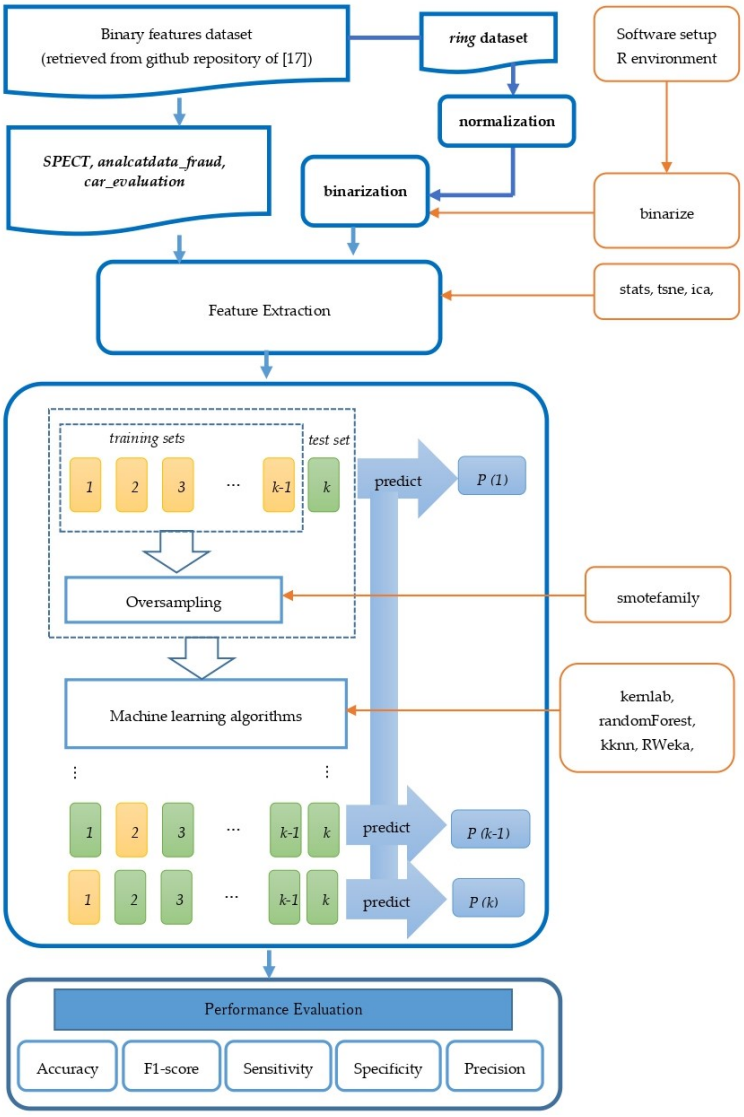

Figure 5.1: The flowchart of the proposed approach.

In this study, we investigated an approach of using feature extraction for converting binary feature into numerical feature prior oversampling. The flowchart of our approach is shown in Figure 5.1. Using various datasets from different domains and imbalance ratios, classifiers, feature extraction methods, and oversampling methods, classification performances were measured. Through comprehensive experiments, the effectiveness of

the approach was confirmed.

5.2 Material and Methods

All of the datasets used in this study are binary features, which means that all of the data input is binary (values of target class are also represented as binary, but they are treated as categorical values). SPECT, analcatdata fraud, and ring are three benchmark datasets downloaded from Github repository [1] . SPECT and analcatdata_fraud are datasets which all the input data are binary. The ring dataset's input data are originally numeric with two class label (binary) which then were normalized and binarized using the k-means clustering algorithm of the "Binarize" R packages. It was done because of the lack of available benchmark datasets of imbalanced class especially for binary features. The ring_1500vs3000, ring_100vs500, ring_100vs2000, and ring_60vs3000 are randomly drawn from the ring dataset. The same procedure was also done by [28] in order to evaluate their models on different degrees of imbalance due to the lack of available datasets.

The remaining three datasets are biological data from Kanazawa University Hospital. Two MRSA datasets have practically identical feature data, which represents the presence of mutations in each MRSA strain associated to each sample [29]. The target class is the main difference between two MRSA datasets. In MRSA pathogenicit0y dataset, 0 and 1 represent latent and developed, respectively. In MRSA drug resistance dataset, target class represent the resistance of each strain to Clindamycin (CLDM). Similarly, C. difficile pathogenicity dataset contains feature data representing the exist-ence of mutations in each strain of C. difficile, which causes diarrhea in human and has difficulty in antibiotic treatment. To generate the feature data, whole genome of 77 strains were sequenced by Hiseq 2500 with 150 base reads. The reads were mapped to the refer-ence genome NC_009089 in RefSeq (the same as AM180355.1 in GenBank) by BWA. After the mapping and file conversion by SAMtools, mutations were detected by Varscan. Between Indels and SNPs type of mutation, Only Indels (insertions and deletions) were used in this study. If the feature data (presence of mutation) of two or more Indels were completely the same in 77 strains after Indel identification , the features were combined as one because redundant features might have a negative impact on machine learning classification performance. Finally, 77 samples are used along with 2610 generated features, where each of the feature correspond to one or more mutations.

The basic information regarding the number of samples and other characteristics of the datasets are summarized in Table 5.1. All the input variables of these datasets are

Table 5.1: Multiclass LR with different resampling method.

Dataset	**# of features**	**# of samples**	**Target class ratio (0:1)**
SPECT	22	267	55:212
analcatdata_fraud	10	42	29:13
car_evaluation	22	1728	1210:384:65:69
MRSA pathogenicity	1978	96	33:63
MRSA drug resistance	1976	94	75:19
C. difficile pathogenicity	2610	77	46:31
ring_1500vs3000	21	4500	1500:3000
ring_100vs500	21	2100	100:500
ring_100vs2000	21	2100	100:2000
ring_60vs3000	21	3060	60:3000

binary. All the features of these datasets are included in the analysis.

5.3 Oversampling and Features Extraction Method

In the pre-processing step, at first, the binary features are converted into numerical value using some linear and non-linear features extraction methods. For the linear method, principal component analysis (PCA) and independent component analysis (ICA) were performed. What distinguished ICA from PCA is that PCA assumes that signals are subject to multi-variate Gaussian distribution and uses orthogonal bases to decompose signals.

For the non-linear methods, t-distributed stochastic neighbor embedding (tSNE) and uniform manifold approximation and projection (UMAP) were performed. tSNE technique aims at identifying the relevant pattern of a dataset while maintain its lo-cal structure. For each point of a datasets, tSNE models the probability distribution of other points which are closest to it. UMAP is developed based on ideas from topological data analysis and manifold learning tech-niques, which assumes that the data is uniformly distributed on the manifold. Compared to tSNE, UMAP provides much faster computational running time.

After the datasets are converted into numerical data using those features extraction methods, those datasets then were split into test and training data by 10-fold cross validation. Then, the training datasets were oversampled using various methods such as SMOTE, ADASYN, borderline-SMOTE, adaptive neighbor SMOTE, safe-level SMOTE,

relocating-safe-level SMOTE, and DBSMOTE. This resampled dataset were used in the next step of machine learning.

5.4 Classification Methods

In this study, we employ various well-developed classification methods such as k-nearest neighbour (k-nn), decision tree C4.5, random forest, and support vector machine (SVM). According to [30], most research efforts on the class imbalance issue focus on decision tree C4.5 although the existence of a class imbalance makes this classifier a needs many splits to distinguish the minority class. As the ensemble of decision trees, the capability of random forest in imbalance class classification also been studied. Other studies reported that SVMs are less affected by the class imbalance issue [5]. The strength of SVM's method such as the ability to combine with kernel-based learning, allows SVM for a more flexible analysis and optimal solution. We have provided a brief introduction to each mentioned classifier's learning techniques and generated an insight into the inadequacy of each learning method in the Chapter 2 to be comprehended by less knowledgeable readers of these learning algorithms.

In general, while building a classification model, a learning algorithm reveals the underlying relationships between the features and the target class, and then finds the best model that suit the training data. The classifier's goal is to predict the class labels for any input data. As a result, the learning goal is to develop a classification model that is capable of predicts the class labels of previously unseen data with high accuracy.

However, we can't fit and analyze machine learning algorithms on raw data; instead, we have to modify the data to match the requirements of the selected algorithms. Furthermore, in order to obtain the highest performance, we need to pick a data representation that optimally exposes the unknown underlying structure of the prediction issue to the learning algorithms. To achieve the goal, pre-processing task is essential task in this whole process. Regarding the issue of the binary features data mentioned in the previous section, in this study, datasets are pre-processed using oversampling and features extraction methods. After the pre-processing tasks are completed, the datasets are ready to be feed into the machine learning models. In this process, the combination of oversampling methods, features extraction methods, and classification methods are analyzed. If the best combinations are selected, the parameters of the methods and model were tuned to achieve the highest performances.

5.5 Evaluation Metrics

Evaluation metrics play an essential role in both evaluating the classifier's performance and guiding the classifier in the next process. Standard metrics for assessing classification prediction models, such as classification accuracy and classification error, are commonly used. Standard metrics are commonly used because they perform effectively for most tasks. All metrics, however, involve assumptions about the problem or what is important in the problem. As a result,the project stakeholders need to choose an appropriate evaluation metrics that best represents what they feel is essential about the model or forecasts, which makes selecting model evaluation metrics problematic.

When there is a skew in the class distribution, this issue becomes considerably more difficult. The reason behind it is that many standard metrics become inaccurate when facing high skewness in the class imbalance data such as with the ratio of 1:100 or 1:1000 ratio between a minority and majority class. It can drive to misleading result if we report classification accuracy for a highly imbalance data classification. So, using different evaluation metrics for imbalance data classification are highly advised besides also resampling the dataset at first in the pre-processing task [31].

In this study, some evaluation metrics from threshold family such as accuracy, sensitivity, specificity, precision, and F1-score are used. All the mentioned metrics except accuracy are less prone to class imbalance problem, because those metrics evaluate the performance separately in each class of instances. As a result, the class imbalance does not affect these metrics. In addition, we also applied 10-fold cross validation to evaluate the metrics using original dataset (orig), SMOTE, ADASYN, borderline-SMOTE, adaptive neighbor SMOTE, safe-level SMOTE, relocating-safe-level SMOTE, and DBSMOTE.

5.6 Result and Discussion

Six datasets, four classifiers, five feature extraction techniques, including "no feature extraction," and eight oversampling methods, including "no over-sampling," were combined and evaluated. Some combinations were excluded from the experiments due to data and software limitations. For example, for the C. difficile pathogenicity dataset, oversampling methods of ICA, BLS, and ANS were not evaluated.

Table 5.2 shows the results of 10 datasets used in this study based on the evaluation metrics of accuracy, sensitivity, specificity, precision, and F1-score. The results of base model compared to the best combination of feature extraction and oversampling method were observed. The values are the average of 3- or 5- or 10-fold cross-validation on the

Table 5.2: Performance metrics of the best combination compared to base model.

Model	Accuracy	Sensitivity	Specificity	Precision	F1-score
SPECT					
RF base model	0.816896	0.901468	0.490909	0.872211	0.886598
RF+tSNE+RSLS	0.982938	0.99109	0.951515	0.987467	0.989275
	(+0.1660)	(+0.0896)	(+0.4606)	(+0.1153)	(+0.1027)
analcatdata_fraud					
RF base model	0.690476	0.247863	0.888889	0.5	0.331429
RF+umap+RSLS	0.97619	0.940171	0.992337	0.982143	0.960699
	(+0.2857)	(+0.6923)	(+0.1035)	(+0.4821)	(+0.629)
car_evaluation					
RF base model	0.954138	0.84172	0.983518	0.895227	0.867647
RF+PCA+RSLS	0.995804	0.988627	0.998363	0.988205	0.988389
	(+0.0417)	(+0.1469)	(+0.0149)	(+0.0930)	(+0.1207)
MRSA pathogenicity					
RF base model	0.617187	0.873016	0.128788	0.656716	0.749574
RF+ica+RSLS	0.979167	0.988095	0.962121	0.980315	0.98419
	(+0.3620)	(+0.1151)	(+0.8333)	(+0.3236)	(+0.2346)
MRSA drug resistance					
RF base model	0.906915	0.644737	0.973333	0.859649	0.736842
RF+tSNE+RSLS	0.99734	0.986842	1.000000	1.000000	0.993377
	(+0.0904)	(+0.3421)	(+0.0267)	(+0.1404)	(+0.2565)
C. diffcl. pathogenicity					
RF base model	0.750361	0.727599	0.7657	0.676667	0.701209
RF+umap+RSLS	0.984127	0.982079	0.985507	0.982014	0.980322
	(+0.2338)	(+0.2546)	(+0.2198)	(+0.3054)	(+0.2791)
ring_1500vs3000					
RF base model	0.75025	0.915357	0.365	0.770827	0.836898
RF+PCA+RSLS	0.9717	0.976857	0.959667	0.982612	0.979726
	(+0.2215)	(+0.0615)	(+0.5947)	(+0.2118)	(+0.1428)
ring_100vs500					
RF base model	0.849167	0.9925	0.1325	0.851201	0.916436
RF+PCA+RSLS	0.985	0.9945	0.9375	0.987587	0.991031
	(+0.1358)	(+0.002)	(+0.805)	(+0.1364)	(+0.0746)
ring_100vs2000					
RF base model	0.951905	0.9995	0	0.952358	0.97536
RF+PCA+RSLS	0.993413	0.997833	0.905	0.995262	0.996546
	(+0.0415)	(-0.0017)	(+0.9050)	(+0.0429)	(+0.0212)
ring_60vs3000					
RF base model	0.972222	1. 00000	0	0.972222	0.985915
RF+PCA+RSLS	0.996042	0.998786	0.9	0.997148	0.997966
	(+0.0238)	(-0.0012)	(+0.9)	(+0.0249)	(+0.0121)

datasets.

The best combination is selected based on the highest value of the sum of accuracy and F1-score. As can be shown in the Table 3, it can be observed that most of the metrics outperformed the base model without feature extraction and oversampling methods as the preprocessing task. It strongly confirmed our hypothesis that the combination of feature extraction and oversampling method can improve the performances of classifiers. This approach can be applied on the imbalanced binary features dataset with different ratio of imbalance from low to high (33:63 to 60:3000). In our results, the best combination of all datasets is obtained by random forest classi-fier. Random forest is said to be a classifier that prone to class imbalance, but when it is combined with feature extraction methods and oversampling techniques, it outperformed the other classifiers with the same combination. This result was also confirmed by [4] which reported the same result that the inclusion of data sampling improves the classification performance of random forest classifier. In the preprocessing task, RSLS is selected as the best oversampling method for all the datasets, while the best feature extraction method varies in each of the datasets.

The experimental results supported our hypothesis that "converting binary values of features into numerical values might increase over-sampling performance." The values in the rightmost column “MAX – (no feature extraction)” in most of the table rows in S1 and S2 were larger than zero. It indicates that a feature extraction approach tends to increase the initial performance of an oversampling method. For example, the combination of SPECT, random forest (RF), no feature extraction, and RSLS has an accuracy of 0.8169. It was significantly improved to 0.9829 using a feature extraction (tSNE) (in other words, MAX – (no feature extraction) = 0.064). At this point, it’s reasonable to conclude that combining oversampling with feature extraction will give great results. It should be noted that F1 scores also improved similarly. Furthermore, in many cases, the similar combination of feature extraction and oversampling methods produced the highest results in terms of accuracy and F1 score. Because accuracy and F1 score usually show trade-offs for imbalanced data, this finding suggests that the technique used in this work can help improve the performance of a wide range of binary feature datasets. About the applicability, it is also noticeable that this approach was effective for various ra-tios of imbalance (from 557:567 to 19:75) and various ratio between features and samples (from 10:1124 to 2610:77).

The results show that the combination of oversampling and features extraction method sinergistically improved the performance. For example, the accuracies of the combinations (SPECT, RF, no feature extraction, no oversampling) and (SPECT, RF, no feature

Table 5.3: Summary of accuracy improvements about SPECT dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.0062	0.0050	0.0050	-0.0013
SMOTE	-0.0125	0.0108	-0.0008	-0.0109
ADASYN	0.0221	0.0304	-0.0034	0.0025
SLS	-0.0004	-0.0058	0.0067	0.0008
BLS	-0.0154	0.0016	0.0050	0.0163
ANS	-0.0133	0.0200	-0.0063	-0.0033
RSLS	0.0824	0.0220	0.0640	0.0133
DBS	0.0079	-0.0096	-0.0042	0.0071

Table 5.4: Summary of accuracy improvements about analcatdata_fraud dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.0741	0.0370	0.0661	0.0688
SMOTE	0.0291	0.0582	0.0052	0.0634
ADASYN	0.0397	0.0609	-0.0080	0.0185
SLS	0.0450	0.0318	0.0476	-0.0106
RSLS	0.1031	0.0185	0.0291	-0.0132
DBS	0.0159	0.0900	0.0185	0.0556

extraction, tSNE) are so different (0.8169 and 0.8170, respectively). It indicates that applying tSNE on the original data without oversampling has no effect on performance. Despite this, when employed as a preprocessing technique prior to the RSLS oversampling approach, it significantly improved accuracy (as described above). Furthermore, we can see that in many cases, using a feature extraction technique alone degraded performance, while using it in combination with an oversampling method improved the performance. The application of tSNE, for example, reduced the F1 score of the combination (SPECT, C4.5, no over-sampling) from 0.8810 to 0.8512. In contrast, the F1 score of the combination (SPECT, C4.5, RSLS) increased from 0.9003 to 0.9528 by tSNE. Finally, Tables 2-11 summarize the achieved improvements (i.e. "MAX – (no feature extraction)") in S1 and S2. In these summary tables, it can be seen that most of the tables are filled by positive values indicating the improvement by feature extraction.

Table 5.5: Summary of accuracy improvements about MRSA pathogenicity dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.1120	0.0937	0.0312	0.0416
SMOTE	0.0651	0.1640	-0.0052	0.0599
ADASYN	0.0625	0.1380	-0.0052	0.0286
SLS	0.1068	0.1771	0.0182	0.0599
BLS	0.0678	0.1718	0.0078	0.0520
ANS	0.0443	0.1458	0.0078	0.0443
RSLS	0.0417	0.3307	0.0104	0.0469
DBS	0.0547	0.1588	-0.0026	0.0599

Table 5.6: Summary of accuracy improvements about C. difficile pathogenicity dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.0389	0.0144	0.0346	0.0028
SMOTE	0.0159	0.0043	0.0159	0.0245
ADASYN	0.0332	0.0173	-0.0044	0.0346
SLS	0.0418	0.0216	0.0245	0.0188
RSLS	0.1356	0.0375	0.1356	0.0144
DBS	0.0822	0.0274	0.0476	0.0274

Table 5.7: Summary of F1 score improvements about SPECT dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.0059	0.0053	0.0054	0.0030
SMOTE	-0.0114	0.0079	0.0003	-0.0040
ADASYN	0.0149	0.0231	-0.0012	0.0054
SLS	-0.0023	-0.0030	0.0066	0.0015
BLS	-0.0116	0.0047	0.0053	0.0128
ANS	-0.0144	0.0152	-0.0039	-0.0013
RSLS	0.0525	0.0134	0.0404	0.0087
DBS	0.0043	-0.0088	-0.0023	0.0048

Table 5.8: Summary of F1 score improvements about analcatdata_fraud dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.2359	0.2142	0.1815	NaN
SMOTE	0.0493	0.1082	0.1032	0.1560
ADASYN	0.0527	0.1066	0.0835	0.0856
SLS	0.0896	0.1127	0.1551	0.0513
RSLS	0.1186	0.0241	0.0486	-0.0270
DBS	0.0685	0.1830	0.0956	0.1677

Table 5.9: Summary of F1 score improvements about MRSA pathogenicity dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.1308	0.0619	0.0354	0.0351
SMOTE	0.0536	0.3240	0.0043	0.0558
ADASYN	0.0589	0.2979	0.0051	0.0363
SLS	0.0927	0.3444	0.0009	0.0378
BLS	0.0553	0.3279	0.0039	0.0242
ANS	0.0314	0.3261	0.0006	0.0445
RSLS	0.0297	0.4217	0.0078	0.0372
DBS	0.0596	0.3298	0.0063	0.0327

Table 5.10: Summary of F1 score improvements about MRSA drug resistance dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	0.1085	0.0489	0.0526	0.1310
SMOTE	0.0890	0.1998	0.0094	0.1241
ADASYN	-0.0179	0.2772	0.0053	0.0886
SLS	0.0568	0.1615	0.0555	0.1304
RSLS	0.0381	0.1291	0.0199	0.0592
DBS	0.0036	0.1280	0.0196	0.0735

Table 5.11: Summary of F1 score improvements about C. difficile pathogenicity dataset.

Oversampling method	**C4.5**	**KNN**	**RF**	**SVM**
no oversampling	-0.1232	-0.073	-0.0919	-0.1225
SMOTE	0.0037	0.064	0.0098	0.0773
ADASYN	0.0663	0.0191	0.0261	0.0776
SLS	0.0597	0.0192	0.032	0.0059
RSLS	0.2877	0.0905	0.2423	0.0648
DBS	0.0456	-0.0063	0.0172	-0.0165

5.7 Conclusion

An approach to utilize feature extraction methods as a preprocessing prior to oversampling was proposed, focusing on the problem of binary features that are too poor to apply oversampling algorithms like SMOTE. It was proven that this approach works well in many cases through extensive tests using various datasets and methods. Converting binary features into numerical ones using feature extraction methods leads to a better results for oversampling methods. Furthermore, it has been proven that feature extraction and oversampling sinergistically increase classification performance.

Conclusion

We can outlined some conclusion of this study as follows:

1. Some oversampling methods can improve the performance metrics.
2. Converting binary features into numerical ones using feature extraction methods leads to a better results for oversampling methods.
3. Combination of oversampling and feature extractions methods synergistically improve the performance metrics better than sole method.
4. Accuracy and F1 score frequently show a trade-off relationship for imbalanced data, this result indicates that the approach in this study can contribute to the performance improvement of a wide variety of binary feature datasets.
5. It is also noticeable that this approach was effective for various ratio of imbalance (from 557:567 to 19:75) and various ratio between features and samples (from 10:1124 to 2610:77).
6. Furthermore, we can see many cases that simple application of a feature extraction method decreased the performance but the combined use of it with an oversampling method improved it.

Bibliography

[1] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, J. H. Moore, Pmlb: a large benchmark suite for machine learning evaluation and comparison, BioData mining 10 (1) (2017) 1–13.

[2] Y. B. Wah, H. A. A. Rahman, H. He, A. Bulgiba, Handling imbalanced dataset using svm and k-nn approach, in: AIP Conference Proceedings, Vol. 1750, AIP Publishing LLC, 2016, p. 020023.

[3] C. Lemnaru, R. Potolea, Imbalanced classification problems: systematic study, issues and best practices, in: International Conference on Enterprise Information Systems, Springer, 2011, pp. 35–50.

[4] D. J. Dittman, T. M. Khoshgoftaar, A. Napolitano, The effect of data sampling when using random forest on imbalanced bioinformatics data, in: 2015 IEEE international conference on information reuse and integration, IEEE, 2015, pp. 457–463.

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

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

[7] H. He, Y. Bai, E. A. Garcia, S. Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE, 2008, pp. 1322–1328.

[8] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, in: International conference on intelligent computing, Springer, 2005, pp. 878–887.

[9] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Pacific-Asia conference on knowledge discovery and data mining, Springer, 2009, pp. 475–482.

[10] W. Siriseriwan, K. Sinapiromsaran, The effective redistribution for imbalance dataset: Relocating safe-level smote with minority outcast handling, Chiang Mai Journal of Science 43 (1) (2016) 234–246.

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: Kdd, Vol. 96, 1996, pp. 226–231.

[12] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Dbsmote: density-based synthetic minority over-sampling technique, Applied Intelligence 36 (3) (2012) 664–684.

[13] W. Siriseriwan, K. Sinapiromsaran, Adaptive neighbor synthetic minority oversampling technique under 1nn outcast handling, Songklanakarin J. Sci. Technol 39 (5) (2017) 565–576.

[14] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, A. Hooman, An overview of principal component analysis, Journal of Signal and Information Processing 4 (3B) (2013) 173.

[15] S. D. You, M.-J. Hung, Reducing dimensionality of spectro-temporal data by independent component analysis, in: 2020 2nd International Conference on Computer Communication and the Internet (ICCCI), IEEE, 2020, pp. 93–97.

[16] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (11) (2008).

[17] L. McInnes, J. Healy, J. Melville, Umap: Uniform manifold approximation and projection for dimension reduction, arXiv preprint arXiv:1802.03426 (2018).

[18] Y. Sekine, H. Katsura, E. Koh, K. Hiroshima, T. Fujisawa, Early detection of copd is important for lung cancer surveillance, European Respiratory Journal 39 (5) (2012) 1230–1240.

[19] J. L. López-Campos, W. Tan, J. B. Soriano, Global burden of copd, Respirology 21 (1) (2016) 14–23.

[20] S. Anakal, P. Sandhya, Clinical decision support system for chronic obstructive pulmonary disease using machine learning techniques, in: 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), IEEE, 2017, pp. 1–5.

[21] Y. Yao, Y. Gu, M. Yang, D. Cao, F. Wu, The gene expression biomarkers for chronic obstructive pulmonary disease and interstitial lung disease, Frontiers in genetics 10 (2019) 1154.

[22] X. Qian, Y. Ba, Q. Zhuang, G. Zhong, Rna-seq technology and its application in fish transcriptomics, Omics: a journal of integrative biology 18 (2) (2014) 98–110.

[23] ThermoFisher, Genechiptm human genome u133 plus 2.0 array., https://www.thermofisher.com/order/catalog/product/900468 (2001).

[24] L. Trtica-Majnaric, M. Zekic-Susac, N. Sarlija, B. Vitale, Prediction of influenza vaccination outcome by neural networks and logistic regression, Journal of biomedical informatics 43 (5) (2010) 774–781.

[25] A. Tharwat, Classification assessment methods, Applied Computing and Informatics (2020).

[26] D. J. Hand, R. J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, Machine learning 45 (2) (2001) 171–186.

[27] P. Filzmoser, B. Liebmann, K. Varmuza, Repeated double cross validation, Journal of Chemometrics: A Journal of the Chemometrics Society 23 (4) (2009) 160–171.

[28] L. Zhou, K. K. Lai, Benchmarking binary classification models on data sets with different degrees of imbalance, Frontiers of Computer Science in China 3 (2) (2009) 205–216.

[29] B. Abapihi, M. R. Faisal, N. G. Nguyen, M. K. Delimayanti, B. Purnama, F. R. Lumbanraja, D. Phan, M. Kubo, K. Satou, Cross entropy based sparse logistic regression to identify phenotype-related mutations in methicillin-resistant staphylococcus aureus, Journal of Biomedical Science and Engineering 13 (7) (2020) 168–174.

[30] Y. Sun, A. K. Wong, M. S. Kamel, Classification of imbalanced data: A review, International journal of pattern recognition and artificial intelligence 23 (04) (2009) 687–719.

[31] N. Japkowicz, Assessment metrics for imbalanced learning, Imbalanced learning: Foundations, algorithms, and applications (2013) 187–206.

Bibliography

[1] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, J. H. Moore, Pmlb: a large benchmark suite for machine learning evaluation and comparison, BioData mining 10 (1) (2017) 1–13.

[2] Y. B. Wah, H. A. A. Rahman, H. He, A. Bulgiba, Handling imbalanced dataset using svm and k-nn approach, in: AIP Conference Proceedings, Vol. 1750, AIP Publishing LLC, 2016, p. 020023.

[3] C. Lemnaru, R. Potolea, Imbalanced classification problems: systematic study, issues and best practices, in: International Conference on Enterprise Information Systems, Springer, 2011, pp. 35–50.

[4] D. J. Dittman, T. M. Khoshgoftaar, A. Napolitano, The effect of data sampling when using random forest on imbalanced bioinformatics data, in: 2015 IEEE international conference on information reuse and integration, IEEE, 2015, pp. 457–463.

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

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

[7] H. He, Y. Bai, E. A. Garcia, S. Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE, 2008, pp. 1322–1328.

[8] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, in: International conference on intelligent computing, Springer, 2005, pp. 878–887.

[9] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Pacific-Asia conference on knowledge discovery and data mining, Springer, 2009, pp. 475–482.

[10] W. Siriseriwan, K. Sinapiromsaran, The effective redistribution for imbalance dataset: Relocating safe-level smote with minority outcast handling, Chiang Mai Journal of Science 43 (1) (2016) 234–246.

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: Kdd, Vol. 96, 1996, pp. 226–231.

[12] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Dbsmote: density-based synthetic minority over-sampling technique, Applied Intelligence 36 (3) (2012) 664–684.

[13] W. Siriseriwan, K. Sinapiromsaran, Adaptive neighbor synthetic minority oversampling technique under 1nn outcast handling, Songklanakarin J. Sci. Technol 39 (5) (2017) 565–576.

[14] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, A. Hooman, An overview of principal component analysis, Journal of Signal and Information Processing 4 (3B) (2013) 173.

[15] S. D. You, M.-J. Hung, Reducing dimensionality of spectro-temporal data by independent component analysis, in: 2020 2nd International Conference on Computer Communication and the Internet (ICCCI), IEEE, 2020, pp. 93–97.

[16] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (11) (2008).

[17] L. McInnes, J. Healy, J. Melville, Umap: Uniform manifold approximation and projection for dimension reduction, arXiv preprint arXiv:1802.03426 (2018).

[18] Y. Sekine, H. Katsura, E. Koh, K. Hiroshima, T. Fujisawa, Early detection of copd is important for lung cancer surveillance, European Respiratory Journal 39 (5) (2012) 1230–1240.

[19] J. L. López-Campos, W. Tan, J. B. Soriano, Global burden of copd, Respirology 21 (1) (2016) 14–23.

[20] S. Anakal, P. Sandhya, Clinical decision support system for chronic obstructive pulmonary disease using machine learning techniques, in: 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), IEEE, 2017, pp. 1–5.

[21] Y. Yao, Y. Gu, M. Yang, D. Cao, F. Wu, The gene expression biomarkers for chronic obstructive pulmonary disease and interstitial lung disease, Frontiers in genetics 10 (2019) 1154.

[22] X. Qian, Y. Ba, Q. Zhuang, G. Zhong, Rna-seq technology and its application in fish transcriptomics, Omics: a journal of integrative biology 18 (2) (2014) 98–110.

[23] ThermoFisher, Genechiptm human genome u133 plus 2.0 array., https://www.thermofisher.com/order/catalog/product/900468 (2001).

[24] L. Trtica-Majnaric, M. Zekic-Susac, N. Sarlija, B. Vitale, Prediction of influenza vaccination outcome by neural networks and logistic regression, Journal of biomedical informatics 43 (5) (2010) 774–781.

[25] A. Tharwat, Classification assessment methods, Applied Computing and Informatics (2020).

[26] D. J. Hand, R. J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, Machine learning 45 (2) (2001) 171–186.

[27] P. Filzmoser, B. Liebmann, K. Varmuza, Repeated double cross validation, Journal of Chemometrics: A Journal of the Chemometrics Society 23 (4) (2009) 160–171.

[28] L. Zhou, K. K. Lai, Benchmarking binary classification models on data sets with different degrees of imbalance, Frontiers of Computer Science in China 3 (2) (2009) 205–216.

[29] B. Abapihi, M. R. Faisal, N. G. Nguyen, M. K. Delimayanti, B. Purnama, F. R. Lumbanraja, D. Phan, M. Kubo, K. Satou, Cross entropy based sparse logistic regression to identify phenotype-related mutations in methicillin-resistant staphylococcus aureus, Journal of Biomedical Science and Engineering 13 (7) (2020) 168–174.

[30] Y. Sun, A. K. Wong, M. S. Kamel, Classification of imbalanced data: A review, International journal of pattern recognition and artificial intelligence 23 (04) (2009) 687–719.

[31] N. Japkowicz, Assessment metrics for imbalanced learning, Imbalanced learning: Foundations, algorithms, and applications (2013) 187–206.