UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Stacking Ensemble Model for Liver Stiffness Classification with Imbalanced Data

by

Mingjian Wang

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Master of Biomedical Engineering by Research

Sydney, Australia

February 2021

Certificate of Authorship/Originality

I, Mingjian Wang declare that this thesis, is submitted in fulfilment of the requirements for the award of Master of Biomedical Engineering by Research, in the Faculty of Engineering and IT at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:
Signature: Signature removed prior to publication.

Date: 18/02/2021

ABSTRACT

Stacking Ensemble Model for Liver Stiffness Classification with Imbalanced Data

by

Mingjian Wang

Liver cirrhosis is a significant threat to humans; once the liver reaches the last stage of cirrhosis, there is no cure for it. Therefore, discovering cirrhosis in the early stage is one of the effective ways to decrease the mortality rate of cirrhosis. Besides early detection, increasing the correct cirrhosis diagnosis rate is another desirable method to avoid late treatment for patients. This thesis developed an automatic diagnosis approach to predict doctors' opinions for patients regarding the liver stiffness measurements from FibroScan tests.]A model using the Stacking ensemble method was presented to build a classifier for an imbalanced liver stiffness measurement data-set. The data-set was collected from 13,418 Chinese patients who had liver cirrhosis tests by FibroScan. It recorded 30 sets of features, also provided professional doctors' opinions in Chinese. To transfer the Chinese characters to digital, we applied Jieba module in Python which is a natural language processing method to create 6 labels in classification. Each label presents one doctors' opinion. Since this data-set is highly imbalanced, sampling methods such as the under-sampling method and the oversampling method are applied to solve this problem. To identify the most suitable model for the classification, we performed a study of 7 supervised learning algorithms, Logistic Regression (LR), Decision Tree (DT), Naive Bayesian (NB), K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Random Forest (RF) and AdaBoost; also demonstrated the stacking models based on these supervised learning algorithms. The results demonstrated that the use of Synthetic Minority Oversampling Technique (SMOTE) oversampling technique was effective to handle the imbalanced liver data-set, and the best fitting model was constructed by using DT as meta-classifier with four base classifiers (KNN, RF, DT, SVM) in the stacking model.

Dissertation directed by Professor Steven Su

School of Biomedical Engineering

Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisor Steven Su for my postgraduate study. It was a great opportunity for me to work with this wonderful supervisor. I have enhanced my abilities under his helpful guidance, endless patience, consideration and kindness. Also, I would like to thank my co-supervisor Steve Ling for his helpful advice for part of my project.

Many thanks to my labmates at Faculty of Engineering and Information Technology (FEIT), for their support, giving useful recommendations and comments for my project any time I asked for help during these two years of study.

Finally, I would like to thank my parents for their encouragement. I could not achieve this study without their support.

Mingjian Wang

Sydney, Australia, 2021.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cirrhosis is one of the most alarming complications of liver diseases, having the ability to damage liver cells. Cirrhosis results from various liver injury mechanisms leading to necroinflammation and fibrogenesis (Tsochatzis et al. 2014). Liver cirrhosis means a shrunk, scarred, and hardened liver which is potentially damaging to the liver's function. This results in permanent (long-term) hepatic damage caused by multiple factors, which results in gradual scarring of the liver (Tsochatzis et al. 2014). Cirrhosis has been considered an end-stage condition leading to death inevitably unless a liver transplant is performed. Global cirrhosis prevalence varies between 4.5% and 9.5% of the general population (Sarin & Mainwall 2020). The main risk factors of liver cirrhosis are chronic hepatitis B, chronic hepatitis C, Chronic excessive alcohol intake, and fatty liver disease (Liu et al. 2019). This chapter will briefly introduce the background of liver cirrhosis and show the problem statement of this study, including the problem of imbalanced data and the choice of algorithms. Also, the organization of this thesis will be presented.

1.1 Background

Liver cirrhosis is a significant concern to humans, because it is a growing matter of morbidity and one of the top rank leading causes of mortality worldwide. It was estimated by the World Health Organization that liver cirrhosis causes 1.03 million deaths annually worldwide (Tsochatzis et al. 2014), including 257 million people were suffering from chronic hepatitis B virus (HBV) infection, which causes 887,000 deaths worldwide annually (Liu et al. 2019). Liu et al. (2019) also indicated that around 70 million people were infected with HBV in China, which had the largest population (1.39 billion people in 2017). According

to the data collected by the British Liver Trust (2018), over 4000 people in the UK pass away from cirrhosis and around 700 people must have surgery to transplant liver each year to survive. Cirrhosis is a type of liver damage due to the loss of healthy liver cells and scar tissue, and there is a higher cause percentage particularly in those who have excessive drinking of alcohol, viral hepatitis B and C, and fatty liver caused by obesity and diabetes. Better Health Channel (2018) stated that cirrhosis can influence the liver, making it unable to perform its vital functions of metabolism, to produce blood clotting factors, and filter drugs and toxins. Once the situation reaches the last stage of cirrhosis, there is no cure. Therefore, the symptoms of cirrhosis liver disease should be diagnosed in advance.

Recently most hospitals are using a test device to collect and analyse the health condition of patients' liver. FibroScan is a non-invasive device to evaluate liver stiffness based on the technique of transient elastography (TE). Non-invasive tests are recently well established in the management of patients with chronic liver disease, and the tests can reflect the severity of liver disease by stages (William 2013). FibroScan is very sensitive influenced by multiple factors, such as obesity (BMI> 30-35 kg/m^2), older age, and presence of ascites. To further improve the reliability of the FibroScan results, second opinions from doctors are needed. Doctors' opinions for patients are crucial to identify the accuracy of the results from FibroScan. A professional second opinion frequently provides a higher level of reliability to patients so that to have an accurate treatment for the disease in the future, avoiding any delay to prevent deterioration risk. No or few research that indicates a method to increase the reliability of providing accurate opinions for FibroScan results.

Misdiagnosis and missed diagnosis of liver cirrhosis by doctors are common events in the world. A study estimated that the rate of misdiagnosis and missed diagnosis was more than 20%, while 66% of patients required some changes to the initial diagnoses (Such et al. 2017). It means 2 out of 10 patients would be given wrong diagnosis no matter what illness they have. Another study resulted that up to 50% of patients in the early stage of cirrhosis were missed diagnosed or misdiagnosed, which could lead to late medical

treatments causing a worse situation (Guss, Sherigar & Mohanty 2018). In the case of liver cirrhosis diagnosis, besides the FibroScan results providing basic data of liver condition, doctor's opinions take the most important role in the final decision. If a doctor can combine the previous similar cases with the FibroScan results to give a comprehensive opinion to patients, it is more convincing for doctors to give proper treatment for specific illnesses, particularly for inexperienced doctors. A system that can predict second opinions of liver stiffness measurement in natural languages is desirable to be established for hospital use.

Natural language processing (NLP) is an ideal bridge to build effective communication between humans and computers. Different languages are unable to communicate, for example, dogs barking is incomprehensible for humans. Even humans with different languages need translation to communicate. Over the past 20 years, NLP has rapidly grown into practical technologies and scientific research areas (Hirschberg & Manning 2015). There are three popular technologies based on NLP involved in human life: i. Chatting robot Siri is an artificial intelligence assistant software built into Apple's iOS system; ii. Speech recognition is applied in Wechat chatting system and map navigation systems; iii. Google translate can translate up to 104 languages with high accuracy in 2019. Various languages will be processed with different methods. English processing consists of tokenization, stemming, lemmatization, part of speech, named-entity recognition (NER), and chunking (Hirschberg & Manning 2015), but Chinese processing involves Chinese word segmentation, parts of speech, NER and remove stop words. Therefore, NLP is a key function to translate human languages to digital information understanding by a computational prediction system.

Some popular simple techniques have been widely applied to build the prediction systems in biomedical data analysis such as Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbour (KNN), Logistic Regression (LR), and Naive Bayesian (NB). The advantages of most simple algorithms are easy-application, short time consumption, and precise performance in various classification or regression problems. However, each

algorithm has its disadvantages, for example, DT is easy to be overfitting due to less restriction and SVM strongly relies on a kernel function. Enhancing simple algorithms has been continually considered in sustained attention.

In order to improve the advantages of simple algorithms and reduce the influences of the disadvantages, ensemble learning has been proposed to solve this problem. Ensemble learning is an effective technique to improve overall prediction performance consisting of Bagging, Boosting, and Stacking methods. A popular algorithm in the Bagging method is Random Forest proposed in 1995 by Breiman and Cutler. Random Forest (RF) is an algorithm based on DT algorithms and in also combines the Bagging ensemble method with the bootstrap process. RF is not only presenting DT's advantages but also solves the easy-overfitting problem (Pretorius 2016). Similarly, the Stacking ensemble method is another one of the most effective approaches to achieve the improvement of simple algorithms.

Stacking is an ensemble learning method of combining a higher-level meta-classifier and lower-level base classifiers to achieve greater predictive performance. Unlike Bagging and Boosting, Stacking prefers to be built by various learning algorithms to achieve diversity of the base level. A large number of algorithm applications with high diversity at the base level of a stacking model will perform precise classification results (Wang et al. 2011). For over a decade, the Sacking method has contributed to various industries. Syarif et al. (2012) indicated that Stacking was the only method able to decrease the false positive rate by a relatively high amount in the performance of network intrusion detection systems. In biomedical fields, Bhasuran et al. (2016) proposed a stacking ensemble model combined with fuzzy matching for biomedical named entity recognition of diseases that had achieved high-quality performance. Furthermore, Rajaraman et al. (2018) improved TB detection in chest radiographs by a novel stacking model and pre-trained Convolutional Neural Networks (CNNs). According to previous successful cases of Stacking applications, the Stacking method is expected to obtain better results in the prediction of Chinese second opinions of liver stiffness measurements.

1.2 Problem Statement

This thesis discusses two main problems that are features of this project. The first problem is the imbalanced data, due to inequality of class distribution in the data-set. The second problem is the choice of algorithms combination for the Stacking method, which can provide the best performance classifying the data-set.

1.2.1 Imbalanced Data

The imbalanced classification problem is a pattern classification problem in which the number of training samples is unevenly distributed between classes. Learning imbalanced data requires learning useful information from such unevenly distributed data-sets. The imbalance of sample distribution easily leads to the scarcity of rare samples. Specifically, scarcity includes absolute scarcity and relative scarcity.

- **Absolute scarcity** means that the number of training samples of the rare class is absolutely too small, so that this type of information cannot be fully represented by the training samples. The classification error rate of the absolute data-scarce class is much higher than the general class. In addition, when certain types of data are too scarce, it is easy to form small data regions in the feature space, which causes the problem of small disjuncts. Because it is difficult to distinguish small blocks from noise data, there is a high classification error rate in small blocks. Many classifiers perform statistical significance detection in order to prevent over-learning (Huang 2015). For example, in a decision tree, only decision rules and association rules that cover a sufficient number of samples can be retained. While small blocks of data often fail to pass such saliency detection, on the other hand, if the detection threshold is lowered, noise cannot be effectively removed.

- **Relative scarcity** means that the number of samples of the rare class is not too small, but the proportion of the relatively large class is too small. When the total

number of samples is sufficient, the relative scarcity does not necessarily cause the performance of the classifier to decrease. On the contrary, the scarce distribution of rare samples caused by absolute scarcity and the small number will easily cause the performance of the classifier to decrease (Huang 2015).

Therefore, for relatively rare samples, the impact of data imbalance on the performance of the classifier can be reduced by increasing the total number of samples, while absolute scarcity is difficult to solve.

The classification with imbalanced data is also affected by two problems: Noise and Shift (Huang 2015). The existence of noisy data is inevitable and will affect the performance of the classifier to a certain extent. However, for imbalanced classification problems, noisy data will have a greater impact on rare classes. The rare class has a low anti-noise capability and the classifier can hardly distinguish between the rare class sample and the noisy data (Li & Fong 2018).

Most of the traditional pattern classification methods are based on the premise that the number of training samples is balanced. When used to solve imbalanced classification problems, their classification performance often decreases to varying degrees.

- **Classifiers based on feature space decision planes**, such as support vector machines, aim to find an optimal decision plane. In order to reduce the impact of noisy data and prevent the occurrence of over-learning, the optimal decision-making plane must take into account both the training classification accuracy and the complexity of the decision-making plane, that is, using the structural risk minimization rule (Li & Fong 2018). But when the data is unbalanced, the number of support vectors is also unbalanced. Under the principle of structural minimization, the support vector opportunity ignores the influence of a small number of rare support vectors on the structural risk and expands the decision boundary, which ultimately leads to the fact that the actual hyperplane of training is inconsistent with the optimal

hyperplane.

- **Classifiers based on probability estimation**, such as Bayesian classifiers, the classification accuracy depends on the accurate estimation of the probability distribution. When there are too few samples in the rare class, the accuracy of the probability estimation will be much smaller than that of the large class, and the recognition rate of the rare class will decrease (Li & Fong 2018).

- **Rule-based classifiers**, such as decision trees and association rule classification, need to filter rules. Among them, support and credibility are important indicators for rule screening, but when the data is imbalanced, screening based on the above indicators becomes difficult and unreasonable (Li & Fong 2018).

1.2.2 The Choice of Algorithms

In machine learning, the role of a classifier is to determine the category to which a new observation sample belongs based on the labeled training data. The project was based on the following four points are to select suitable algorithms.

1. **Trade-off between generalization ability and fitting**

   Overfitting evaluates the performance of the classifier on the training samples. If the accuracy of a classifier on the training sample is high, it means that the classifier can fit the training data well (Zhou 2012). However, a good classifier that fits the training data has a large bias, so it may not get good results on the test data. If a classifier can get good results on the training data, but the effect drops significantly on the test data, it means that the classifier has overfitted the training data. From another aspect of analysis, if the classifier can achieve good results on the test data, then the generalization ability of the classifier is strong. The generalization and fitting of classifiers are a process of fading away from each other. A classifier with a strong generalization ability is generally weak in fitting, and the other way around.

So the classifier needs to strike a balance between generalization ability and fitting ability.

2. **The complexity of the classification function and the size of the training data**

   The size of the training data is also crucial to the selection of the classifier. If it is a simple classification problem, a classifier with strong fitting ability and weak generalization ability can be obtained from a small part of the training data. Conversely, if it is a complex classification problem, then the classifier learning requires a large amount of training data and a strong generalization learning algorithm (Zhou 2012). Based on the complexity and size of the training data, a good classifier should automatically be capable of balancing the adaptation and generalisation ability.

3. **Dimension of the input feature space**

   If the vector dimension of the input feature space is high, the classification problem will become complicated, even if the final classification function is determined by only a few features (Zhou 2012). This is because an excessively high feature dimension will confuse the learning algorithm and cause the classifier's generalization ability to be too strong, while an excessive generalization ability will cause the classifier to change too much and performance will decrease (Zhou 2012). Therefore, classifiers with high-dimensional feature vector input generally need to adjust parameters to make their generalization ability weak and their fitting ability strong. In addition, in the experiment, this can also boost classifier efficiency by eliminating irrelevant characteristics from the input data or by raising the function dimension.

4. **The homogeneity and the relationship between the input feature vectors**

   If the feature vector contains multiple types of data (such as discrete, continuous), many classifiers such as SVM, Linear Regression, and Logistic Regression are not applicable (Zhou 2012). These classifiers require that the input features must be

numeric and normalized to a similar range, such as between. Classifiers such as K-nearest neighbor algorithm and SVM with Gaussian kernel are more sensitive to the uniformity of data. But another classifier decision tree can handle these heterogeneous data. If there are multiple input feature vectors, each feature vector is independent of each other, that is, the classifier output of the current feature vector is only related to the current feature vector input, then it is best to choose those based on linear functions and distance functions such as Linear Regression, SVM, Naive Bayes, etc. Conversely, if there are complex interrelationships between feature vectors, then decision trees and neural networks are more suitable for this type of problem (Zhou 2012).

1.3 Dissertation Contributions

This thesis presents the research and development of building a model to classify liver stiffness and automatically predict doctors' opinions by using Stacking ensemble method. The primary contribution of this study is including two aspects: the selection of the best imbalanced learning method and the selection of the best classification algorithm were obtained.

1. **Selection of the best imbalanced learning method**

   Due to the imbalance influence of the liver stiffness measurement data-set, this study evaluated the results of 7 different algorithms (KNN, RF, LR, NB, DT, SVM and AdaBoost) on EasyEnsemble undersampling method and SMOTE oversampling method, respectively, to select the best imbalanced learning method for the classification. In general, all evaluation scores of SMOTE oversampling method increased to a very high level, in contrary to the EasyEnsemble undersampling method. It is clear that SMOTE oversampling method provides a powerful ability on balancing the liver stiffness measurement data-set.

2. **Selection of the best classification algorithm**

   In the performance comparisons by 7 different algorithms (KNN, RF, LR, NB, DT, SVM and AdaBoost), the evaluation scores of KNN, RF, DT and SVM on SMOTE oversampled data-set are high and similar, therefore they are selected to be the base classifiers for the stacking models. In the performance comparisons by 7 different stacking models, the best classification performance is provided by the stacking model using DT as meta-classifier.

3. **Selection of the best stacking model**

   Among the stacking models, the stacking DT model performs on the SMOTE oversampled data-set resulting in the mean values of Precision rate (97.37%), Recall rate (97.36%), Specificity (99.47%), F1-score (97.36%) and F2-score (97.36%), and the highest Accuracy 97.38% with less deviation than the other models, in addition, greater than any of the supervised algorithms in the experiment.

1.4 Research Objectives

This work focuses on building a model to classify a real cirrhosis data-set collected by FibroScan from hospitals in China. The cirrhosis data-set is the actual conditions of patients consisting of 13,418 patients' information. According to these patients being given professional second opinions in Chinese characters by doctors, this model will analyse and classify these data, then predict a second opinion on the basis of new input patient data. Due to the digital limitations of programming in python, Chinese characters cannot be determined to process in the classification. Natural Language Processing (NLP) is perfectly suitable to analyse those Chinese characters and translate into digits. In the python module system, there is a NLP package named Jieba, which can segment a Chinese sentence into word groups and each group has a figure based on the dictionary edited by Jieba or user-edited. Besides the data pre-processing, to build an efficient model, one capable classifier is desirable to perform high-quality classification. Therefore, various algorithms will be

analyzed to select the best classifier for the cirrhosis data-set. Furthermore, based on the selected classifiers, combinations of algorithms will be proposed to improve the deficient performance of the model. Since there are 30 features included in the classification task, a genetic algorithm will be applied to select the optimised features groups for the best combinations of algorithms to enhance its performance and reduce its time consumption for training.

1.5 Thesis Organization

This thesis is organised as follows:

- *Chapter 2*: This chapter presents a comprehensive literature review. The background of liver cirrhosis, imbalanced learning methods, natural language processing, several classification algorithms and ensemble methods are outlined.
- *Chapter 3*: Methodologies are derived in this chapter. It presents the features of the liver stiffness measurement data-set, evaluation criteria of classification performance and the experiment procedure.
- *Chapter 4*: This chapter presents the results of my experiments. It discusses the comparisons of different algorithms and the stacking models' performance.
- *Chapter 5*: The main findings are presented in this chapter. It also indicates the contributions of this dissertation and offers recommendations for improvements in future work.

Chapter 2

Literature Review

In this chapter, we first provide a brief introduction and understanding of liver cirrhosis and the technique of FibroScan device used to measure liver stiffness. Then we introduce the imbalanced learning methods to balance the imbalance of the liver stiffness measurement data-set. To solve the natural language issues, we provide Jieba to translate the Chinese characters into computer language. Finally, a literature review of five popular classification algorithms including Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbour (KNN), Logistic Regression (LR) and Naive Bayesian (NB), also three ensemble methods such as Bagging, Boosting and Stacked Generalisation will be introduced.

2.1 Liver Cirrhosis

Cirrhosis is a chronic liver injury inflicting regenerative nodules encircled by fibrous bands, leading to malignant hypertension and end-stage disease (Schuppan 2008). Scar tissue replaces liver cells, but the scar tissue has no function. Scar tissue affects and prevents blood flow in the liver, thereby limiting the blood flow to the remaining liver cells. This lack of blood flow in the liver causes liver cells to die, causing a chain reaction that causes more scar tissue to form. In addition, due to the lack of blood flow, it causes an increase in pressure in the venous blood vessels (portal veins) from the intestine to the liver - a condition known as portal hypertension (usually occurring in the cirrhosis stage). It is worth noting that when fibrosis is caused by biliary obstruction, it can progress more rapidly—such as by primary biliary cirrhosis (PBC) or primary sclerosing cholangitis (PSC) (Schuppan 2008). It is also worth noting that short-term liver damage or destruction, even

if it is severe, usually does not cause liver fibrosis. This damage will take a while to affect liver health. If the cause can be confirmed and corrected in time, liver fibrosis can sometimes be reversed. However, fibrosis can progress after repeated or sustained liver damage for months or years. Large areas of scar tissue can be found in the liver, destroying the inner structure of the liver and affecting the function and self-renewal capacity of the liver. Such severe scarring is called cirrhosis. In general, there are four stages of liver damage shown in Figure 2.1. When a healthy liver stores up to a certain amount of fat, it can cause liver enlargement, which can gradually form scar tissues to destroy the liver cells leading to cirrhosis.

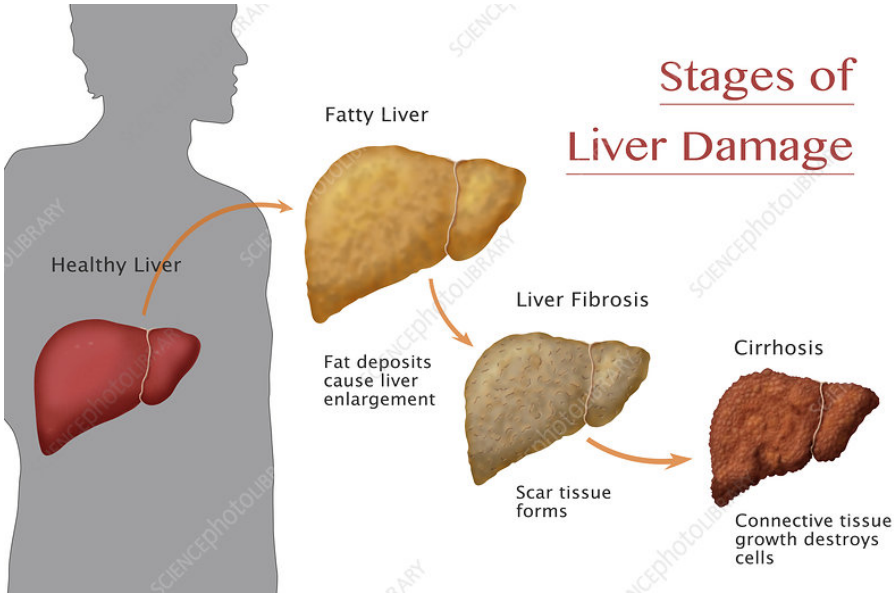


Figure 2.1 : Stages of liver damage (Science Source)

2.2 Liver Stiffness Measurement by FibroScan

The gold standard for the assessment of hepatic fibrosis is presently liver biopsy. However, It is a painful, invasive procedure with uncommon but life-threatening complications, which restricts its acceptability and repeatability in typically asymptomatic patients. In addition, due to sampling error and inter-observer variability, which can lead to underestimating the stage of cirrhosis, the precision of liver biopsy in evaluating fibrosis may

be challenged (Foucher & Chanteloup 2005). Therefore, the complete spectrum of liver fibrosis, cirrhosis and severity in hepatic conditions need non-invasive tests to be developed and validated.

There is growing interest in creating non-invasive means for evaluating liver fibrosis in patients with chronic liver disease to determine the seriousness of the disease, prognosis and ideal therapy. Previously, transient elastography (TE) has been shown to predict the existence or lack of advanced fibrosis (Klibansky, Mehta & Curry 2011). It is essential that cirrhosis is not diagnosed in an invasive way. Liver stiffness for the highest modality for compensated cirrhosis with TE, the aspartate aminotransferase to platelet ratio index (APRI), ast / alt ratio, hyaluronic acid, and clinical signs have been tested in many biomedical examinations. TE can precisely recognize that hepatic stiffness $> 12\ kPa$ is a significant clinical metric for cirrhosis diagnosis (Malik & Lai 2010). FibroScan with TE is the most popular device to make accurate measurements of liver stiffness.

TE does not evaluate fibrosis conditions and thus, for several reasons, incorrect elevations (i.e. overestimation of liver fibrosis) are noted (Table 2.1). Liver stiffness measurements should therefore be properly interpreted and these potential confusing factors should be addressed. Moreover, precise measurements cannot be obtained in some patients. This is because the method of ultrasound needs appropriate liver visualization for reading. A major European research study found that FibroScan was unable to achieve precise measurements in approximately 20 percent of patients. Body mass index (BMI) $> 30\ kg/m^2$, increasing age and characteristics of metabolic syndrome among others have been correlated with reading failure or non-reliability (Kemp & Roberts 2013). The existence of ascites avoids the propagation of the vibration wave, which is why measurements sometimes fail. However, in the testing progress using TE will not be terminated as fail because above correlated factors. Therefore, doctors' opinions become the most critical determinant.

Table 2.1 : Possible factors effecting the accuracy of FibroScan results

Over-estimation	**Failure or unreliable readings**
Liver inflammation eg. active hepatitis	Obesity (BMI $> 30\ kg/m^2$)
Cholestasis eg. biliary obstruction	Older age (over 60 years old)
Mass lesions within the liver eg. tumour	Presence of ascites
Liver congestion eg. heart failure	Features of metabolic syndromes (type 2 diabetes, hypertension, increased waist circumference)

FibroScan using TE technique recently enabled quick liver rigidity measurements. Vibrations with moderate amplitude and low frequency (50 Hz) through the liver tissue are transferred through an ultrasound transducer probe. This leads to an elastic shear wave spreading through the underlying hepatic tissue. In order to follow the shear wave spread and evaluate its speed, the sensor uses a pulse-echo ultrasound (Wilder & Patel 2014). The speed of the wave is directly linked to fibrosis-related tissue radiation. In this process, various parameters including vibration speed, wave propagation speed, and elastic modulus can be evaluated. TE enables the identification of the seriousness of the disease owing to the alteration of fibrotic liver mechanical characteristics. TE is an extremely easy and secure 5 to 10 minutes method that can be performed in an outpatient or specialty hospital (Wilder & Patel 2014). Preparation for the patients is required, because of the potential rise in hepatitis stiffness owing to post-prandial blood flow, for patients 2 to 3 hours before the operation. The patient is positioned dorsally in a maximal abduction position with the correct arm. The test starts with the positioning of the probe in the intercostal room to get an overview of the liver's right lobe. Wilder and Patel (2014) stated that the FibroScan probe is used to achieve ten readings when an area with a thickness of approximately 6 cm free of vascular structures or a big bladder has been recognized. The real quantity of the region measured by the probe exceeds the average biopsy of the

liver. As fibrous tissue is harder than ordinary liver, liver stiffness may deduce the degree of hepatic fibrosis. Kemp and Roberts (2013) suggested that to improve test reliability, the results expressed in kPa are taken to achieve a minimum of 10 valid readings with a minimum success rate of 60% and an interquartile range of $\leq 30\%$ of the median value.

The outcomes of FibroScan measurement range between 2.5 and 75 kPa. A liver scarring measurement < 7.0 kpa (median is 5.3 kPa) is expected between 90% to 95% of healthy individuals without liver disease (Kemp & Roberts 2013). The ideal cut-off in cirrhosis detection is approximately 14 kPa, according to validation research, including an extensive systemic review of research using liver biopsy as the gold standard for evaluating liver scarring. The likelihood of cirrhosis in patients with chronic hepatitis C and hepatitis > 14 kPa, whereas patients with liver rigidity > 7 kPa have a likelihood of cirrhosis of about 85%, or more of fibrous diseases. However, for patients with chronic liver disease, FibroScan can not entirely exclude the chance of important liver disease, even though liver stiffness is < 7 kPa, the sensitivities and specificity of > 7 kPa measurements for major fibrosis are only about 79% and 78%, respectively (Kemp & Roberts 2013).

Table 2.2 : Various conditions of liver disease with different cutoff values: F_0 to F_1 is the first stage of liver fibrosis; F_2 is the second stage of liver fibrosis; F_3 to F_4 is the stage of liver cirrhosis (University of Health Network 2018)

Condition	Fibrosis stages and approximate cutoff values			
	F_0 to F_1	F_2	F_3	F_4
Hepatitis B Hepatitis C HIV (co-infection)	2 to 8	8 to 10	10 to 14	14 or higher
Cholestatic Liver	2 to 7	7 to 9	9 to 17	17 or higher
NASH or NAFLD	2 to 7	7 to 10	10 to 14	14 or higher
Alcoholic Hepatitis	2 to 7	7 to 11	11 to 19	19 or higher

2.3 Imbalanced Learning Methods

Imbalanced data always appears in most reality classification tasks. At present, minority class detection and learning based on imbalanced data are not only concerned about as problems in the field of data mining, but have become problems across research fields (from management to engineering). In the medical field, the degree of data imbalance can easily exceed 100: 1, and even more, 100 000: 1. An imbalanced data classification is more biased towards the majority class, for example in Figure 2.2, the class (Circle) with more instances is called the majority class, the class (Triangle) with relatively few instances is called the minority class. Obviously, Class 1 instances are much more than Class 2 instances. It can have 78.57% classification accuracy by a classifier with all Class 2 instances mis-classified, which it is not acceptable.

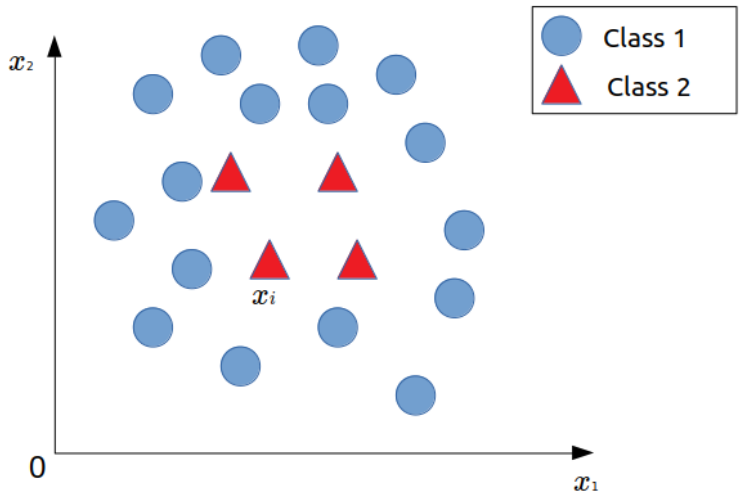


Figure 2.2 : Example of imbalanced data distribution: 20 instances (16 circles belonging to Class 1 and 4 triangles belonging to Class 2) with 5:1 amount proportion

In the case of imbalanced data, the number of instances in the minority class is much less than that in the majority class, which will generate more sparse instances (those in the subclass with a small number of instances). Due to the lack of sufficient data, the classifier cannot describe sparse instances, and it is difficult to effectively classify these sparse instances (Almeida 2015). In Figure 2.3, it is two classes classification. The solid

frame is the ideal decision area, and the dashed frame is the actual decision area. For sparse instances (a small number of + class instances on the right), a small shift in the decision region will lead to a higher classification error rate. As can be seen from the figure, a slight shift in the same decision region results in the sparse sample part. The error rate is close to 50%, and the non-sparse instance is only about 10%.

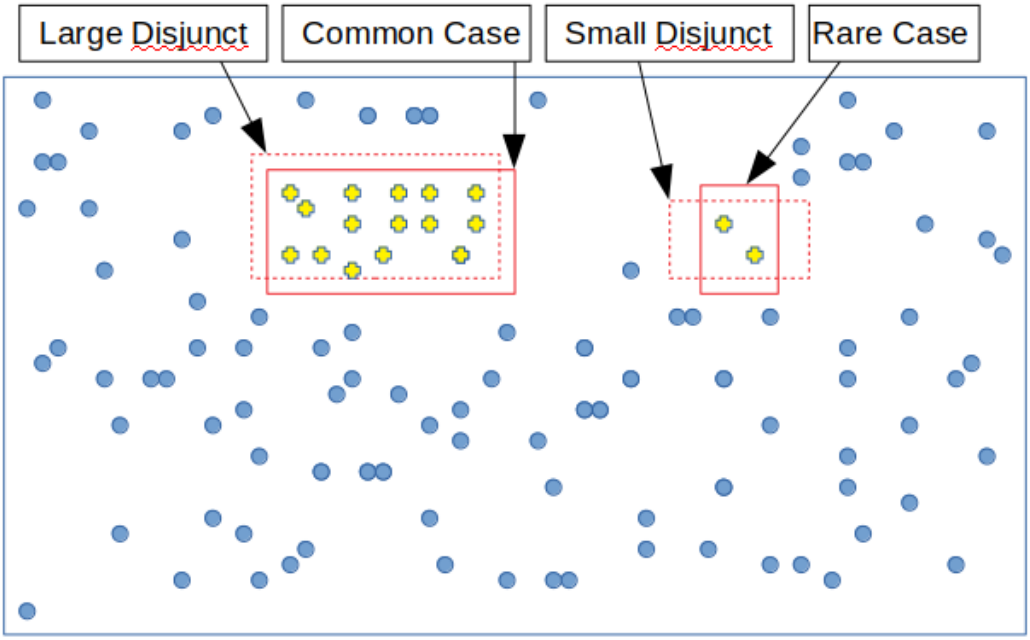


Figure 2.3 : Decision Tree classifier decision area problem due to imbalanced data

In the case of considering data noise, the sparse instances described in Figure 2.3 may cause more problems, and a small amount of noise data may have a greater impact on the classification of sparse instances. Taking the decision tree classification as an example, Figure 2.4 shows the change of the decision area when the data does not contain noise and the noise is included. It can be seen that the data noise makes the number of sparse instance sub-classes from 2 to 1. The generation condition of the branch of the decision tree is not satisfied (the number of instances under the branch is greater than or equal to 2), which causes the decision region corresponding to the sparse instance to disappear, and the classifier cannot classify the sparse instance of this subclass.

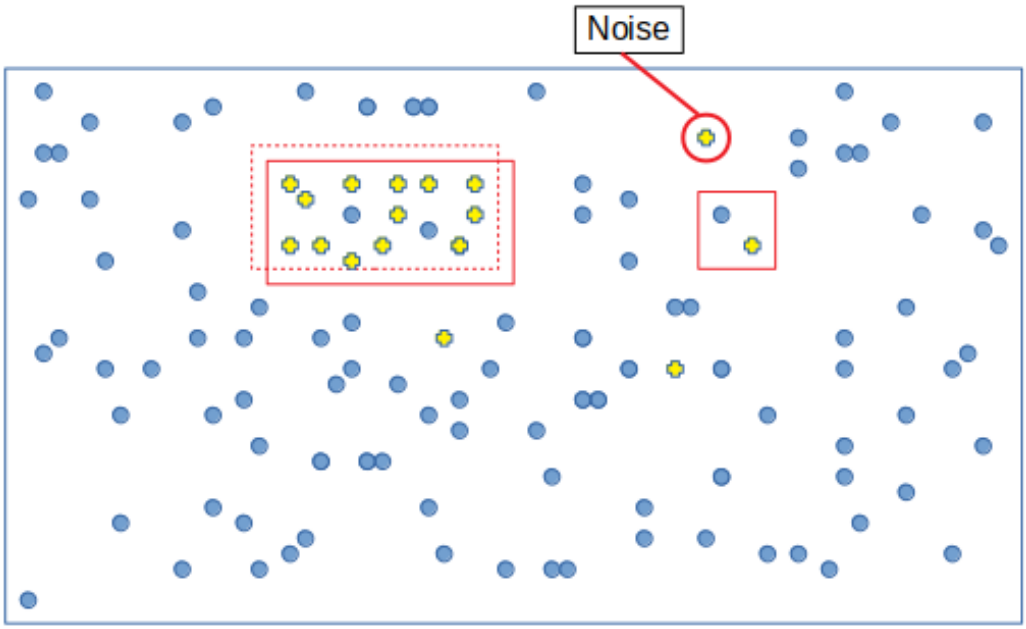


Figure 2.4 : Influence of noisy imbalanced data for Decision Tree classification

2.3.1 Sampling Methods

The pre-processing method of imbalanced data is to change the distribution of the imbalanced data-set through some algorithms to obtain a balanced data-set, for example, Sampling methods. For imbalanced data, using Sampling methods to re-balance the sample space can alleviate its imbalance. Sampling methods are mainly divided into two categories, Oversampling, Undersampling.

2.3.1.1 Oversampling

Oversampling is a method to reduce the imbalance by increasing the number of training instances for the rare class. There are two popular methods: Random Oversampler and Synthetic Minority Oversampling Technique (SMOTE).

2.3.1.1.1 Random Oversampler

Random oversampling is to randomly sample from a small number of instances, and then add the sampled instances to the data-set (Huang 2015). The biggest advantage

of this approach is simplicity, but its shortcomings are also very obvious, and the trained models tend to be severely over-fitting. Due to repeated sampling, some points in the space will appear repeatedly (Almeida 2015). As a result, when the model divides a sample point that is repeatedly sampled, it also divides the repeated sample points together, making the model sub-pairing or mismatching many sample points. A simple and effective solution is to add a slight random perturbation when a new sample point is generated for each sampling.

2.3.1.1.2 Synthetic Minority Oversampling Technique (SMOTE)

In 2002, a study proposed an oversampling method called Synthetic Minority Oversampling Technique (SMOTE) which was based on the technique of Random Oversampler (Chawla et.al. 2002). SMOTE is an optimised technique of Random Oversampler. Random Oversampler adopts a strategy of simply copying instances to increase a small number of instances, this easily leads to the problem of over-fitting the model, that is, the information learned by the model is too specific to generalize (Cao 2016). Unlike Random Oversampler, SMOTE aims at evaluating a small number of instances and synthesising new instances artificially based on a limited number of instances for addition to the data-set (shown in Figure 2.5) (Xie et al. 2019).

As the number of instances of a minority class in the training set is T, then the SMOTE algorithm will synthesize NT new instances for this minority class. It is required that N must be a positive integer. If $N < 1$ is given, the number of instances in the minority class is $T = NT$ and N will automatically equal to 1 (Lei et al. 2019). Consider an instance i of this minority class x_i, $i \in 1, ..., T$:

1. First find the k nearest neighbors $x_{i(n)}$of instance x_i from all T instances of the minority class;

2. Then randomly select an instance $x_{i(n)}$ from these k neighbors, and then generate a

random number as coefficient θ between 0 and 1 to synthesize a new sample $x_{i(new)}$:

$$x_{i(new)} = x_i + \theta * (x_{i(n)} - x_i)$$

3. Repeat step 2 N times to generate N new instances $x_{i(new)}, new \in 1, ..., N$.

If the feature dimension of an instance is in two dimensions, each instance can be represented by a point on a two-dimensional plane. A new instance $x_{i(new)}$ synthesized by the SMOTE algorithm is equivalent to a point on the line segment between the point representing the instance x_i and the point representing the instance $x_{i(n)}$ (see Figure 2.5).

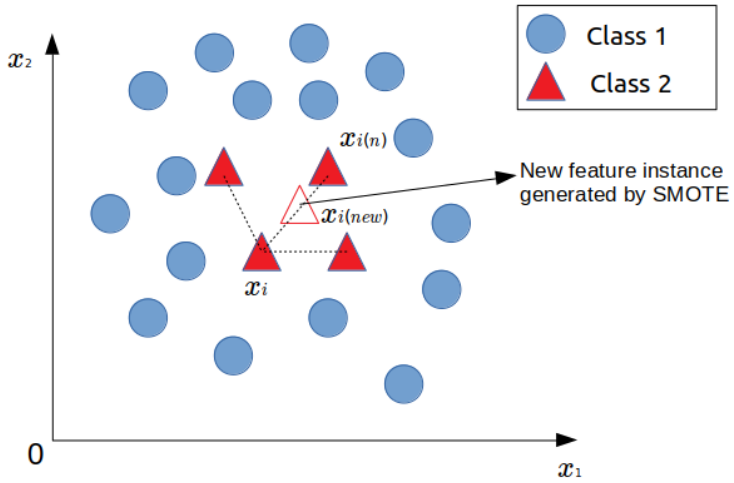


Figure 2.5 : Example of new instance creation by SMOTE: the new instance $x_{i(new)}$ is located between x_i and $x_{i(n)}$

2.3.1.2 Undersampling

Undersampling is a method to alleviate class imbalance, which is achieved by discarding samples. It discards the instances belonging to the class with a large number of instances to match the amount of the class with less number of instances to balance the data-set.

2.3.1.2.1 EasyEnsemble

EasyEnsemble randomly divides most types of instances into n subsets, and the number of each subset is equal to the number of minority types of instances, which is equivalent to undersampling. Then, each subset is combined with a small number of instances to train a model separately, and n models are finally integrated. In this way, although the number of instances in each subset is less than the overall instance, the total information after integration does not decrease.

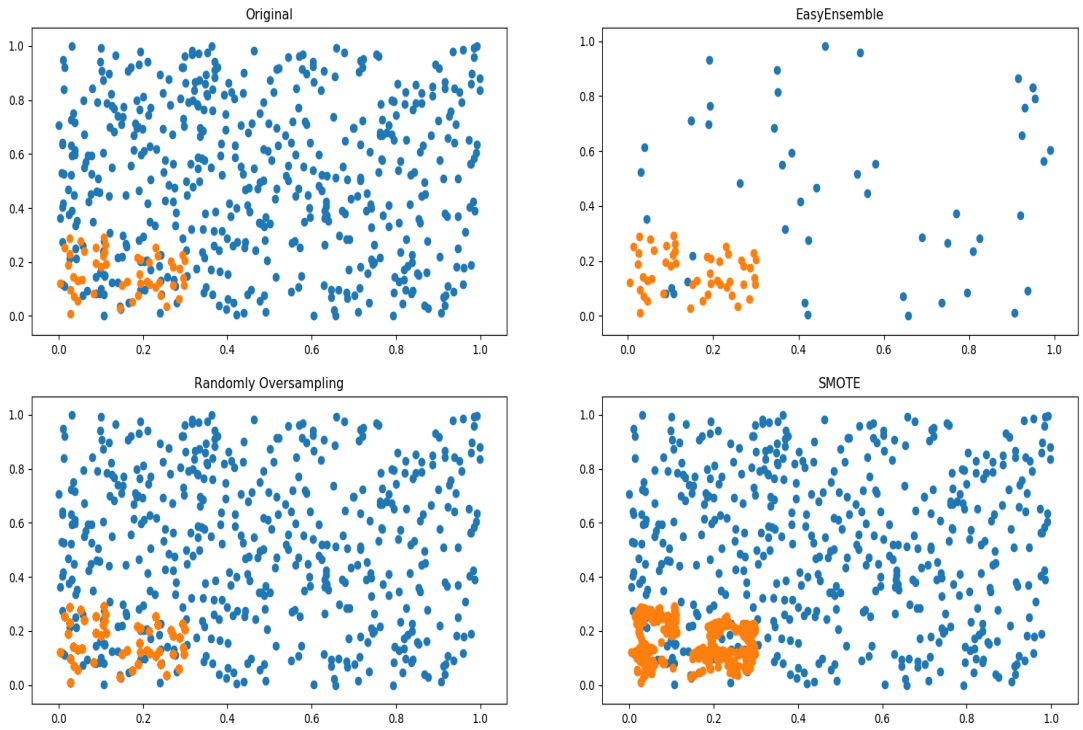


Figure 2.6 : Original data: without any sampling process; EasyEnsemble: randomly select 50 instances from the negative class and compare with the positive class; Randomly Oversampling: repeatedly extract and generate 500 data from the positive class (which will inevitably repeat), and compare with the negative class; SMOTE: By finding the nearest neighbors of the instances in the positive class, "$500-50=450$ new instances in the positive class" are synthesized and merged with the original data.

Let a data with 550 original data in two dimensions consists of 500 instances belonging to the negative class and 50 instances belonging to the positive class, which is an imbalanced data-set. A visualized diagram is presenting the performance of EasyEnsemble, Randomly Oversampling and SMOTE in Figure 2.6:

i. EasyEnsemble discards most of the negative instances, thereby weakening the impact of the negative part, which may cause a model with large deviations.

ii. Randomly Oversampling simply repeats the positive instances, so it will overemphasize the existing positive instances. If some of these points are marked incorrectly or are noisy, the errors are easily magnified. So the biggest risk is over-fitting the positive instances.

iii. SMOTE can be seen to be significantly different from Randomly Oversampling, because instead of simply repeating positive instances, new positive instances are generated in local areas through K-nearest neighbors. Compared to simple oversampling, SMOTE: Reduced risk of over-fitting (K-nearest neighbors can be understood as a kind of ensemble learning in locally synthesized data, which reduces variance); More resistant to noise.

2.4 Natural Language Processing

Natural Language Processing (NLP) represents a field in which computer science, information engineering and intelligence deal in specific how computers can be programmed to process and analyze large quantities of natural linguistic data, concerning interactions between computers and human (natural) languages (Raut 2017). Approximately, 80% of the information generated is unstructured with digitalisation (Raut 2017). Types of unstructured data include audio, video, social media, data generated from conversations with customer service agents, legal documents and financially processed messages. Organisations use NLP to understand the myriad unstructured data available online, in call logs and other sources.

NLP is a computer science subfield devoted to the use of computer technology to learn, understand and develop human language content. NLP is an artificial intelligence branch that has a number of significant consequences for the interaction of computers and

people. Machine learning enabled computers to analyse human language ambiguities (Raut 2017). Computer language systems can be designed for a number of purposes: i.facilitating the communication between humans, such as machine translation (MT); ii.to support communication between humans and machines, for example with conversational agents; iii.to benefit both people and machines through the analysis and learning of the vast amount of human language content (Hirschberg & Manning 2019). There are a variety of Open-source NLP libraries that are used in actual-world apps, like Natural Language Toolkit (NTLK), Apache OpenNLP and Stanford NLP. The most basic and common problem of understanding what human language content is to deal with written words or text analyse in different human languages.

The underlying tasks of NLP can be roughly divided into lexical analysis, syntactic analysis and semantic analysis from easy to difficult. Word segmentation is the most basic task in the lexical analysis (also including part-of-speech tagging and named entity recognition). The initial step in learning language is to extract words from speech (Gambell & Yang 2005). In the case of my study, Chinese is the only language needed to be processed. In Chinese text, phrases or sentences are depicted as strings of Chinese characters without comparable natural bounder, unlike the English text, in which phrases are white space-delimited word sequences. Thus, the first stage of Chinese language processing is to define the sequence of the words in a phrase or sentence and determine borders in relative positions (Xue 2003). Segmentation of the Chinese words refers to the division into one word of a sequence of Chinese characters. Word segmentation is the process by which sequences of words are recombined into word sequences according to certain standards.

There are three categories of existing word segmentation algorithms: string-based segmentation methods, understand-based word segmentation methods and statistical-based word segmentation methods (Luo 2011).

- String-based segmentation methods: This method is also called mechanical word

segmentation method. It is based on a certain strategy to match the Chinese character string to be analyzed with the term in a "sufficiently large" machine dictionary. If found in the dictionary A string, the match is successful (a word is recognized).

- Understand-based word segmentation methods: This word segmentation method allows the computer to simulate human understanding of the sentence by identifying words. The basic concept is to conduct syntax and semanticized research simultaneously with word segmentation and to deal with ambiguity with syntactic information and semantic information. The sub-sectioning of words, the syntactic and semantic subsystems, and the general control part are usually three components. The word segmentation subsystem can obtain, under coordination of the general control section, syntactic and semantic information about the word, phrases, etc. for the purpose of determining the ambiguity of the participle. A large amount of language awareness and information is required for this method of word segmentation. Diverse linguistic information can hardly be organised in a form that machines can interpret directly, as Chinese language knowledge is complicated. Therefore, the understanding-based word segmentation method is still at the experimental level.

- Statistical-based word segmentation method: Give a large number of texts that have been segmented, and use the statistical machine learning model to learn the rules of word segmentation (called training), so as to achieve the segmentation of unknown text. For word segmentation, for instance, the highest likelihood word segmentation method and the maximum entropy word segmentation methods are widely used (Meng 2019). The method for the segmentation of Chinese words based on statistics has gradually become the mainstream approach by establishing large-scale corpus, research and development of statistical machine learning methods (Meng 2019). The primary statistical models are N-gram, Hidden Markov (HMM) and Maximum Entropy Model (ME) and Conditional Random Fields (CRF).

There are three most common Chinese word segmentors accessible with Python APIs: ICTCLAS, THULAC, and Jieba segmentor. ICTCLAS is expected to be a standard in Chinese word segmentation developed by Dr. Zhang Huaping. THULAC invented by Tsinghua University also integrates part-of-speech (POS) tagging into the software, similar to ICTCLAS. Jieba is the most easily and quickly adapted open-source Chinese text segmentor in Python compared with Python APIs in other segmentors (Peng, Cambria & Hussain 2017).

2.4.1 The Principle of Jieba Word Segmentation

Jieba is supporting three words segregation modes: Precise mode, which is attempting to make the sentence more accurate, appropriate for analysis of texts; Full mode, which scan all the words that can be uttered in a sentence and its processing speed is very fast, but the ambiguity can not be solved; Search engine mode, which is based on the precise mode and able to split long words to improve the recall rate (Zhu et al. 2017). The principles of Jieba Chinese word segmentor is simply concluded in Figure 2.7:

a) Based on the Trie tree structure, a directed acyclic graph (DAG) is generated consisting of all possible Chinese idioms (Fang et al. 2020). Jieba comes with a dictionary called *dict.txt*, which contains more than 20,000 Chinese words, including the number of occurrences and part of speech. When the dictionary generates a Trie tree, the number of occurrences of each word is also converted into a frequency. According to the Trie tree, it generates the DAG to record the position of each word.

b) Jieba uses dynamic programming to find the highest likelihood direction and the highest word frequency segmentation combination (Fang et al. 2020). The program determines the segmented words in the sentence and finds the frequency of the occurrence of the word, and if there is no such word, use the frequency of the least frequently occurring word in the dictionary as the frequency of the word. According

to the dynamic programming, calculating the maximum probability for the sentence from right to left to find the maximum probability path.

c) The HMM model based on the Chinese character capability and the Viterbi algorithm is used for the non-registered words in the dictionary (Fang et al. 2020). There are mainly three probability tables to be counted: 1) position transition probability, that is, the transition probability of the four states of B (beginning), M (middle), E (end), and S (independent word formation); 2) position to single word Launch probability, such as P (" 和" | M) means the probability of the word " 和" appearing in the middle of the word; 3) There are only two kinds of probability that a word starts with a certain state, either B or S. Then find the hidden state sequence by Viterbi algorithm. The Viterbi function first calculates the log probability value of each initial state, and then calculates it recursively. The log probability value of a state at each moment depends on the log probability value of the previous moment, the state of the previous moment, and the state of this moment. It consists of three parts: the transition probability and the emission probability of the state transition to the current word at this moment. The maximum probability log and the optimal path will be the output of the program.

2.5 Classification Algorithms

In this section, five popular classification algorithms will be presented: Support Vector Machine, Decision Tree, K-Nearest Neighbour, Logistic Regression and Naive Bayesian. The principles of each algorithm will be briefly introduced following.

2.5.1 Support Vector Machine

Support vector machine (SVM) is a type of generalized linear classifier based on supervised machine learning to classify data points (Ma et al. 2019). Using a kernel trick, SVM discovers the optimal separating hyperplane within the sample data and utilizes the

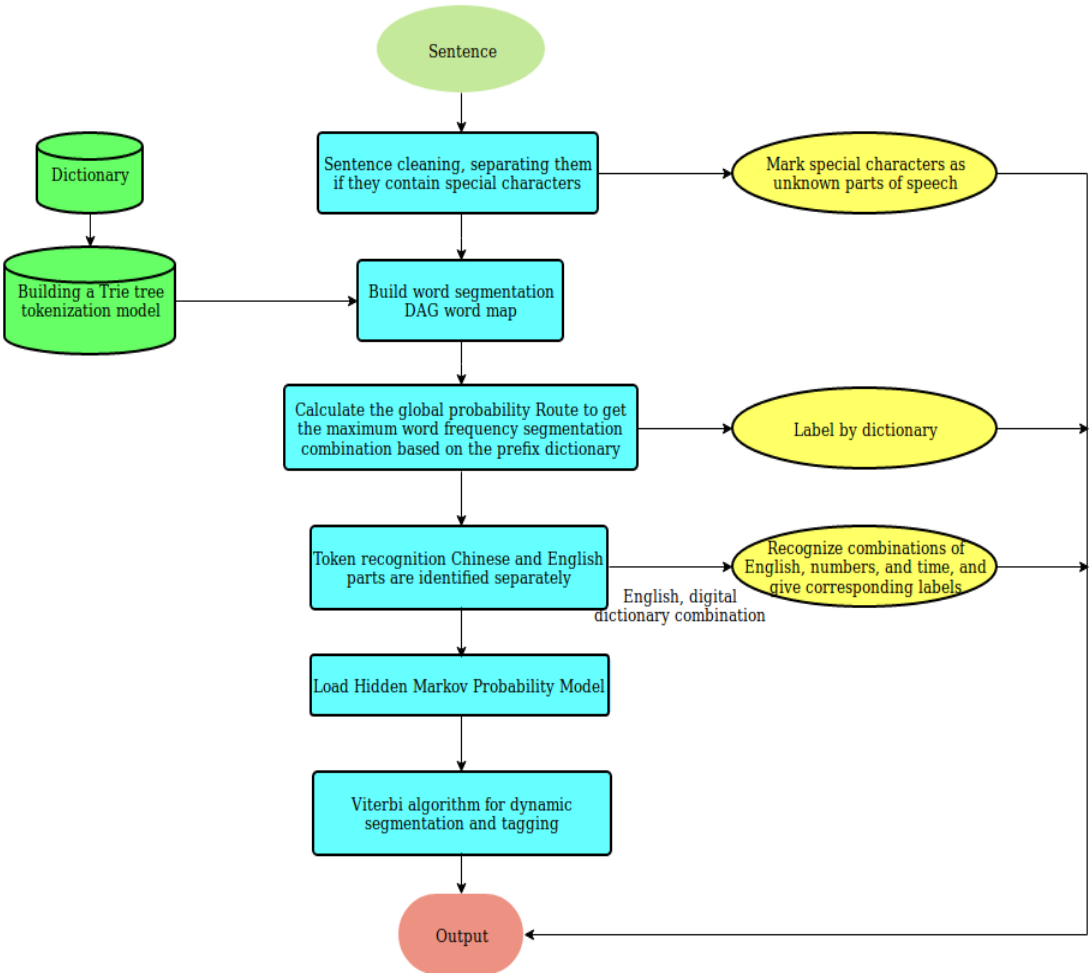


Figure 2.7 : The flow chart of Jieba principle

regularization parameter C to monitor the complexity of the model and training error of a training range from an input space to a high-dimensional feature space (Li, Wang & Sung 2008). The higher C is, the less the error is tolerated and the easier it is to over-fit, conversely, the easier it is to under-fit. The parameter γ of the Radial Basis Function (RBF) describes a nonlinear space-to-high-dimensional mapping functionality (Li & Kong 2014). This defines implicitly the distribution of the data to a new space after mapping. The γ value is adversely commensurate with the number of support vectors. Therefore, the parameters (C and γ) affect strongly the effectiveness and performance of the SVM system.

Given input data and learning objectives in the classification problem: $X = \{X_1, ..., X_N\}$,

$y = \{y_1, ..., y_N\}$, where each input instance comprises many characteristics and thus constitutes the space for the characteristics: $X_i = \{x_1, ...x_n\} \in \chi$, the learning objectives are binary variables: $y = \{-1, 1\}$ corresponds to a negative class and a positive class (Li & Kong 2014). If the feature space in which the input data is located exists as a hyperplane of the decision boundary, the learning target is separated by a positive class and a negative class, and the point-to-plane distance of any instance is greater than or equal to 1:

Decision boundary: $$\omega^T X + b = 0 \tag{2.1}$$

point to the plane distance: $$\mathrm{y}_i(\omega^T X_i + b) \geq 1 \tag{2.2}$$

The classification problem is then linearly separable, ω, b parameters are the normal vector and hyperplane interception. The boundary of decision that fulfils this condition constructs two parallel hyperplanes to discriminate between the instance classification (Fan et al. 2019):

$$\begin{cases} \omega^T X_i + b \geq 1 & y_i = +1 \\ \omega^T X_i + b \leq 1 & y_i = -1 \end{cases}$$

The positive class is the case for all instances above the upper interval boundary and the negative class for all samples below the upper interval boundary. Distance between two spaced boundaries: $d = \frac{2}{\|\omega\|}$. Some linear indivisible problems can be nonlinear, i.e. the hypersurface between the feature space is positive and negative. It use nonlinear functions to map nonlinearly separable problems from the original feature space to higher-dimensional Hilbert spaces, thus transforming it into a linearly separable problem, the hyperplane as the decision boundary is expressed as follows: $\omega'\phi(X) + b = 0$, (ϕ is a mapping function). Since the mapping function has a complex form and it is difficult to calculate its inner product, a kernel method can be used, that is, the inner product of the mapping function is defined as a kernel function $\kappa(X_1, X_2) = \phi(X_1)^T\phi(X_2)$.

Popular kernel functions:

Polynomial kernel: $\kappa(X_1, X_2) = (X_1^T X_2)^n$

RBF kernel: $\kappa(X_1, X_2) = exp(-\frac{\|X_1 - X_2\|^2}{2\sigma^2})$

Laplacian kernel: $\kappa(X_1, X_2) = exp(-\frac{\|X_1 - X_2\|}{\sigma})$

Sigmoid kernel: $\kappa(X_1, X_2) = tanh[a(X_1^T X_2) - b]$

Linear SVM: hard-margin & **soft-margin**

Hard-margin SVM is a solution algorithm for a linearly separated problem to the maximum marginal hyperplane. The limit is that the distance between the instances and the decision boundary is equal to or greater than 1. SVM's hard-margin can be converted into a convex quadratic optimisation (Fan et al. 2019):

$$\max_{\omega,b} \frac{2}{\|\omega\|} \quad \Leftrightarrow \quad \min_{\omega,b} \frac{\|\omega\|^2}{2}$$

$$s.t. \quad y_i(\omega^T X_i + b) \geq 1$$

Although the normal vector for hyperplanes is the only optimisation target, the learning data and the interception of the hyperplanes are restricted and affect the solution to the optimisation problem. The SVM with hard-margins is an SVM with a soft-margin where the regularisation rate is 0.

A hard-margin SVM will cause classification errors in linear inseparable problems, so a loss function may be introduced to maximise margins in order to create a new problem of optimisation. The SVM uses the hinge loss function, which is a consequence of the SVM hard-margin optimisation problem. The SVM soft-margin optimisation problem is as follow (Fan et al. 2019):

$$\min_{\omega,b} \frac{\|\omega\|^2}{2} + C \sum_{i=1}^{N} L_i \tag{2.3}$$

$$L_i = max[0, 1 - y_i(\omega^T x_i + b)] \tag{2.4}$$

$$s.t. \quad y_i(\omega^T X_i + b) \geq 1 - L_i$$

The above equation shows that the soft margin SVM is an L2 regularization classifier, where L_i represents the hinge loss function. Using the slack variable: $\zeta \geq 0$ to deal with the segmentation value of the hinge loss function, the above formula can be turned into:

$$\min_{\omega,b} \frac{\|\omega\|^2}{2} + C\sum_{i=1}^{N} \xi_i \tag{2.5}$$

$$s.t. \quad y_i(\omega^T X_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

The Lagrangian function can be obtained by Lagrangian multiplier $\alpha = \{\alpha_1, ..., \alpha_N\}$, $\mu = \{\mu_1, ..., \mu_N\}$ to solve the above soft margin SVM so that to have the dual problem of the primal problem.

$$\begin{aligned} L(\omega, b, \zeta, \alpha, \mu) =& \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{N} \xi_i \\ &+ \sum_{i=1}^{N} \alpha_i[1 - \xi_i - y_i(\omega^T X_i + b)] - \sum_{i=1}^{N} \mu_i \xi_i \end{aligned} \tag{2.6}$$

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}[\alpha_i y_i (X_i)^T (X_j) y_j \alpha_j] \tag{2.7}$$

$$s.t. \qquad \sum_{i=1}^{N} \alpha_i y_i = 0, \qquad 0 \leq \alpha_i \leq C$$

The constraint condition of the dual problem contains unequal relations, so its local optimal condition is that the Lagrange multiplier satisfies the Karush-Kuhn-Tucker (KKT) condition.

$$\begin{cases} \alpha_i \geq 0, \quad \mu_i \geq 0 \\ \xi_i \geq 0, \quad \mu_i \xi_i = 0 \\ y_i(\omega^T X_i + b) - 1 + L_i \geq 0 \\ \alpha_i[y_i(\omega^T X_i + b) - 1 + L_i] = 0 \end{cases}$$

According to the above KKT conditions, for any sample (X_i, y_i) always have $a_i = 0$, the sample will not affect the decision boundary ($\omega^T X_i + b = 0$), or $y_i(\omega^T X_i + b) = 1 - \xi_i$, which means the sample is a support vector (Fan et al. 2019). It can be seen that the determination of the soft margin SVM decision boundary is only related to the support vector, and the SVM is sparse using the hinge loss function.

SVM is a small sample learning method. SVM can also maximize the classification margin to separate different class instances, which can reduce the error rate. In non-linear classification cases, SVM uses the inner product kernel function to map high-dimensional space. It means that the decision function of SVM is determined by a few vectors instead of the dimensions, which is more flexible to classify the data. However, SVM classifier is difficult to select the most suitable kernel function providing the best performance while solving specific classification cases (Fan et al. 2019). When a data-set is too large and consists of a number of missing data, SVM might fail to have precise classification results, especially for multi-classification problems.

Table 2.3 : Advantages and disadvantages of Support Vector Machine

Advantages	**Disadvantages**
Inner product kernel function instead of non-linear mapping to high-dimensional space	Difficult to implement on large training samples
Maximize classification margin	Difficulties in solving multi-classification problems
The decision function determined by only a few support vectors	Sensitive with missing data
Small sample learning method	

2.5.2 Decision Tree

A decision tree (DT) is defined as a clustering procedure, it uses a flow diagram-like tree structure which divides data into smaller subdivisions recursively by a set of tests defined in the tree at each branch (or tree node) (Yu et al. 2010). DT are extremely efficient instruments for many fields such as mining of data and text, extraction of information, machine learning and pattern recognition (Bhargava et al. 2013). DT as a logical model shows how the value of the target variable can be predicted by using the values from a number of predictor variables. The tree consists of root nodes, a set of internal nodes (splits), and a set (leaves) of terminal nodes. Every node has a single parent node and two or more downstream nodes in a decision tree (Bhargava et al. 2013). A data-set is therefore classified by sequence, according to the tree-defined decision framework, and each observation is assigned a class label according to the leaf node into which the observation falls. In this framework (Friedl & Brodley 1997). DT are mainly ID3, C4.5, C5.0 and CART algorithms, ID3, C4.5, and CART are greedy (non-backtracking) algorithms. In this paper, CART algorithm is involved due to the DT model setup in Python.

The CART classification tree predicts the classification of discrete data, selects the optimal feature using the *Gini* index, and determines the optimal segmentation point of the feature in binary (Lu et al. 2020).

Assuming K classes in a case of classification, the probability that the instance points belong to the K_{th} class is P_k, the *Gini* index of the probability distribution is defined as:

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2 \tag{2.8}$$

According to the definition of the *Gini* index, the *Gini* index of the sample set D can be obtained, where $C_k$represents the subset of samples belonging to the K_{th} class in the data

set D.

$$Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2 \tag{2.9}$$

If the data set D is segmented according to the feature A on a certain value a, and the two parts D_1 and D_2 are obtained, then the $Gini$ coefficient of the set D under the feature A is as follows (Lu et al. 2020). The $Gini$ coefficient $Gini(D)$ reflects the uncertainty of the D set, and the Gini coefficient $Gini(D, A)$ represents the uncertainty of the set D after $A = a$ partitioning (Ren et al. 2019). The uncertainty of the sample set is proportional to the $Gini$ index.

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{2.10}$$

For attribute A, calculate the arbitrary attribute value to divide the data set into two parts, $Gain\_Gini$, and select the minimum value as the optimal binary searching tree cases obtained by attribute A. Then for the training set S, calculate the optimal binary searching tree cases of all attributes, and select the minimum value as the optimal cases of the sample and S.

$$\min_{i \epsilon A}(Gain\_Gini(D, A))$$

$$\min_{A \epsilon Attribute}(\min_{i \epsilon A}(Gain\_Gini(D, A)))$$

DT is one of the choices to make classification with a large data-set. It provides efficient and effective performance with high-dimensional data, even when the data-set has missing data because DT does not require to clean up the data. However, DT is difficult to predict a result when the instances are in a time-sequence relationship and hard to classify data with strongly relative features.

Table 2.4 : Advantages and disadvantages of Decision Tree

Advantages	**Disadvantages**
No need for Data cleaning	Difficult to predict time-sequence data
Not sensitive with missing data	Difficult to classify data with strongly relative features
Efficient and good performance with large data-set	Easy over-fitting
Able to process irrelevant features	

2.5.3 K-Nearest Neighbour

The K-Nearest Neighbor (KNN) is a method of instance-based learning where the function is locally approximated and all the calculations are postponed until classification (Imandoust & Bolandraftar 2013). The core concept of the KNN algorithm is that if most of the neighbouring instances in a feature space are defined as belonging to a certain class, the testing instance will also be classified as belonging to this class and has similar characteristics in this class (Suguna & Thanushkodi 2010). The procedure specifies the instance class by which the instance is to be categorised by evaluating the classification decision based only on a class of the nearest one or more instances. For class decision-making, the KNN method is only applicable to a very limited number of neighbouring samples. Since the KNN method relies primarily on the restricted samples across the category instead of relying on the discrimination domain process, to classify the crossover or overlapping sample set, the KNN method is more suitable than the other methods.

KNN algorithm procedure (Li et al. 2019):

Input training data:

$$T = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N) \tag{2.11}$$

Where $x_i \in X \subseteq Rn$ is the eigenvector of the instance, $y_i \in Y = c_1, c_2, ..., c_k$ is the category of the instance, $i = 1, 2, ..., N$; the instance feature vector x;

Output:

class y to which instance x belongs

(1) In the training set T, the k point is closest to x and includes a field of k, denoted as $N_{k(x)}$, as determined by the distance metric for the point given.

(2) In $N_{k(x)}$, the y category is determined by the x ruling on classification (for example, majority voting):

$$y = argmax_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j), i = 1, 2, ..., N \tag{2.12}$$

In the above formula, I is an indication function, that is, when $y_i = c_j$, I is 1, otherwise I is 0. The special case of KNN is the case of $k = 1$, which is called the nearest neighbor algorithm. For the input instance point (feature vector) x, the nearest neighbor algorithm classifies the training data set with the x nearest neighbor as the class of x.

In the KNN algorithm, there are three popularly used distances, namely Manhattan distance, Euclidean distance and Minkowski distance (Wang et al. 2020). Let the feature space X be a n-dimensional real vector space $R_n, x_i, x_j \in X, x_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(n)})^T, x_j = (x_j^{(1)}, x_j^{(2)}, ..., x_j^{(n)})^T, L_p$ distance of x_i, x_j is defined as:

$$L_p(x_i, x_j) = (\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^p)^{\frac{1}{p}} \qquad (p \geq 1) \tag{2.13}$$

When $p = 1$, it is called Manhattan distance, and the formula is:

$$L_p(x_i, x_j) = (\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|) \tag{2.14}$$

When $p = 2$, it is called Euclidean distance:

$$L_p(x_i, x_j) = (\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^p)^{\frac{1}{p}} \tag{2.15}$$

When $p = \infty$, it is the maximum value of each coordinate distance, and the calculation formula is:

$$L_\infty(x_i, x_j) = max_l \left| x_i^{(l)} - x_j^{(l)} \right| \tag{2.16}$$

KNN algorithm is simple and easy to implement a method with no parameter estimation or no training required. It can provide high precision performance, being insensitive to outliers (individual noise data has little effect on the results). It is suitable for classifying rare events, especially for multi-classification problems, even better than SVM. However, KNN is a lazy learning method. It needs a lot of calculation to make predictions, which means it has high classification time consumption when the value of K is large. The output of classification is not interpretable so it is difficult to explain the results. In addition, KNN is not suitable to classify an imbalanced data-set.

Table 2.5 : Advantages and disadvantages of K-Nearest Neighbour

Advantages	**Disadvantages**
Can be used for non-linear classification	Lazy learning
Not sensitive with outlier data	The output is not interpretable
Classification based on limited neighboring samples	A lot of calculation
Able to process irrelevant features	High error with imbalanced data-set
Fit class domain cross samples	Category classification is not standardized

2.5.4 Logistic Regression

Logistic regression (LR) is the statistical model that employs a logistic function in its fundamental form, while many more difficult extensions exist, to model a binary dependent variable. A binary logistic model has a dependent variable mathematically with two potential values, such as pass or fail, shown by indicator variable with two values labelled "0" and "1". A machine learning model limits the decision function to a certain set of conditions. This set of qualifications determines the hypothesis space of the model. Of course, we also hope that this set of qualifications is simple and reasonable. The assumptions made by the LR model are:

$$P(y = 1|x; \theta) = g(\theta^{^T} x) = \frac{1}{1 + e^{-\theta^T * x}} \tag{2.17}$$

Where g(h) is the sigmoid function mentioned above, and the corresponding decision function is:

$$y^* = 1, \quad if P(y = 1|x) > 0.5$$

It is a general practice to choose 0.5 as the threshold. In the actual application, different thresholds can be selected. If the accuracy of the positive example is high, you can choose a larger threshold. If the recall is high, you can select a smaller threshold.

2.5.4.1 Parameter solving

After the mathematical form of the model is determined, the rest is how to solve the parameters in the model. A method commonly used in statistics is maximum likelihood estimation, which is to find a set of parameters such that the probability (probability) of our data is larger under this set of parameters. In a LR model, the likelihood can be:

$$L(\theta) = P(D|\theta) = \prod P(y|x; \theta) = \prod g(\theta^T x)^y (1 - g(\theta^T x))^{1-y} \tag{2.18}$$

The log likelihood can be:

$$l(\theta) = \sum ylog(g(\theta^T x)) + (1-y)log(1-g(\theta^T x)) \tag{2.19}$$

The average logarithmic loss on the entire data set：

$$J(\theta) = -\frac{1}{N} l(\theta) \tag{2.20}$$

In the logistic regression model, the maximized likelihood function and minimized log loss function are equivalent. There are several solutions for this optimisation problem, for example, gradient descent.

2.5.4.2 Regularization

When the parameters of the model are too many, it is easy to encounter the problem of over-fitting. There is a need to have a way to control the complexity of the model. The typical approach is to add regular terms to the optimisation goal and prevent over-fitting by penalizing excessive parameters:

$$J(\theta) = -\frac{1}{N} \sum ylog(g(\theta^T x)) + (1-y)log(1-g(\theta^T x)) + \lambda \left\| w \right\|_p \tag{2.21}$$

In general, take $p = 1$ or $p = 2$, which corresponds to L1 and L2 regularization respectively. L1 regularization tends to make the parameter 0, so the sparse solution can be generated. In practical application, the dimension of our data may be very high. L1 regularization is more widely used because it can produce sparse solutions.

2.5.4.3 Softmax

If y is not a value in $[0, 1]$ but a value in K categories, then the problem becomes a multi-class problem. There are two ways to deal with this type of problem: one is that we train a binary classifier (One-vs-all) for each category, when K categories are not mutually

exclusive, such as which category the user will buy. This method is suitable. If the K categories are mutually exclusive, for instance, $y = i$ means that y cannot take other values, such as the user's age range, Softmax regression is more appropriate in this case. Softmax regression is a direct extension of logistic regression in multiple classifications. The corresponding model can also be called Multinomial Logistic Regression. The model models the probability through the softmax function, as follows:

$$P(y = i|x, \theta) = \frac{e^{\theta_i^T x}}{\sum_j^k e^{\theta_j^T x}} \tag{2.22}$$

The decision function is:

$$y^* = argmax_i P(y = i|x, \theta) \tag{2.23}$$

Similarly, gradient descent or other high-order methods can be applied to solve this problem.

When classifying a large data-set, LR can rapidly give output because it has less calculation cost. It is barely affected by slight multicollinearities, even it can use regularization to solve strong multicollinearities such as L1 and L2 regularization methods. However, LR classifier is easy to over-fit and sensitive to missing data.

Table 2.6 : Advantages and disadvantages of Logistic Regression

Advantages	**Disadvantages**
Less calculation cost on large data-set	Easy over-fitting
Not particularly affected by slight multicollinearity	Sensitive with missing data
Can use Regularization to solve multicollinearity	weak performance on high-dimensional

2.5.5 Naive Bayesian

In machine learning, the naive Bayesian (NB) classifier is a series of simple probability classifiers based on the strong independent Bayesian theorem between hypothetical features. NB classifiers are very scalable and involve a series of linear parameters in the number of variables in a learning problem. Maximum likelihood can be trained by assessing a closed-form term, which requires linear time instead of a costly iterative approach used in many other classification kinds (Hand & Yu 2001).

Theoretically, the probability model classifier is a conditional probability model.

$$P(C|F_1, F_2, ..., F_n)$$

The independent categorical variable C has several categories, and the condition depends on several characteristic variables $F_1, F_2, ..., F_n$. But the problem is that if the number of features n is large or each feature can take a large number of values, the probability table is listed based on the probability model (Wang et al. 2019). Bayes' theorem has the following formula:

$$P(C|F_1, F_2, ..., F_n) = \frac{P(C)P(F_1, F_2, ..., F_n|C)}{P(F_1, F_2, ..., F_n)} \tag{2.24}$$

In practice, the molecular part of the fraction is the only element to be considered, because the denominator does not depend on C and the value of the characteristic F_i is given, so the denominator can be considered a constant. Such a molecule is equivalent to a joint distribution model.

Repeat the chain rule, which can be written as a conditional probability as follows:

$$
\begin{aligned}
&P(C|F_1, F_2, ..., F_n) \\
&\quad \propto P(C)P(F_1, F_2, ..., F_n|C) \\
&\quad \propto P(C)P(F_1, C)P(F_2, ..., F_n|C, F_1) \\
&\quad \propto P(C)P(F_1, C)P(F_2|C, F_1)P(F_3, ..., F_n|C, F_1, F2) \\
&\quad \propto P(C)P(F_1, C)P(F_2|C, F_1)...P(F_n|C, F_1, F2, ...F_n - 1)
\end{aligned}
$$

Assume that each feature F_i is conditionally independent for other features F_j, $j \neq i$. then:

$$P(F_i|C, F_j) = P(F_i|C)$$

For $i \neq j$, the joint distribution model can be expressed as (Wang et al.2019):

$$
\begin{aligned}
&P(C|F_1, F_2, ..., F_n) \\
&\quad \propto P(C, F_1, F_2, ..., F_n) \\
&\quad \propto P(C)P(F_1|C)P(F_2|C)P(F_3|C)... \\
&\quad \propto P(C)\prod_{i=1}^{n} P(F_i|C)
\end{aligned}
$$

This means that under the above assumption, the conditional distribution of the class variable C can be expressed as:

$$P(C|F_1, F_2, ..., F_n) = \frac{1}{Z}P(C)\prod_{i=1}^{n} P(F_i|C) \tag{2.25}$$

Where Z (evidence factor) is a scaling factor that depends only on $F_1, F_2, ..., F_n$, etc., and is a constant when the value of the characteristic variable is known. Due to the decomposition into the class prior probability $P(C)$ and the independent probability distribution $P(F_i|C)$, the controllability of the above probability model is greatly improved. If this is a k-

classification problem, and each $P(F_i|C = c)$ can be expressed as r parameters, then the corresponding simple Bayesian model has $(k-1) + nrk$parameters (Wang et al. 2019). Wang et al. (2019) also indicated that in practical applications, $k = 2$ (two-class problem) and $r = 1$ (Bernoli distribution are characteristic), so the number of parameters of the model is $2n + 1$, where n is the number of binary classification features.

NB is one of the classifiers that can provide stable classification efficiency. Although NB is not sensitive to missing data, it is sensitive with the representation of the input data. The precondition of classification implementation is needing prior probability.

Table 2.7 : Advantages and disadvantages of Naive Bayesian

Advantages	**Disadvantages**
Stable classification efficiency	Need prior probability
Not sensitive to missing data	Sensitive with the representation of input data
Good performance with small data-set	

2.6 Ensemble Methods

An ensemble model is a technique that utilizes various algorithms or combines them to enhance the strength and efficiency of any of the components' algorithms. Syarif et al. (2012) proposed that the benefit of ensemble methods is that they can be adjusted more appropriately than single models in any modifications in the controlled data stream. An ensemble classifier is more accurate than individual classification methods. The achievement of the ensemble strategy relies on the diversity of the individual classification systems in relation to erroneous situations. There are four methods to accomplish this diversity: i. to use various training data to form single classification systems. ii. to use a variety of training parameters. iii. to use a range of characteristics to train classifiers. iv. to combine distinct types of classifiers (Syarif et al. 2012).

2.6.1 Bagging-Random Forest

Bagging, which implies aggregation of the bootstrap, is one of the most simple but effective ensemble methods to enhance volatile issues of classification (Figure 2.8 (a)). This technique is generally used in the design of a DT and can be used for other classification algorithms, including NB, KNN, rule induction, etc. Bagging techniques are highly useful for large and high-dimensional data, for example, intrusion data-sets, where the problem is complex to identify a good model or classifier that can work in simple steps (Syarif et al. 2012).

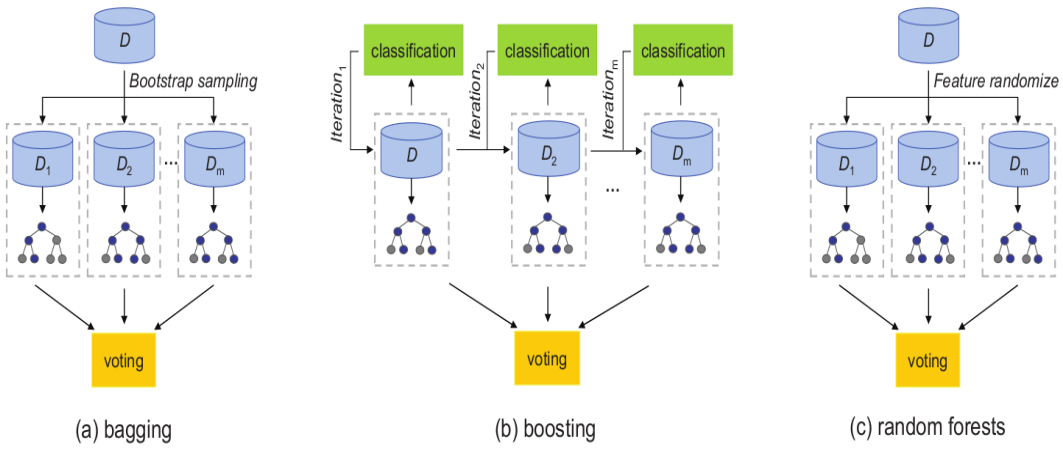


Figure 2.8 : Schematic illustration of bagging, boosting and random forest (Yang et al. 2016)

A random forest (RF) classifier is an ensemble classifier that generates a random set of training samples and variables, using numerous DT algorithms (progress shown in Figure 2.9). The ensemble method of RF is comparable to bagging methods. Bagging utilizes only the bootstrap to sample data, which means one sample can be repeatedly selected, but RF also randomly sample the data and randomly choose features to classify besides bootstrapping, preventing over-fitting and decreasing variance (Figure 2.8 (a) and (b)) (Belgiu & Drăgut 2016). The out-of-bag (OOB) error is a significant characteristic of RF. For some trees, each observation is an OOB observation, which means that it is not used

for its construction and can thus be seen as a collection of inner validation data on the trees. The RF's OOB error is merely the average error frequency acquired by the use of trees that are OOB when observations in the data-sets are predicted (Boulesteix et al. 2012). With this internal validation, the estimated error is less optimistic and is generally considered as a decent estimator of the error for independent data.

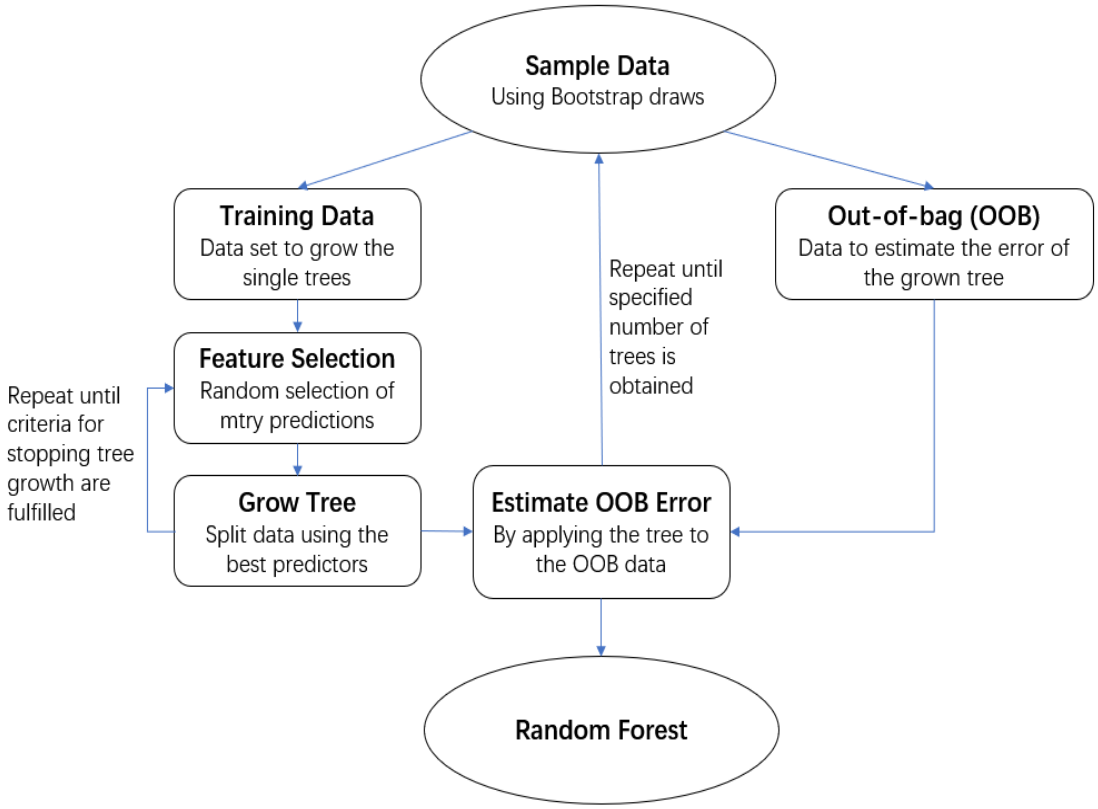


Figure 2.9 : The principle of Random Forest (Boulesteix et al. 2012)

RF is excellent for processing large and high dimensional data-sets, despite having missing data (See Table 2.8). For classification, there is no need to normalize the data-set since RF is a tree model. RF method is highly paralleled to be easily distributed implementation. When the classification has imbalance issues, RF can provide an effective approach to balancing the data-set errors. Furthermore, RF model able to determine the most important variables and show the significance of features, thus it also can be a method to reduce dimensionality. However, While solving the regression problem, the random

forest is not as effective as it does in classification, because it does not provide a continuous output. When regression is performed, random forests hardly predict outside of the range of training data, since it might cause over-fitting while certain data with particular noise is modeling. RF is a black box for many statistical modelers - the operation within the model can hardly be controlled and only between various parameters and random seeds can be attempted. Therefore, RF model can be simply established to solve my liver cirrhosis data-set and have enhancement where parameters are optimised.

Table 2.8 : Advantages and disadvantages of Random Forest

Advantages	**Disadvantages**
Excellent performance on high-dimensional data	Not able to make predictions in regression problem
Able to deal with missing data	Hard to control the operation
Able to balance data-set error	Ignore the correlation between attributes
Highly parallelized for easy distributed implementation	
No need to normalize data-set	

2.6.2 Boosting-Adaboost

Boosting is a machine learning strategy, based on the concept of generating an extremely precise classifier by combining several comparatively weak classifiers that are only slightly correlated with the true classification (Schapire 2013). Boosting provides the predictors with sequential learning. The former predictor learns from the entire data-set, while the following predictors learn from the training data-set based on the previous performance. The misclassified examples are marked and their weights are increased so that they are more likely to appear on the next predictor's training set (Syarif et al. 2012). In this paper, AdaBoost algorithm is selected, which is one of the most frequently used

boosting methods for building a powerful classification to be a linear combination of weak classifications.

The AdaBoost is the most popular Boosting method by creating a collection of classifiers of components by maintaining a set of weights over training samples, adapting them after each iteration: the weights of the training samples that are wrongly classified by the current component classifiers are increased whilst weights correctly classified are decreased (Li, Wang & Sung 2007; Lupascu, Tegolo & Trucco 2010). It means that AdaBoost takes a weighted majority vote, in particular, increased weight of weak classifiers with low classification error rates to play a bigger role in voting and reduced weight of weak classifiers with a higher classification error rate to a smaller role in voting.

In general, AdaBoost algorithm consists of the following three steps:

1) Initialize the weight distribution of the training set $T = \{(x_1, y_1), ..., (x_n, y_n)\}$,
$y_i \in \{+1, -1\}$:

$$D_1 = (\omega_{11}, \omega_{12}, ..., \omega_1 N), \omega_1 i = \frac{1}{N}, i = 1, 2, ..., N$$

2) Train weak classifiers. If a sample has been specifically classified, its weight in the construction of the following training set shall be reduced in the particular training process; if, however, a sample point is not correctly classified, its weight shall be increased. Then the up-to-date weight sample is used to train the next classifier, which gradually increases the classification during the iterative process if the entire training process is iteratively performed.
In each iteration $t = 1, 2, ..., T$, select a weak classifier h with the lowest error rate as the $t - th$ base classifier H_t, and calculate the weak classifier h_t (Nie et al. 2018). The error of the weak classifier on the distribution D is:

$$e_t = P(H_t(x_i) \neq y_i) = \sum_{i=1}^{N} w_{ti} I(H_t(x_i) \neq y_i) \tag{2.26}$$

Indicate the weight of this weak classifier:

$$\alpha_t = \frac{1}{2} ln(\frac{1 - e_t}{e_t}) \tag{2.27}$$

When $e_m \leq \frac{1}{2}, \alpha_m \geq 0$, α_m increases with the reduction of e_m, which means that the less error in the classification, the greater the effect of the basic classifier in the final classifier.

Then update the weight of distribution for the next iteration:

$$D_{t+1} = \frac{D_t(i) exp(-\alpha_t y_i H_t(x_i))}{Z_t} \tag{2.28}$$

where $Z_t = 2\sqrt{e_t(1 - e_t)}$ is a normalization factor so that D_{t+1} is a distribution.

3) The classifiers have weak performance obtained from each training are combined into a strong classifier. After the path of the weak classification classifier has finalised, weights are increased of the weak classifiers with a small error rate, which are more important in deciding the final classification feature, then the weak classifier with a large error classification rate is reduced. Therefore, in the final ranking, a poor classifier with a low error rate has a greater weight, otherwise, it is smaller. Based on the weight α_t from each iteration to combine those weak classifiers:

$$f(x) = \sum_{t+1}^{T} \alpha_t H_t(X) \tag{2.29}$$

The final classifier is as follow:

$$H_final = sign(f(x)) = sign(\sum_{t+1}^{T} \alpha_t H_t(X)) \tag{2.30}$$

AdaBoost is easily implemented in classification cases. It requires a few parameters to adjust the classifier performance. Using AdaBoost to make classification is not easy to

over-fit, since it includes a number of sub-classifiers to predict outputs then using voting or averaging method to select the final result which can reduce the risk to be over-fitting. However, AdaBoost is sensitive to noise and unable to ensure the output whether it is optimised or not.

Table 2.9 : Advantages and disadvantages of AdaBoost

Advantages	**Disadvantages**
Easy implementation	Unable to ensure the optimised output
A few parameters adjustment	Sensitive with noise
Not easy to over-fit	
No need for feature selection	

2.6.3 Stacked Generalisation

Stacking is an ensemble learning, which refers to the process of the model automatically extracting valid features from the original data. Stacking and neural networks are similar in some ways, and neural networks can also be seen as an ensemble learning. The learning capacity of Stacking comes mainly from feature representation, which is compatible with the idea of neural networks. The first layer in Stacking can be equal to the first layer in the neural network, and the final Stacking classification layer can be analogised to the last neural network output layer (Syarif et al. 2012). The difference is that different classifiers in Stacking represent heterogeneous representations of different features. Neural networks are processes from homogeneous to heterogeneous and have distributed representations. There should also be distributed features in Stacking, mainly because the results of multiple classifiers are not completely different, but a large degree of similarity. The stacking ensemble framework has two recommendations for the base classifier to have better performance: greater diversity and higher accuracy.

Stacked generalisation or Stacking is a method to combine new features from multiple classifiers and output more precise predictions. In contrast to bagging or boosting, Stacking is generally used to combine multiple various classifiers, such as decision tree, neural network, inductive rule induction, naive bays, logistic regression, etc (Syarif et al. 2012). In general, two layers are sufficient for Stacking. Multi-layered Stacking faces more complex over-fitting problems with limited benefits. For example, a stacking model consists of two layers: level-0 and level-1 stacking layers. The base layer (level-0) uses a wide range of models to learn from a data-set. The outputs of each model are collected to generate a new data-set. Each feature or sample in the new data-set is associated with the real value. The new data-set is then inputted to the stacking model layer level-1 to predict the final output. During the process of generating meta-features from layer level-0, K-fold cross-validation method is desirable to split the training set in order to reduce the impact of over-fitting.

For example (Figure 2.10), the original training set is divided into 4 folds, which are labeled $fold_1, fold_2, fold_3$&$fold_4$. In layer level-0, $fold_1$ is predicted and others are trained in each classifier (LR and NB). The output of predictions are being the meta-features training in layer level-1. Similarly and until all folds training and prediction complete to generate meta-features. In addition, each classifier also makes a prediction of the original testing set to generate a new testing set for layer level-1 meta-classifier. It should be noted that when generating the second layer feature, each base model should use the same K-fold, and each fold of the obtained meta feature (corresponding to the previous K fold) will not leak into the fold data so that to minimize the risk of over-fitting.

Theoretically, assuming a data set $A = \{(x_n, y_n), n = 1, ..., N\}$, and using K-fold cross-validation to split the data into K almost equal parts $A_1, ..., A_K$. Define A_k and $A^{(-k)} = A - A_k$ to be the testing and training sets for the k^{th} fold. Given M classifiers, which are level-0 models, invoke m^{th} classifier on the data in the training set $A^{(-k)}$ to induce a model $G_m^{(-k)}$, for m=1,...,M (Wang & Zhao 2008).

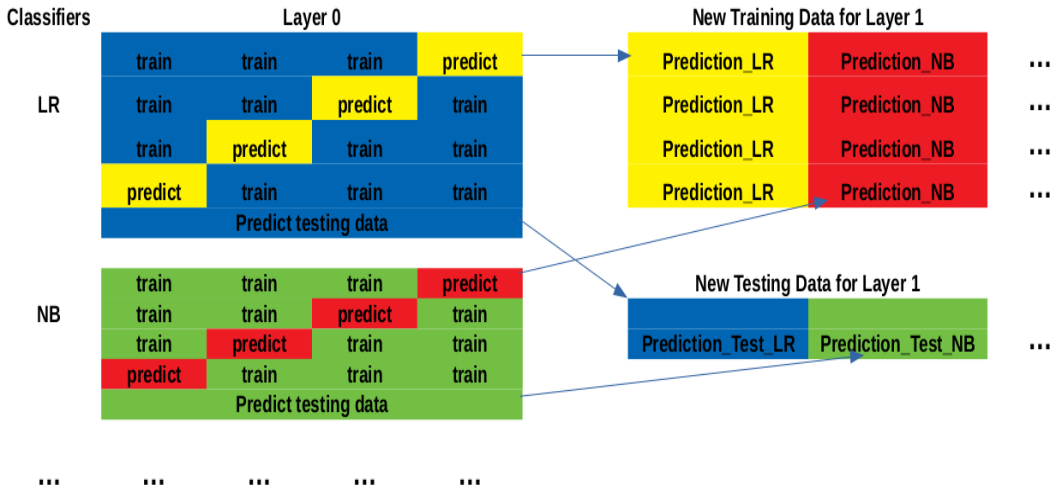


Figure 2.10 : Illustration of a Stacking Model with Logistic Regression and Naive Bayesian classifiers

for each x in A_k, let the testing set under the k^{th} cross-validation fold, and v_m^{-k} denote the prediction of the model $G_m^{(-k)}$ on x:

$$z_{mn} = v_m^{-k}(x_n) \tag{2.31}$$

After completing the entire cross-validation process, the data combined from the outputs of the M classifiers is:

$$A_{CV} = \{(z_{1n}, ..., z_{Mn}, y_n), n = 1, ..., N\} \tag{2.32}$$

This is the data for layer level-1 classifier to train a model $\widetilde{G}$. Now given a new data-set B, model $G_m^{(-k)}$ makes predictions to produce a vector $z' = (z_1, ..., z_m)$ for layer level-1 model $\widetilde{G}$ as the testing set. Therefore, A_{CV} and z' are the input to the layer level-1 model $\widetilde{G}$ whose output is the final classification result for data-set B (Wang & Zhao 2008).

Chapter 3

Methodology

This chapter will present the features of the liver stiffness measurement data-set. The features will be pre-processed with scaling transformation and demonstrated with sampling methods to balance the data distribution. Then, we will introduce the evaluation criteria to exam the classification performance. Finally, we will present the parameters setting of algorithms involved in the experiment and how to build the stacking models to compare performance.

3.1 Liver Stiffness Measurement Data-set

There is one liver stiffness measurement (LSM) data-set used to evaluate the performance of the stacking model. The whole data-set was collected by Shen Zhen Yi Ti hospital, including 13,418 participating patients' FibroScan testing records with second opinions recorded in Chinese characters by professional doctors from 55 hospitals in different cities of China, between the year 2015 and 2018. Thirty variables were chosen as the criteria for the evaluation. These variables cover patients' personal body information, and liver conditions, and are listed in Table 3.1. The second opinions in Chinese characters included in the data-set and the distribution of each class instances are shown in Table 3.2. In addition, in Table 3.3, the mean and variance values of part of features of LSM data-set are shown.

3.1.1 Data Pre-processing

While Chinese characters are not able to be analysed in Python, Jieba module in Python can translate those second opinions to fixed numbers which are treated as labels in the experiments. After translation, there were 6 labels in total. Then all the data features

Table 3.1 : Data-set variables description

Variables	
Patients' information	BMI, Age, Height
	Weight and Gender
FibroScan estimation	Liver stiffness measurements
	Liver fat measurements
	Average liver stiffness measurement
	Average liver fat measurement
	Liver stiffness measurement IQR
	Liver fat measurement IQR
	Liver pressure

Table 3.2 : Second opinion in Chinese character and the output of Jieba module in Python

Second opinions in Chinese character	Jieba output	Labels	Amounts
肝脏弹性值正常（正常范围参考值 1.0-7.0kpa）	8.27094	0	7,236
肝纤维化 F0-F1，建议定期检查	9.88895	1	4,320
肝纤维化 F1-F2，建议 3 个月后复查	8.55347	2	494
肝纤维化 F2-F3 阶段，需结合临床检查综合判断	8.14162	3	280
肝脏硬度值偏高，正常参考值范围为（1.0-7.0kpa）	8.03283	4	816
肝硬化 F3-F4，ALT 正常不排除肝硬化可能	8.30539	5	272

x_i were standardized by the formula:

$$z_i = \frac{(x_i - u)}{s} \tag{3.1}$$

where u is the mean value of x_i and s is the standard deviation of x_i.

The purpose of scaling transformations for different feature dimensions is to make features between different metrics comparable. At the same time, the distribution of the original data is not changed.

Table 3.3 : The mean and variance values of part of features of liver stiffness measurement data-set

Features	Class 0	Class1	Class 2	Class 3	Class 4	Class 5
BMI (Mean)	23.77	24.79	24.24	23.67	23.04	24.11
BMI (Variance)	10.63	9.38	17.55	18.73	15.02	18.39
AGE (Mean)	31.32	49.35	50.80	50.39	46.25	49.74
AGE (Variance)	156.20	176.01	202.89	192.72	200.88	158.92
HEIGHT (Mean)	166.18	166.39	166.03	164.99	165.43	166.29
HEIGHT (Variance)	59.18	44.38	51.74	72.92	63.21	71.57
WEIGHT (Mean)	65.81	68.78	66.93	64.62	63.24	66.92
WEIGHT (Variance)	123.54	107.67	165.58	179.59	156.40	191.41
Average LSM (Mean)	5.34	4.78	8.04	10.65	11.89	22.07
Average LSM (Variance)	1.18	1.61	1.64	15.08	70.18	121.30

EasyEnsemble undersampling method and SMOTE oversampling method were applied to the imbalanced LSM data-set to balance the classes, respectively. The output of those sampling methods would be the input of the classification system.

In the experiments, the data-set was divided into training data-set and testing data-set by using K-fold cross-validation, shown in Section 2.6.3.

3.2 Evaluation Criteria of Classification Performance

The evaluation criteria of the experiments are confusion matrix, the established standard measurements evaluate the performance of a classifier model, and receiver operating characteristic curve (ROC curve) to observe the relationship of classifier sensitivity and specificity. The classification accuracy is determined to demonstrate the ratio of correct predictions made by the classifier. The confusion matrix outlines the predictions can be demonstrated in Table 3.4.

The True Positive (TP) and the True Negative (TN) are the positive and negative objects precisely classified into the positive and negative classes, respectively (Pruengkarn

Table 3.4 : Confusion matrix of binary classification tasks

	Predicted Positive	Predicted Negative
Real Positive class	True Positive (TP)	False Negative (FN)
Real Negative class	False Positive (FP)	True Negative (TN)

et al. 2015). The False Positive (FP) presents the negative objects classified as positive, while the False Negative (FN) presents the positive objects classified as negative. To evaluate the accuracy *Acc* of a classifier, a formula is computed below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

where the sum of all True predictions is divided by the total number of objects in the data-set. A high *Acc* stands for a good classifier, but it is not favorable for classifying an imbalanced data-set. For example, in a data-set with 90% of objects belonging to Class 1 and 10% of objects belonging to Class 2, the classifier can simply classify all the objects as Class 1 to achieve 90% accuracy and ignoring all the objects in Class 2. If Class 2 obtains the most important information for the classification task, evaluating a model performance by only calculating the accuracy is not sufficient due to the great possibility of missing actual meaningful objects. Therefore, to properly evaluate models' performance on classification tasks with an imbalanced data-set, other metrics are usually applied.

The measurement $F_\beta - Score$ is commonly demonstrated to evaluate classifiers dealing with imbalanced data-sets (Japkowicz & Shah 2011). These metrics calculate the evaluation score by utilizing the Precision and Recall of classifiers, which can represent the classification performance on not only the majority class but also the minority class in an imbalanced data-set classification.

Precision indicates the ability of a classifier on resulting in more relevant than irrelevant objects. This measure is calculated by the number of correctly predicted objects

(True Positives) divided by the total number of predictions labeled as the positive class (True Positives and False Positives).

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

Recall, also known as Sensitivity, represents the ability of a classifier on predicting the relevant objects in the universe class. This measure is calculated by the number of correctly predicted objects (True Positives) divided by the total number of predictions belonging to the positive class (True Positives and False Negative).

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

Specificity is another measure to estimate the classifier performance based on the confusion matrix. It indicates the accuracy with which the algorithm classifies negative samples in data classified as negative. The greater the specificity, the more accurate the classification algorithm is in classifying true negative samples.

$$Specificity = \frac{TN}{TN + FP} \tag{3.5}$$

The above formulas are for binary classification tasks, but most reality tasks are multi-class classification. Therefore, the Precision, Recall and Specificity measurements of each class can be calculated as the following:
An example of confusion matrix with 3 classes classification is shown in Figure 3.1. In this case, the Precision and Recall of class 0 will be:

Precision:

$$Precision_{class0} = \frac{a}{a + d + g} \tag{3.6}$$

Recall:

$$Recall_{class0} = \frac{a}{a+b+c} \tag{3.7}$$

Specificity:

$$Specificity_{class0} = \frac{e+f+i+h}{d+e++f+g+h+i} \tag{3.8}$$

For class 1 and class 2, TP is e and i, respectively. The calculation methods of Precision and Recall are similar to class 0. The Accuracy of the classification is:

$$Accuracy = \frac{a+e+i}{a+b+c+d+e+f+g+h+i} \tag{3.9}$$

Confusion Matrix		Predict		
		0	1	2
Actual	0	a	b	c
	1	d	e	f
	2	g	h	i

Figure 3.1 : Example of confusion matrix in multi-class classification tasks

$F\beta - Score$ combines the results of Precision and Recall's output. The physical meaning of $F\beta$ is a weighted average of Precision and Recall. The weight of the Recall is times Precision.

$$F\beta - Score = \frac{(1+\beta^2) * Recall * Precision}{\beta^2 * Precision + Recall} \tag{3.10}$$

The $F1 - Score$ considers Recall as important as Precision. It is defined as the harmonic mean of Precision and Recall. The value of $F1 - Score$ is between 0 and 1, with

1 as the best model result and 0 as the worst model output (Guo et al. 2019).

$$F1 - Score = \frac{2 * Recall * Precision}{Precision + Recall} \tag{3.11}$$

The $F2 - Score$ presents that Recall is twice as important as Precision. Recall is more important, that is, the ability of the model to recognize positive samples.

$$F2 - Score = \frac{5 * Recall * Precision}{4 * Precision + Recall} \tag{3.12}$$

Receiver Operating Characteristic curve (ROC curve) is a comprehensive index that reflects the continuous variables of sensitivity and specificity (Huang 2015). It uses composition to reveal the correlation between sensitivity and specificity. This sets a number of thresholds for continuous variables in the measurement of a range of Sensitivity and Specificity, and then draws the curve with sensitivity as the ordinate and (1-specificity) as the abscissa. On the ROC curve, the critical value with greater sensitivity and precision is the point closest to the left top of the image.

For binary classification tasks, confusion matrix is presented in Table 3.4. True Positive Rate (TPR), describes the ratio of positive instances correctly identified by the classifier to all instances. False Positive Rate (FPR), describes the ratio of negative instances incorrectly identified by the classifier to all instances, and True Negative Rate (TNR) can be calculated for the ROC curve (Wu et al. 2019):

$$TPR = \frac{TP}{TP + FN} \tag{3.13}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.14}$$

$$TNR = \frac{TN}{FP + TN} = 1 - FPR \tag{3.15}$$

In a binary classification model, for the continuous results obtained, it is assumed that a threshold value has been determined, for example 0.6, instances greater than this value are classified as positive, and less than this value are classified as negative. If the threshold reduces to 0.5, it can identify more positive classes, that is, increase the ratio of the identified positive instances which is TPR, but also classify more negative instances as positive, which means that the FPR is improved.

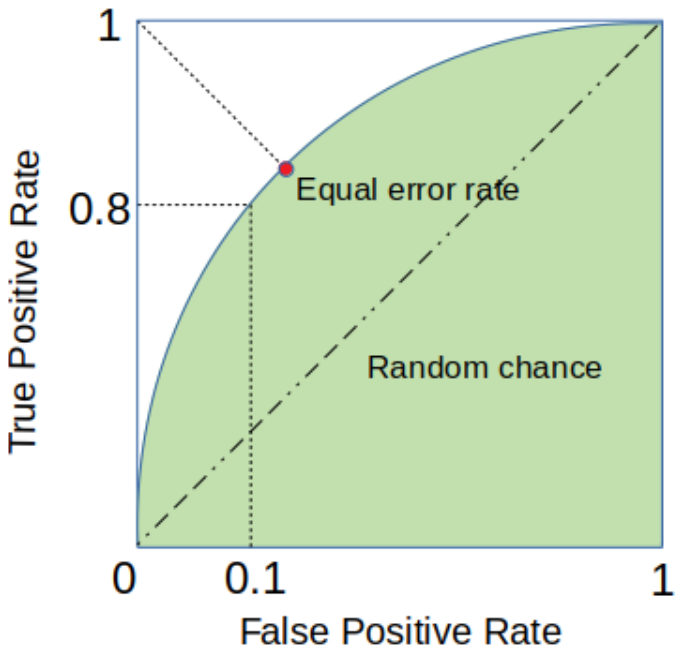


Figure 3.2 : Example of ROC curve in binary classification tasks

In Figure 3.2, each point on the ROC curve corresponds to a Threshold. For a classifier, there will be a TPR and FPR under each Threshold. For example, at the maximum Threshold, $TP = FP = 0$, which corresponds to the origin; At the minimum Threshold, $TN = FN = 0$, it corresponds to the point (1, 1) in the upper right corner.

The Area Under Curve (AUC) is the area under the ROC curve. Basically, the model is better when the curve is closer to the upper left corner (TPR = 1, FPR = 0) (Huang 2015). The AUC value is used as the criterion for assessment since the ROC curve normally does not show which classifier performs better but a greater AUC classifier performs more

effectively.

$$\begin{cases} AUC = 1, \quad Perfect\ classifier \\ 0.5 \leq AUC \leq 1, \quad Effective\ prediction\ value \\ AUC = 0.5, \quad Random\ prediction \\ AUC < 0.5, \quad Worse\ than\ random\ prediction \end{cases}$$

The reason for using ROC curve is, even when the distribution of positive and negative cases in the test set varies, the ROC curve remains unchanged. The class imbalance phenomenon often occurs in the actual data-sets, which means that the distribution of positive and negative instances in the test data can also change over time, compared with positive instances (Xie et al. 2019).

To solve multi-class classification tasks, ROC curve can be created to obtain the AUC value of each class to observe the classifier performance in each class. For example, let instances in class 0 as positive, instances in other classes as negative, it simply changes the objective to binary classification (Other classes do the same process). Based on the method above, the ROC curve and the AUC value can be easily obtained for all classes.

3.3 Experiment Procedure

First of all, seven supervised learning algorithms, Logistic Regression (LR), Decision Tree (DT), Naive Bayesian (NB), K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Random Forest (RF) and AdaBoost, were implemented respectively in Python to classify the LSM data-set. *sciki-learn* module was provided in Python to implement these classifiers because of its simple implementation and easy-monitored structure. Parameter set-up was necessary for the experiment. LR classifier was setting up with L2 regularisation as penalty and using 'liblinear' as solver. DT classifier was using CART algorithm for the classification and all parameters were set as default. Parameters of NB classifier are also set as default values. KNN classifier was implemented with 5 'neighbour'. SVM classifier

was using RBF kernel with parameter C, γ setting as default. RF and AdaBoost were ensemble methods whose parameters are set as default.

Secondly, the stacking model structure was building up after completing all classifications individually. Based on the performance of each classifier, several classifiers were selected with high accuracy plus balanced precision-recall values to keep diversity and stability, being the base classifiers in layer level-0 of the stacking model. In layer level-1, all classifiers had the opportunity to be the meta-classifier for the final classification (see an example of a stacking model in Figure 3.3).

Thirdly, the data-set was split with 80% training data and 20% testing data, then the training data was used with 10-fold cross-validation to train the base classifiers and collect all the predictions to be a new training data for the meta-classifiers. Besides, the base classifiers make predictions on the testing data to obtain a new testing data for the final classification.

Finally, all experimental results were evaluated by the evaluation criteria and based on the outputs from evaluation criteria to have a conclusion and future work is provided.

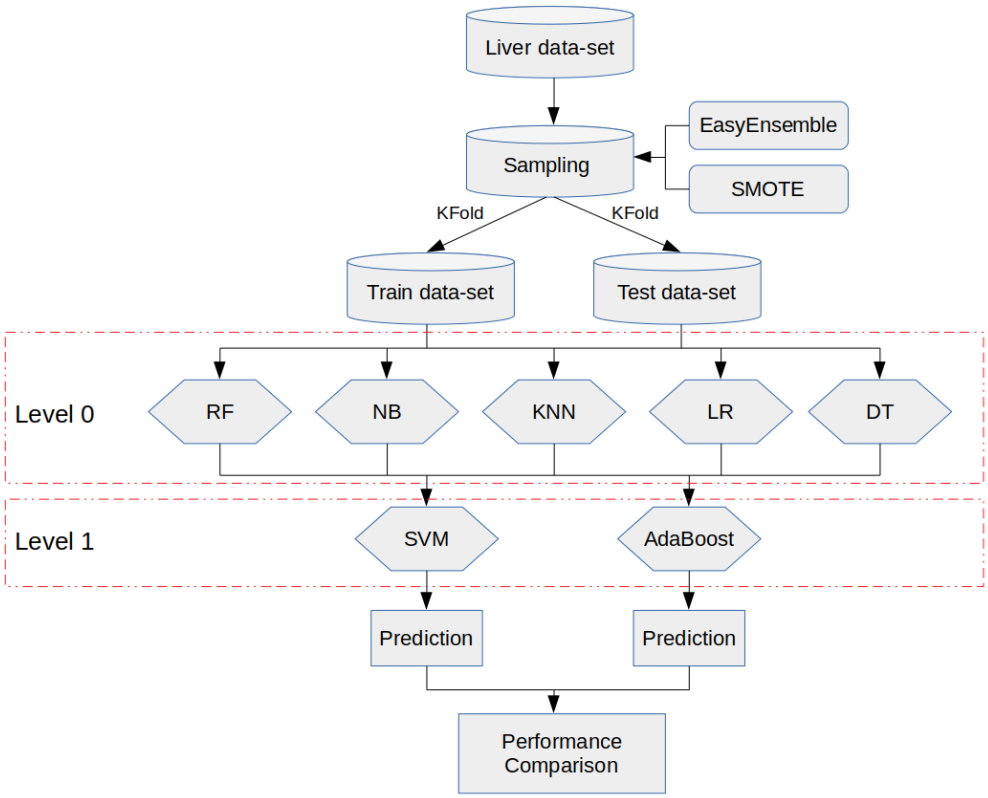


Figure 3.3 : Example of a stacking model assuming RF, DT, LR, KNN and NB as base classifiers in Level-0, SVM and AdaBoost as meta-classifiers in Level-1

Chapter 4

Experiment Results and Discussion

In this chapter, we will show two experiment results: classification performance of popular simple algorithms (KNN, NB, LR, DT, SVM), a bagging algorithm (RF) and a boosting algorithm (AdaBoost), and classification performance of stacking models. Then we will discuss to select the best model for classifying the liver stiffness measurement (LSM) data-set by comparing all classification performance.

4.1 Equipment

The experiments described in this section were performed on PC with 3.20 GHz Core i7 in 8^{th} generation CPU and 8 Gb RAM, using Linux operating system. Spyder (Python 3.7 version) in Anaconda 3.0 was used for programming classification. Anaconda is an open-source software that consists of collections of machine learning algorithms by downloading modules such as *sciki-learn* and a NLP module Jieba which is solving Chinese characters issues in the experiments.

4.2 Experiment Results and Discussion

There are two experiments included: implementation performance of popular simple algorithms, a bagging algorithm and a boosting algorithm, and implementation performance of the stacking models with different meta-classifiers will be presented after choosing the best base classifiers for the stacking models. The LSM data-set has 6 classes, the classification quality of each class were covered. All results were evaluated by the evaluation criteria proposed in Section 3.2, including **Accuracy**, **Precision**, **Recall**, **Specificity**, **F1-score** and **F2-score**. Each class should have less classified difference and a low value

of evaluation is not acceptable. To minimize the non-expecting effects of the variability of the training data, the experiments were repeated 10 times. At each iteration, all classifiers were trained on the same training partition of the LSM data-set with 10-fold cross-validation method, which will split the data-set with 70% as training data-set and 30% as testing data-set. All parameters were indicated in Section. 3.3.

4.2.1 Simple Algorithms, Bagging and Boosting Algorithms Performances

For the task addressed in my research, five algorithms such as KNN, NB, LR, DT and SVM, which are popular and acting successful role on solving classification problem recently, and Bagging algorithm RF and Boosting algorithm AdaBoost were implemented in this experiment. Each algorithm was presented with its performance confusion matrix, the evaluation value of all criteria and the ROC curve showing how valuable the classifier performance is. Also, each algorithm experimented on undersampled and oversampled data-set based on the original LSM data-set. The goal of this experiment was to choose an effective sampling method to balance the imbalance LSM data-set and select several classifiers with high quality of performance to be the base classifiers in the stacking model.

4.2.1.1 KNN Algorithm

In Table 4.1, this is the KNN algorithm performance on the **original** LSM data-set. Due to the high accurately classified rate of Class 0, the Accuracy of KNN on **original** data-set reaches 86.41%. Although Class 0 instances obtains with almost all evaluation over 90%, the Specificity of Class 0 is the lowest (86.62%), which means the successful classified rate of other classes is low. Obviously, Class 3 with the lowest Precision (44.39%), Recall (29.64%), F1-score (35.55%) and F2-score (31.75%), which means most of Class 3 instances could not be accurately classified. In Table 4.2, it is the KNN algorithm performance on the **EasyEnsemble undersampled** LSM data-set. The classification performance of Class 3 under undersampled data-set has slightly improved around 20% in average, but

the rates of Class 4 drop to 49.23% (Precision), 38.03% (Recall), 42.50% (F1-score) and 39.70% (F2-socre). Although the mean value of Specificity of all classes can reach up to 93.61%, the Accuracy of KNN on **undersampled** data-set decreases to 68.06%, which is worse than the performance on the original data-set. In Table 4.3, this is the KNN algorithm performance on the **SMOTE oversampled** LS data-set. All evaluation values have improved to a higher level, the lowest value of Precision rate, Recall rate, Specificity, F1-score and F2-score increase 49.35%, 57.99%, 12.38%, 55.43% and 57.19%, respectively. The Accuracy of KNN algorithm on **oversampled** data-set is 96.01%, which is 9.6% higher than the original, 27.95% higher than the undersampled. In Figure 4.1, in the original case (a), most of the instances belong to Class 0. Although almost all Class 0 instances are successfully classified, a numbers of instances belonging to other classes are misclassified, which leads high Accuracy but low reliability. In (a) and (b), all classes instances are balanced, the undersampled case is obviously worst than the original case according to the AUC value in the ROC curve. The AUC values of each class in oversampled case almost reach 1 which means the model is desirable for the classification.

Table 4.1 : Performance obtained by KNN classifier on original LSM data-set

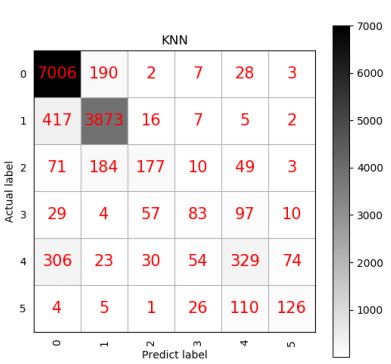
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	89.44	96.82	**86.62**	92.98	95.25
1	90.51	89.65	95.54	90.08	89.92
2	62.54	35.83	99.18	45.56	39.18
3	**44.39**	**29.64**	99.21	**35.55**	**31.75**
4	53.24	40.32	97.71	45.89	42.38
5	57.80	46.32	99.30	51.43	48.24
Mean	66.32	56.43	96.26	60.25	57.77
SD	(+/- 17.60)	(+/- 26.58)	(+/- 4.51)	(+/- 22.62)	(+/- 25.11)
Accuracy			**86.41** (+/- 0.34)		
Weighted AUC			**0.9467**		

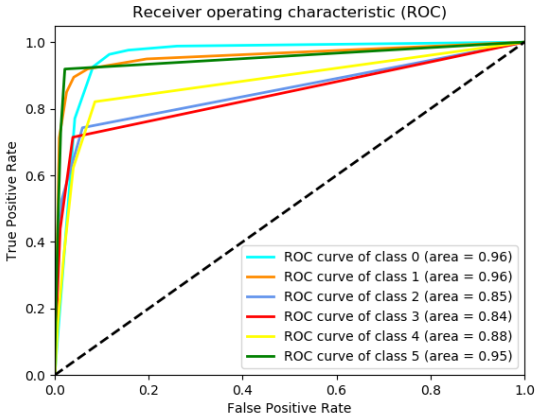
Table 4.2 : Performance obtained by KNN classifier on undersampled LSM data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	76.23	88.74	94.46	82.00	85.91
1	84.26	82.91	96.89	83.56	83.16
2	63.78	67.67	92.29	65.64	66.83
3	55.30	57.61	**90.67**	56.40	57.11
4	**48.23**	**38.03**	91.85	**42.50**	**39.70**
5	76.56	73.37	95.51	74.90	73.97
Mean	67.39	68.06	93.61	67.50	67.78
SD	(+/- 12.74)	(+/- 16.77)	(+/- 2.18)	(+/- 14.57)	(+/- 15.85)
Accuracy			**68.06** (+/- 0.30)		
Weighted AUC			**0.8952**		

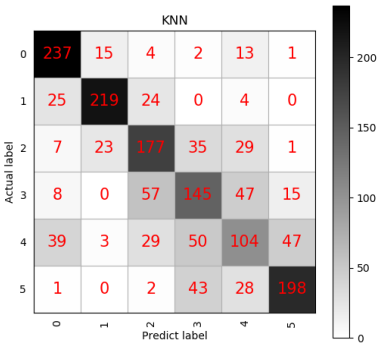
Table 4.3 : Performance obtained by KNN classifier on oversampled LSM data-set

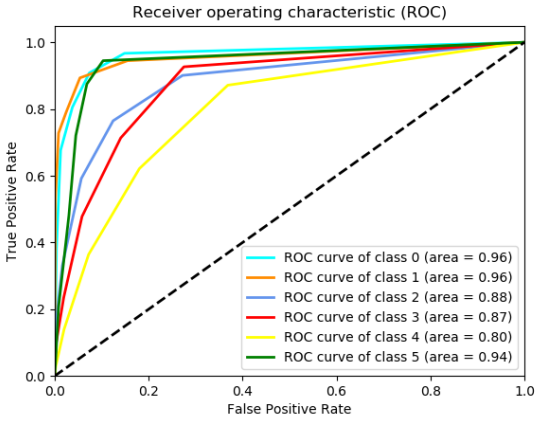
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	94.59	**87.63**	**99.00**	**90.98**	**88.94**
1	95.30	90.13	99.11	92.64	91.12
2	94.79	99.79	98.90	97.23	98.75
3	98.43	99.70	99.68	99.06	99.44
4	**93.74**	98.92	98.68	96.26	97.84
5	99.25	99.92	99.85	99.58	99.79
Mean	96.02	96.02	99.20	95.96	95.98
SD	(+/- 2.06)	(+/- 5.11)	(+/- 0.42)	(+/- 3.17)	(+/- 4.40)
Accuracy			**96.01** (+/- 0.13)		
Weighted AUC			**0.9923**		



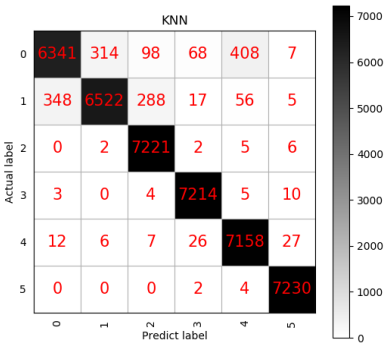


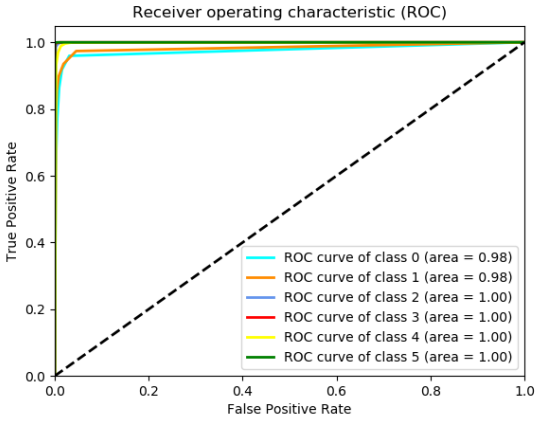
(a)





(b)





(c)

Figure 4.1 : Confusion matrix and ROC curve of KNN classifier on (a) Original LSM data-set (b) EasyEnsemble undersampled LSM data-set (c) SMOTE oversampled LSM data-set

4.2.1.2 RF Algorithm

Table 4.4 shows the RF algorithm performance on the **original** LSM data-set. Similar to KNN algorithm, the high accurately classified rate of Class 0 leads the Accuracy of RF on **original** data-set to be 89.04%. Class 3 with the lowest Precision (48.21%), Recall (43.21%), F1-score (45.57%) and F2-score (44.13%). In Table 4.5, this is the RF algorithm performance on the **EasyEnsemble undersampled** LSM data-set. The classification performance of Class 3 under undersampled data-set has slightly improved, but the rates of Class 4 decreased. The Accuracy of RF on **undersampled** data-set drops from 89.04% to 78.92%. In Table 4.3, this is the RF algorithm performance on the **SMOTE oversampled** LSM data-set. The mean value of all evaluation have been enhanced to over 95.99% and the Accuracy of RF algorithm on **oversampled** data-set is 95.73%, which is 6.69% higher than the original, 16.81% higher than the undersampled. In Figure 4.2, the ROC curve of the oversampled case has become perfect which is much better than the original and undersampled cases.

Table 4.4 : Performance obtained by RF classifier on original LSM data-set

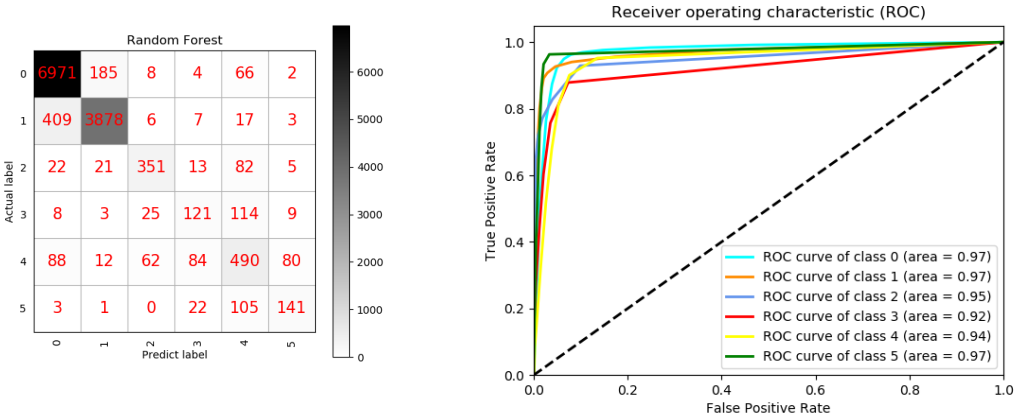
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	92.93	96.34	**91.43**	94.60	95.64
1	94.59	89.77	97.56	92.12	90.69
2	77.75	71.05	99.22	74.20	72.28
3	**48.21**	**43.21**	99.01	**45.57**	**44.13**
4	56.06	60.05	96.95	57.99	59.21
5	58.75	51.84	99.25	55.08	53.09
Mean	71.37	68.71	97.24	69.93	69.17
SD	(+/- 18.14)	(+/- 19.25)	(+/- 2.74)	(+/- 18.60)	(+/- 18.97)
Accuracy			**89.04** (+/- 0.50)		
Weighted AUC			**0.9663**		

Table 4.5 : Performance obtained by RF classifier on undersampled LSM data-set

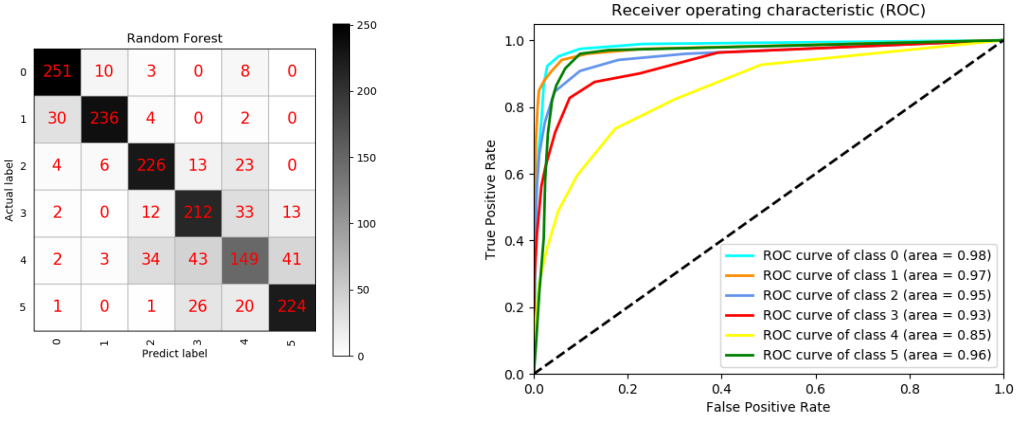
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	86.68	90.28	97.22	88.43	89.53
1	90.54	88.40	98.15	89.44	88.81
2	78.83	83.62	95.49	81.13	82.60
3	71.81	79.40	**93.76**	75.39	77.74
4	**62.98**	**48.82**	94.28	**54.94**	**51.08**
5	79.31	82.42	95.69	90.92	81.77
Mean	78.36	78.82	95.76	78.36	78.59
SD	(+/- 9.11)	(+/- 13.90)	(+/- 1.53)	(+/- 11.51)	(+/- 12.96)
Accuracy			**78.92** (+/- 0.17)		
Weighted AUC			**0.9335**		

Table 4.6 : Performance obtained by RF classifier on oversampled LSM data-set

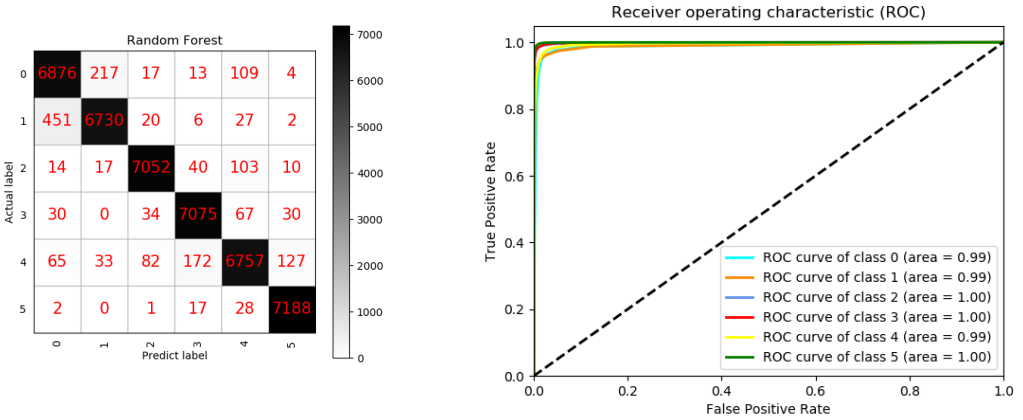
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	**92.44**	95.02	**98.45**	**93.71**	94.49
1	96.18	**93.01**	99.26	94.57	**93.63**
2	97.86	97.46	99.57	97.66	97.54
3	96.61	97.78	99.31	97.19	97.54
4	95.29	93.38	99.08	94.33	93.76
5	97.65	99.34	99.52	98.49	99.00
Mean	96.00	96.00	99.20	95.99	95.99
SD	(+/- 1.81)	(+/- 2.35)	(+/- 0.037)	(+/- 1.85)	(+/- 2.11)
Accuracy			**95.73** (+/- 0.023)		
Weighted AUC			**0.9948**		



(a)



(b)



(c)

Figure 4.2 : Confusion matrix and ROC curve of RF classifier on (a) Original LSM data-set (b) EasyEnsemble undersampled LSM data-set (c) SMOTE oversampled LSM data-set

4.2.1.3 NB Algorithm

As the NB algorithm performance on the **original** LSM data-set can be seen in Table 4.7. NB algorithm is difficult to predict Class 4, which only has 6.25% Recall rate, 10.52% F1-score and 7.46% F2-score. In Table 4.8, it is the NB algorithm performance on the **EasyEnsemble undersampled** LSM data-set. The capability of classifying Class 4 under undersampled data-set is weakened, Precision rate reduces to 18.85%, Recall rate to 5.08%, F1-score to 7.93%, F2-score to 5.93%. The Accuracy of NB on **undersampled** data-set drops from 81.87% to 61.28%. In Table 4.9, it is the NB algorithm performance on the **SMOTE oversampled** LSM data-set. The performance of NB algorithm is failed to improve, unlike KNN and RF. Class 4 is still not be able to successfully classified. The Accuracy of RF algorithm on **oversampled** data-set is 63.94%, which is 17.93% lower than the original, 2.66% higher than the undersampled. In Figure 4.3, according to the ROC curves of two sampled cases, the AUC values of Class 2, 3, 4 decrease to a very low level. Therefore, both sampled cases are worst than the original case.

Table 4.7 : Performance obtained by NB classifier on original LSM data-set

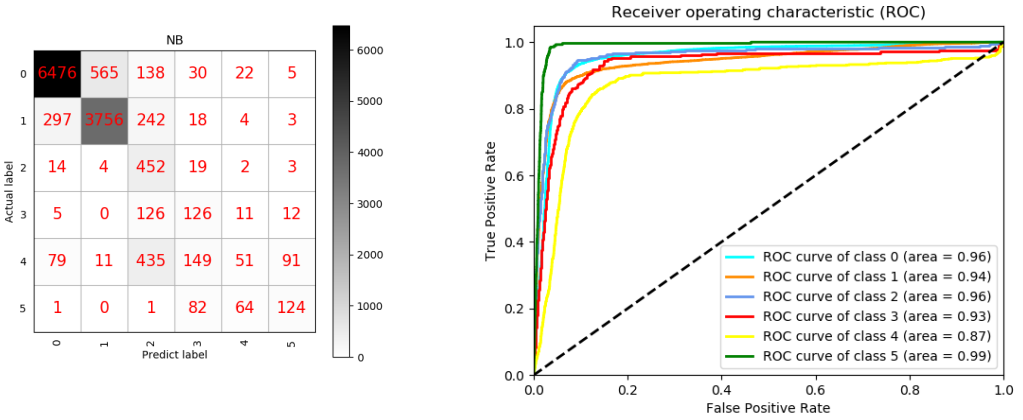
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	94.24	89.50	93.59	91.81	90.41
1	86.62	86.94	93.62	86.78	86.88
2	32.42	91.50	**92.71**	47.88	67.06
3	**29.72**	45.00	97.73	35.80	40.80
4	33.12	**6.25**	99.18	**10.52**	**7.46**
5	52.10	45.59	99.13	48.63	46.76
Mean	54.70	60.80	95.99	53.57	56.56
SD	(+/- 26.38)	(+/- 31.37)	(+/- 2.74)	(+/- 28.26)	(+/- 28.67)
Accuracy			**81.87** (+/- 0.90)		
Weighted AUC			**0.9461**		

Table 4.8 : Performance obtained by NB classifier on undersampled LSM data-set

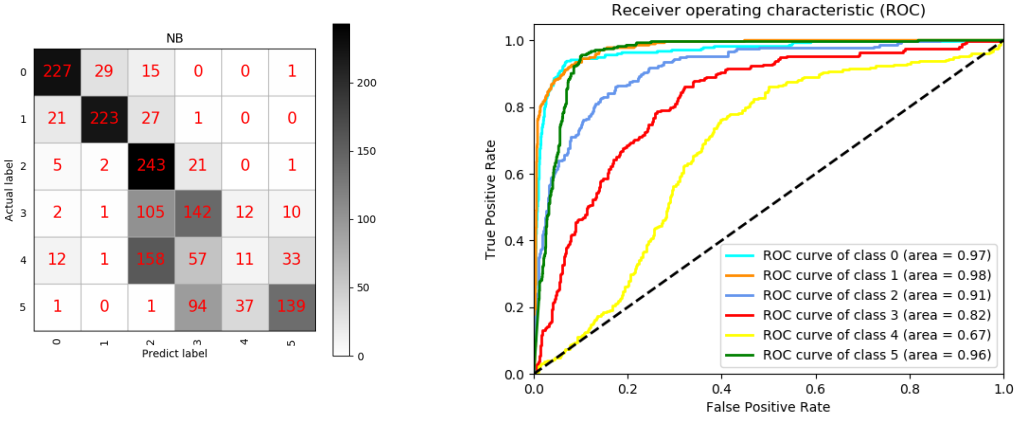
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	86.70	86.08	97.34	86.35	86.18
1	88.17	85.29	97.70	86.69	85.84
2	45.66	91.19	**78.18**	60.00	75.96
3	45.93	51.25	88.01	48.36	50.03
4	**18.85**	**5.08**	95.66	**7.93**	**5.93**
5	74.43	48.77	96.64	58.92	52.38
Mean	59.96	61.28	92.26	58.17	59.39
SD	(+/- 25.21)	(+/- 30.28)	(+/- 7.10)	(+/- 26.57)	(+/- 27.95)
Accuracy			**61.28** (+/- 0.53)		
Weighted AUC			**0.8832**		

Table 4.9 : Performance obtained by NB classifier on oversampled LSM data-set

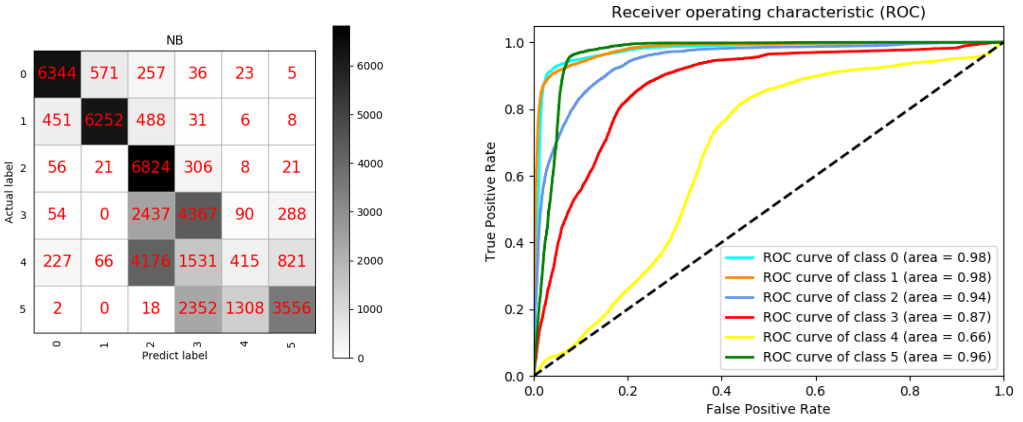
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	88.93	87.67	97.82	88.30	87.92
1	90.48	86.40	98.18	88.39	87.19
2	48.86	94.31	**79.61**	63.67	79.09
3	50.64	60.35	88.24	55.07	58.12
4	**22.43**	**5.74**	96.03	**9.14**	**6.74**
5	75.68	49.14	96.84	69.59	52.85
Mean	62.70	63.93	92.79	60.69	61.99
SD	(+/- 24.53)	(+/- 30.58)	(+/- 6.78)	(+/- 26.55)	(+/- 28.14)
Accuracy			**63.94** (+/- 0.027)		
Weighted AUC			**0.8994**		



(a)



(b)



(c)

Figure 4.3 : Confusion matrix and ROC curve of NB classifier on (a) Original LSM data-set (b) EasyEnsemble undersampled LSM data-set (c) SMOTE oversampled LSM data-set

4.2.1.4 LR Algorithm

The LR algorithm performance on the **original** LSM data-set is shown in Table 4.10. The Recall rate, F1-score and F2-score of both Class 2 and Class 3 are less than 5%, which can be taken as a failure, although the Accuracy is 82.43%. In Table 4.11, it is the LR algorithm performance on the **EasyEnsemble undersampled** LSM data-set. The evaluation value has been obviously improved, but the Accuracy of LR on **undersampled** data-set reduces to 61.84%. In Table 4.12, it is the LR algorithm performance on the **SMOTE oversampled** LSM data-set. The performance of LR algorithm is similar to NB algorithm, which has less enhancement compared to the original case. The Accuracy of LR algorithm on **oversampled** data-set is 67.27%, which is 15.16% lower than the original, 5.43% higher than the undersampled. In Figure 4.4, the AUC values of Class 2, 3, 4 in both sampled cases are worst than the original case.

Table 4.10 : Performance obtained by LR classifier on original LSM data-set

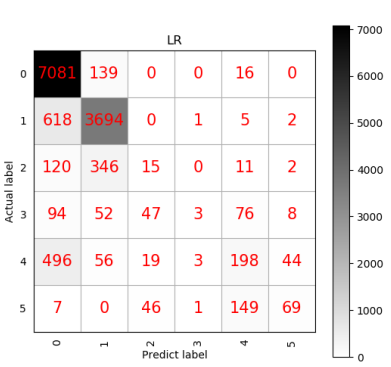
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	84.18	97.86	**78.41**	90.48	94.77
1	86.17	85.51	93.48	85.94	85.64
2	**11.81**	3.04	99.13	4.84	3.57
3	37.50	**1.07**	99.96	**2.08**	**1.33**
4	43.52	24.26	97.96	31.15	26.62
5	55.20	25.37	99.57	34.76	28.44
Mean	53.06	39.52	94.75	41.52	40.06
SD	(+/- 26.15)	(+/- 38.21)	(+/- 7.62)	(+/- 35.16)	(+/- 37.01)
Accuracy			**82.43** (+/- 0.21)		
Weighted AUC			**0.9433**		

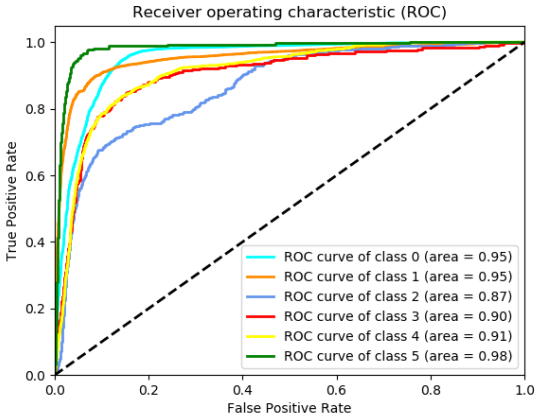
Table 4.11 : Performance obtained by LR classifier on undersampled LSM data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	74.47	92.20	93.66	82.37	88.00
1	86.00	81.44	97.34	83.64	82.30
2	54.86	59.10	90.27	56.85	58.17
3	**39.73**	**25.10**	92.43	**30.66**	**27.05**
4	39.84	44.61	**86.51**	42.05	43.53
5	69.57	68.61	93.99	69.07	68.79
Mean	60.75	61.84	92.37	60.77	61.31
SD	(+/- 17.40)	(+/- 22.38)	(+/- 3.36)	(+/- 19.71)	(+/- 21.25)
Accuracy			**61.84** (+/- 0.14)		
Weighted AUC			**0.8584**		

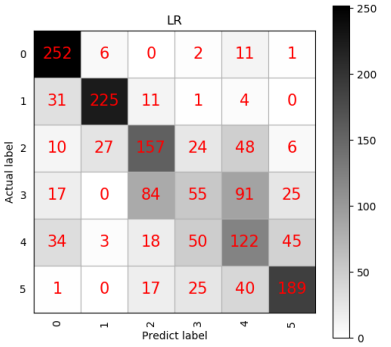
Table 4.12 : Performance obtained by LR classifier on oversampled LSM data-set

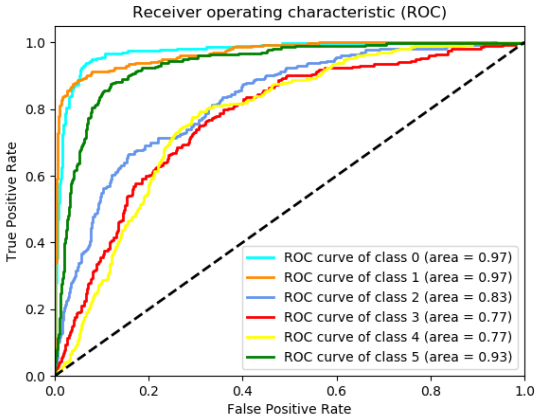
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	78.25	93.93	94.78	85.38	90.31
1	94.42	83.25	99.02	88.48	85.27
2	60.60	71.68	90.68	65.78	69.15
3	49.78	**26.58**	94.64	**43.66**	**29.31**
4	**44.61**	52.36	**87.00**	48.18	50.60
5	73.81	75.84	94.62	74.81	75.43
Mean	66.91	67.27	93.46	66.20	66.68
SD	(+/- 17.14)	(+/- 22.12)	(+/- 3.76)	(+/- 19.40)	(+/- 20.98)
Accuracy			**67.27** (+/- 0.53)		
Weighted AUC			**0.9028**		



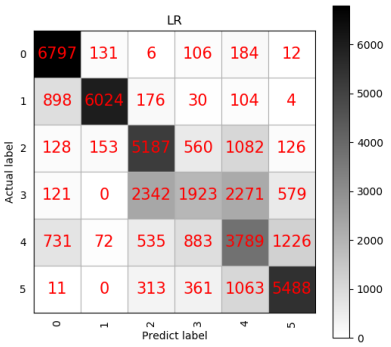


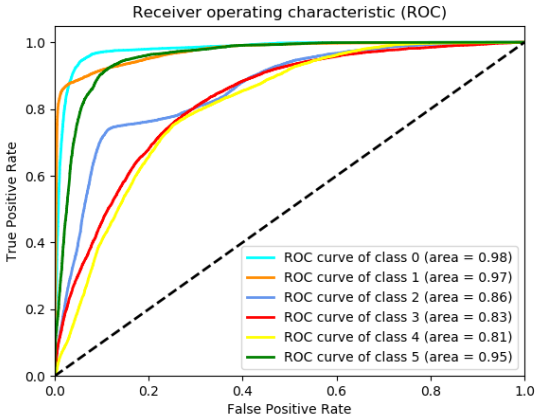
(a)





(b)





(c)

Figure 4.4 : Confusion matrix and ROC curve of LR classifier on (a) Original LSM data-set (b) EasyEnsemble undersampled LSM data-set (c) SMOTE oversampled LSM data-set

4.2.1.5 DT Algorithm

Comparing the data in Table 4.13 and 4.14, which are the original case and undersampled case of DT algorithm performance, the Precision rate, Recall rate, F1-score and F2-score of the undersampled are around 9.00% higher than the original, but the Accuracy drops 10.58%. In Table 4.15, it is the DT algorithm performance on the **SMOTE oversampled** LSM data-set. The lowest value of all evaluations has enhanced to 88.11% (Precision), 86.64% (Recall), 97.66% (Specificity), 87.32% (F1-score) and 86.85% (F2-score). The Accuracy of DT algorithm on **oversampled** data-set is 92.01%, which is 6.68% higher than the original, 17.26% higher than the undersampled. In Figure 4.5, the AUC values of the oversampled case are greater than the original and the undersampled case.

Table 4.13 : Performance obtained by DT classifier on original LSM data-set

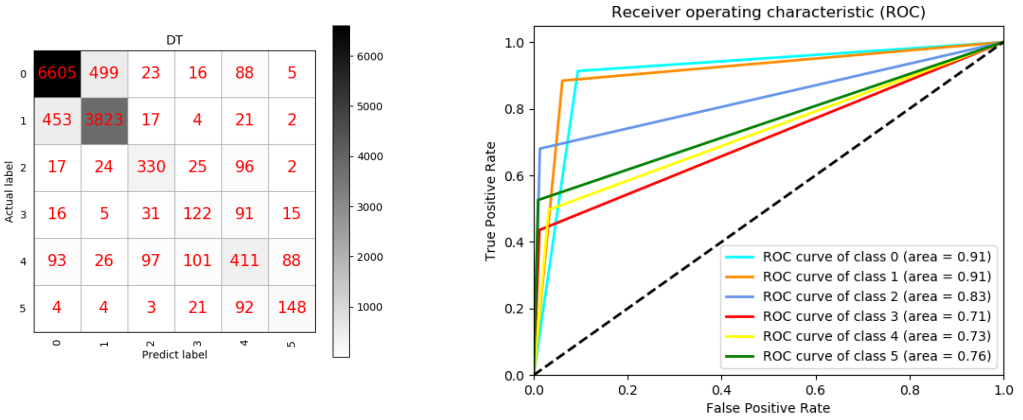
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	91.89	91.28	**90.57**	91.58	91.40
1	87.26	88.50	93.87	87.88	88.25
2	65.87	66.80	98.68	66.33	66.61
3	**42.21**	**43.57**	98.73	**42.88**	**43.29**
4	51.44	50.37	96.92	50.90	50.58
5	56.92	54.41	99.15	55.64	54.89
Mean	65.93	65.82	96.32	65.87	65.84
SD	(+/- 18.18)	(+/- 18.38)	(+/- 3.13)	(+/- 18.27)	(+/- 18.34)
Accuracy			**85.33** (+/- 0.48)		
Weighted AUC			**0.8910**		

Table 4.14 : Performance obtained by DT classifier on undersampled LSM data-set

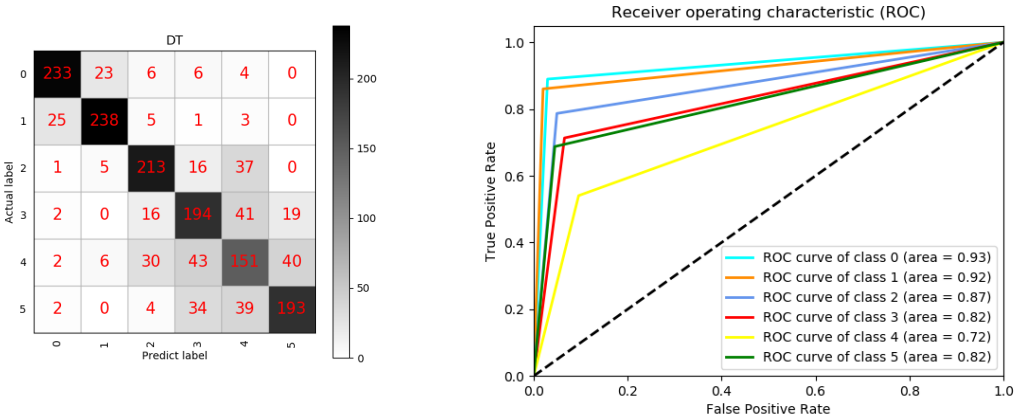
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	84.94	83.86	97.01	84.37	84.06
1	86.54	85.95	97.32	86.22	86.05
2	79.79	79.21	95.98	79.47	89.31
3	68.76	74.52	93.31	71.49	73.27
4	**52.11**	**52.66**	**90.31**	**52.36**	**52.53**
5	76.72	71.32	95.67	73.90	72.33
Mean	74.81	74.59	94.92	74.64	72.33
SD	(+/- 11.69)	(+/- 11.02)	(+/- 2.45)	(+/- 11.25)	(+/- 11.08)
Accuracy			**74.75** (+/- 0.26)		
Weighted AUC			**0.8391**		

Table 4.15 : Performance obtained by DT classifier on oversampled LSM data-set

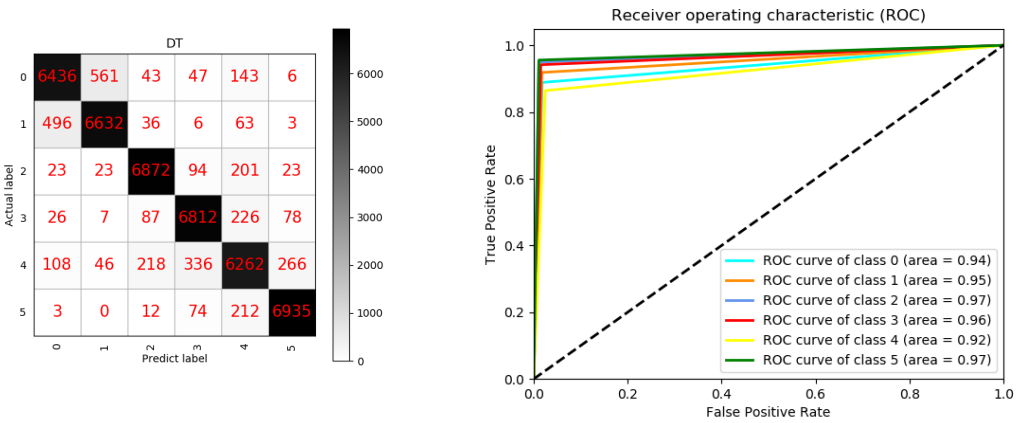
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	90.75	88.94	98.19	89.84	89.30
1	91.24	91.65	98.24	91.44	91.57
2	94.55	94.97	98.91	94.76	94.89
3	92.44	94.14	98.46	93.28	93.80
4	**88.11**	**86.54**	**97.66**	**87.32**	**86.85**
5	94.86	95.84	98.96	95.64	95.64
Mean	91.99	92.01	98.40	92.01	92.01
SD	(+/- 2.31)	(+/- 3.35)	(+/- 0.45)	(+/- 2.81)	(+/- 3.13)
Accuracy			**92.01** (+/- 0.09)		
Weighted AUC			**0.9514**		



(a)



(b)



(c)

Figure 4.5 : Confusion matrix and ROC curve of DT classifier on (a) Original LSM data-set (b) EasyEnsemble undersampled LSM data-set (c) SMOTE oversampled LSM data-set

4.2.1.6 SVM Algorithm

As can be seen in Table 4.16, the Accuracy of SVM algorithm on the **original** LSM data-set is 88.48%, with weakness on classified Class 3 instances. In Table 4.17, the capability of classifying Class 3 under undersampled data-set is improved, Precision rate increase to 68.45%, Recall rate to 64.51%, F1-score to 66.37%, F2-score to 65.23%, but the Accuracy of SVM on **undersampled** data-set drops to 75.75%. In Table 4.18, it is the SVM algorithm performance on the **SMOTE oversampled** LSM data-set. The Accuracy of SVM algorithm is 90.95%, which is 2.47% higher than the original, 15.20% higher than the undersampled. In Figure 4.6, the ROC curve variation of the oversampled case is stabler and the AUC value is higher than other cases.

Table 4.16 : Performance obtained by SVM classifier on original LSM data-set

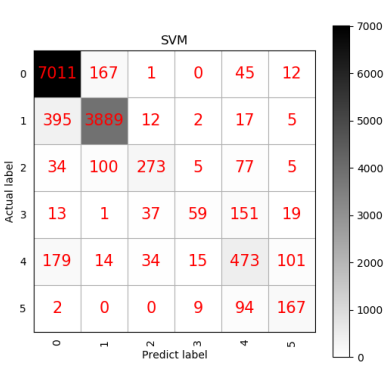
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	91.84	96.89	**89.92**	94.30	95.84
1	93.24	90.02	96.90	91.60	90.65
2	76.47	55.26	99.35	64.16	58.51
3	65.56	**21.07**	99.76	**31.89**	**24.38**
4	55.19	57.97	96.95	56.55	57.39
5	**54.05**	61.40	98.92	57.49	59.77
Mean	72.72	63.77	96.97	66.00	64.42
SD	(+/- 15.85)	(+/- 24.90)	(+/- 3.34)	(+/- 21.54)	(+/- 23.75)
Accuracy			**88.48** (+/- 0.52)		
Weighted AUC			**0.9703**		

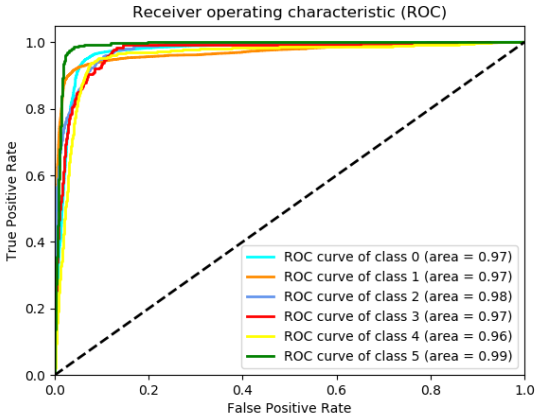
Table 4.17 : Performance obtained by SVM classifier on undersampled LSM data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	85.76	90.13	97.00	87.87	89.21
1	92.98	83.56	98.73	87.99	85.27
2	77.36	74.77	95.60	76.00	75.24
3	68.45	64.51	94.03	66.37	65,23
4	**55.20**	**50.72**	**91.76**	**52.78**	**51.51**
5	74.50	90.81	93.77	81.84	86.99
Mean	75.71	75.75	95.15	75.47	75.58
SD	(+/- 12.09)	(+/- 14.42)	(+/- 2.28)	(+/- 12.58)	(+/- 13.52)
Accuracy			**75.75** (+/- 0.21)		
Weighted AUC			**0.9409**		

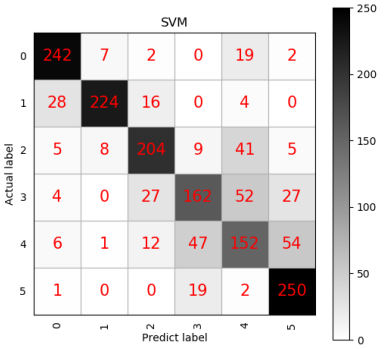
Table 4.18 : Performance obtained by SVM classifier on oversampled LSM data-set

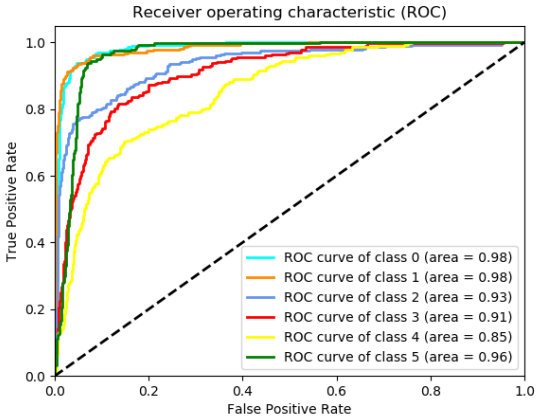
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	89.95	88.95	97.90	91.81	92.96
1	95.72	95.72	99.20	92.61	90.83
2	93.62	93.62	98.82	90.08	88.08
3	90.71	90.71	98.13	90.94	91.09
4	**81.02**	**81.02**	**95.99**	**83.24**	**84.64**
5	95.62	95.62	99.10	97.12	98.04
Mean	91.11	90.95	98.19	90.97	90.94
SD	(+/- 5.02)	(+/- 4.38)	(+/- 1.09)	(+/- 4.12)	(+/- 4.13)
Accuracy			**90.95** (+/- 0.17)		
Weighted AUC			**0.9901**		



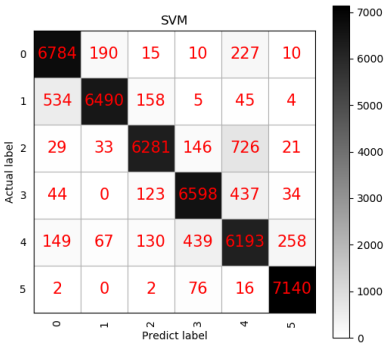


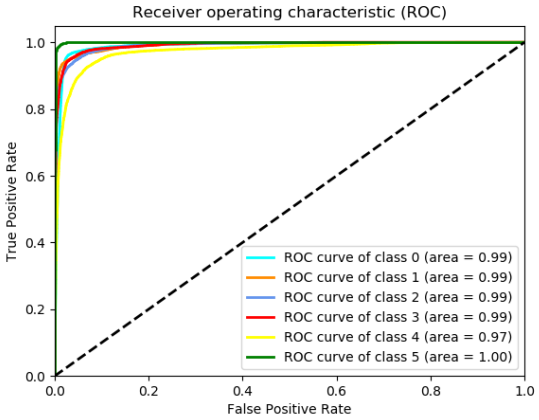
(a)





(b)





(c)

Figure 4.6 : Confusion matrix and ROC curve of SVM classifier on (a) Original LSM data-set (b) EasyEnsemble undersampled LSM data-set (c) SMOTE oversampled LSM data-set

4.2.1.7 AdaBoost Algorithm

In Table 4.19, this is the AdaBoost algorithm performance on the **original** LSM data-set. The Precision rates of Class 3 and Class 5 are "nan", which means those instances in Class 3 and Class 5 are predicted as belonging to other classes. Similar to the original case, in Table 4.20, The Precision rates of Class 3 and Class 5 in the unsersampled case are "nan". Both Accuracy of the original case and undersampled case are 87.49%, but due to the loss of prediction on Class 3 and Class 5, the Accuracy is non-meaningful. After SMOTE oversampling the data-set (Table 4.21, AdaBoost algorithm becomes able to predict the instances in Class 3 and Class 5, but the performance of classifying Class 4 instances is weakened. In Figure 4.7, obviously Class 3 and Class 5 the number of accurately classified instances are 0 or low amount. The AUC value of the oversampled case is slightly lower than the original case, however, combing all evaluation criteria values, the oversampled case is the best.

Table 4.19 : Performance obtained by AdaBoost classifier on original LSM data-set

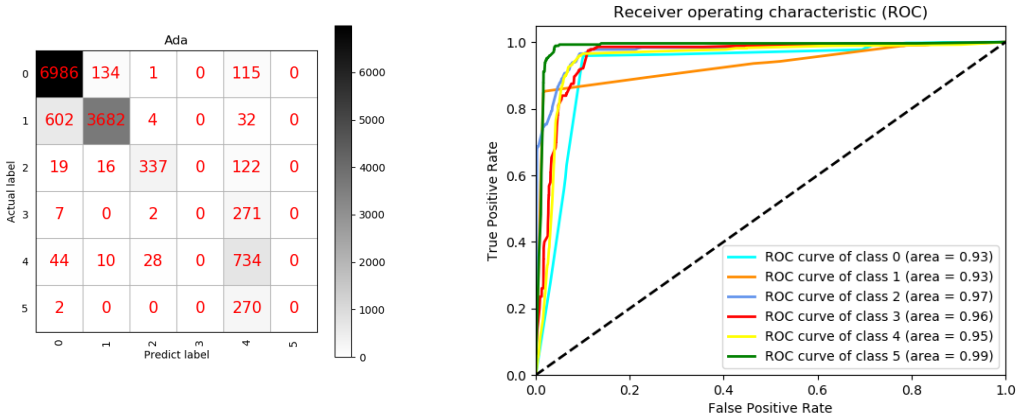
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	91.20	96.55	**89.10**	93.80	95.43
1	95.84	85.23	98.24	90.22	87.16
2	90.59	68.22	99.73	77.83	71.76
3	**nan**	**0.00**	100.00	**nan**	**nan**
4	47.54	89.95	93.57	62.20	76.33
5	**nan**	**0.00**	100.00	**nan**	**nan**
Mean	nan	56.66	96.77	**nan**	**nan**
SD	(+/- nan)	(+/- 40.97)	(+/- 4.10)	(+/- nan)	(+/- nan)
Accuracy			**87.49** (+/- 0.62)		
Weighted AUC			**0.9328**		

Table 4.20 : Performance obtained by AdaBoost classifier on undersampled LSM data-set

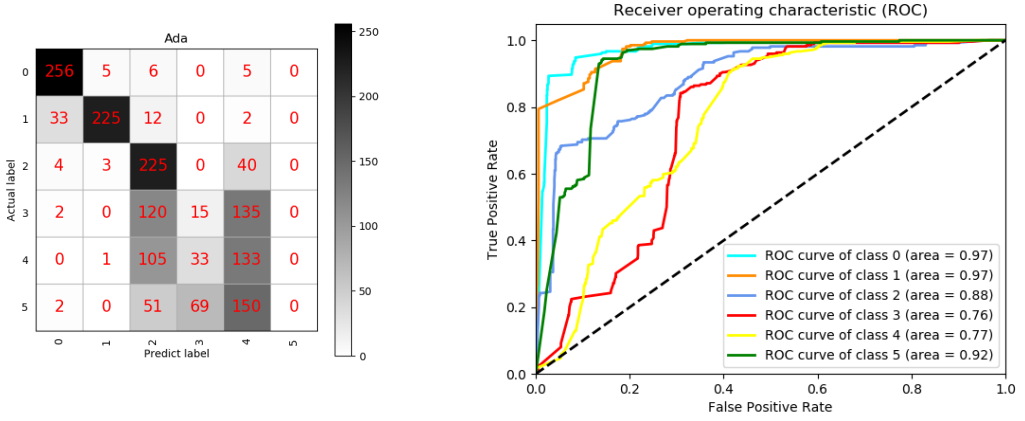
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	83.71	90.24	96.48	86.63	88.71
1	94.90	80.32	99.13	86.72	82.72
2	49.56	77.99	**83.13**	60.02	69.40
3	**nan**	18.98	90.54	**nan**	**nan**
4	34.36	58.25	76.99	41.24	49.55
5	**nan**	**3.73**	99.64	**nan**	**nan**
Mean	nan	54.92	90.98	nan	nan
SD	(+/- nan)	(+/- 32.53)	(+/- 8.46)	(+/- nan)	(+/- nan)
Accuracy			**87.49** (+/- 0.62)		
Weighted AUC			**0.8789**		

Table 4.21 : Performance obtained by AdaBoost classifier on oversampled LSM data-set

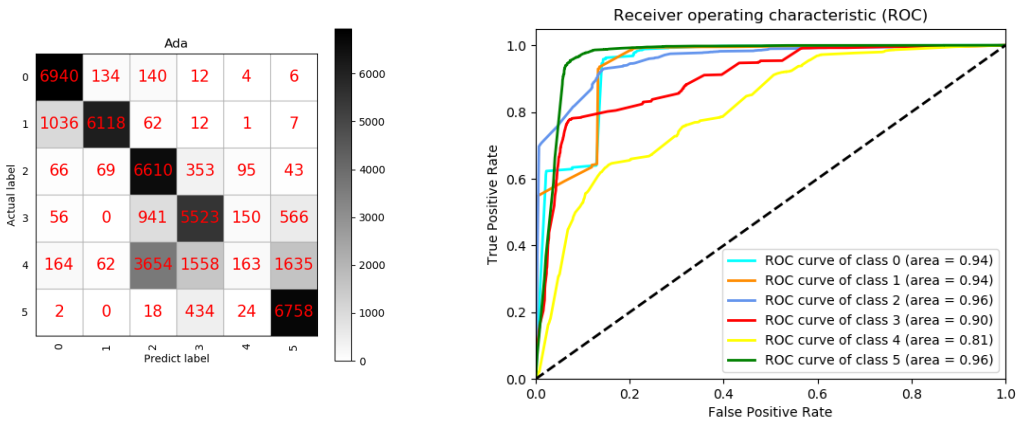
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	83.98	95.91	96.34	89.55	93.26
1	95.85	84.55	99.27	89.85	86.59
2	57.86	91.35	**86.69**	70.85	81.87
3	69.98	76.33	93.45	73.02	74.97
4	**37.30**	**2.25**	99.24	**4.24**	**2.77**
5	74.96	93.39	93.76	83.17	89.01
Mean	69.99	73.96	94.79	68.45	71.41
SD	(+/- 18.73)	(+/- 32.72)	(+/- 4.03)	(+/- 29.64)	(+/- 31.23)
Accuracy			**73.97** (+/- 0.06)		
Weighted AUC			**0.9284**		



(a)



(b)



(c)

Figure 4.7 : Confusion matrix and ROC curve of AdaBoost classifier on (a) Original LSM data-set (b) EasyEnsemble undersampled LSM data-set (c) SMOTE oversampled LSM data-set

In conclusion, in this experiment, it is found that the SMOTE oversampling method is

effective on processing the imbalance LSM data-set to achieve optimising the classification performance of all algorithms, not only the Accuracy but also all evaluation criteria of all classes. To select the suitable algorithms to be the base classifiers of the stacking model in the next experiment, the algorithms should have similar performance on SMOTE sampled data-set and diversity. Regarding the performance of SMOTE sampled cases in this experiment, KNN, RF, DT and SVM have high classification accuracy 96.01%, 95.73%, 92.01% and 90.95%, respectively, and all classes are reliably predictable. Therefore, KNN, RF, DT and SVM are the base classifiers of the stacking models in the next experiment.

4.3 Stacking Models Performances

In this experiment, the base classifiers combined with one of the meta-classifier to build a stacking model. The meta-classifiers were KNN, RF, NB, LR, DT, SVM and AdaBoost. All stacking models had a classification individually on the same SMOTE sampled live data-set to compare performance. The evaluation criteria are same as the first experiment. The goal of this experiment was to select the best meta-classifier for the final stacking model.

4.3.1 Stacking KNN Model

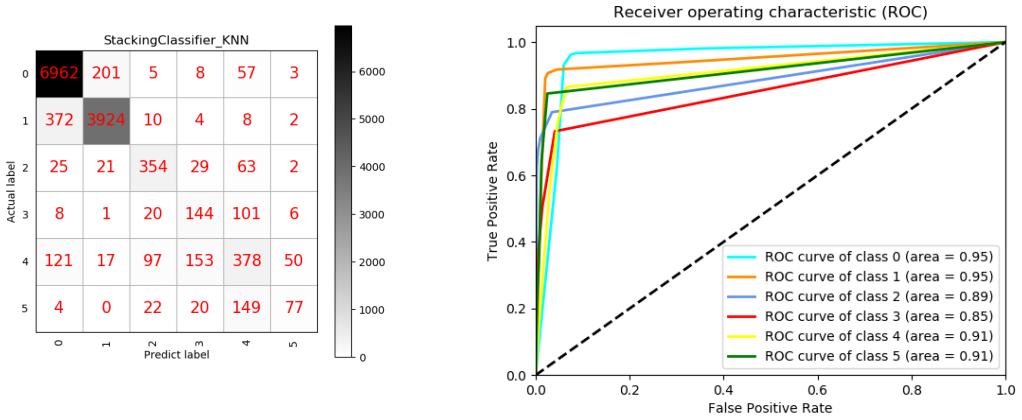
In Table 4.23, it is the Performance of the stacking KNN model on the **SMOTE oversampled** LSM data-set. Compared to the original case in Table 4.22, the evaluation scores of all classes are improved. However, compared to the oversampled case of KNN algorithm, the Accuracy is 5.06 % less than the KNN classifier and the mean value of evaluations is lower. Furthermore, the performance of the stacking KNN model is worst than all of the base classifiers, which means the stacking KNN model is not suitable to classify the LSM data-set.

Table 4.22 : Performance obtained by the stacking KNN model on original data-set

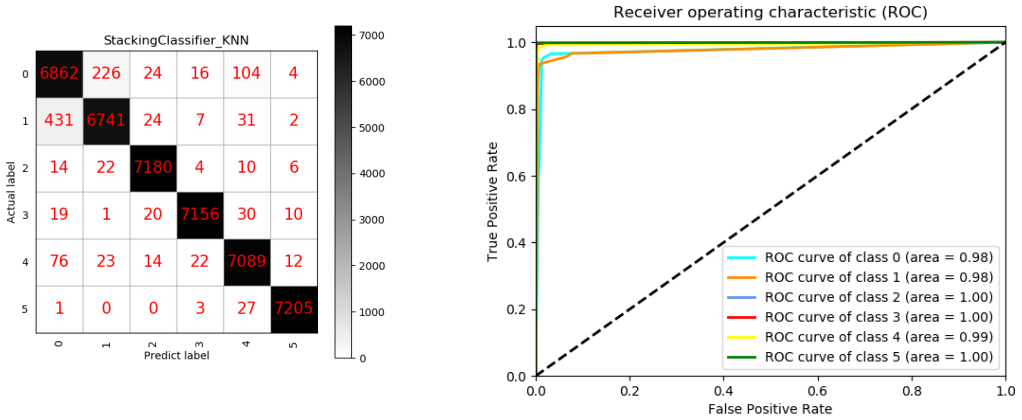
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	91.84	96.89	**89.92**	94.30	95.84
1	93.24	90.02	96.90	91.60	90.65
2	76.47	55.26	99.35	64.16	58.51
3	65.56	**21.07**	99.76	**31.89**	**24.38**
4	55.19	57.97	96.95	56.55	57.39
5	**54.05**	61.40	98.92	57.49	59.77
Mean	72.72	63.77	96.97	66.00	64.42
SD	(+/- 15.85)	(+/- 24.90)	(+/- 3.34)	(+/- 21.54)	(+/- 23.75)
Accuracy			**88.48** (+/- 0.52)		
Weighted AUC			**0.9469**		

Table 4.23 : Performance obtained by the stacking KNN model on SMOTE oversampled data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	89.95	88.95	97.90	91.81	92.96
1	95.72	95.72	99.20	92.61	90.83
2	93.62	93.62	98.82	90.08	88.08
3	90.71	90.71	98.13	90.94	91.09
4	**81.02**	**81.02**	**95.99**	**83.24**	**84.64**
5	95.62	95.62	99.10	97.12	98.04
Mean	91.11	90.95	98.19	90.97	90.94
SD	(+/- 5.02)	(+/- 4.38)	(+/- 1.09)	(+/- 4.12)	(+/- 4.13)
Accuracy			**90.95** (+/- 0.17)		
Weighted AUC			**0.9899**		



(a)



(b)

Figure 4.8 : Confusion matrix and ROC curve of the stacking KNN model on (a) Original data-set (b) SMOTE oversampled data-set

4.3.2 Stacking RF Model

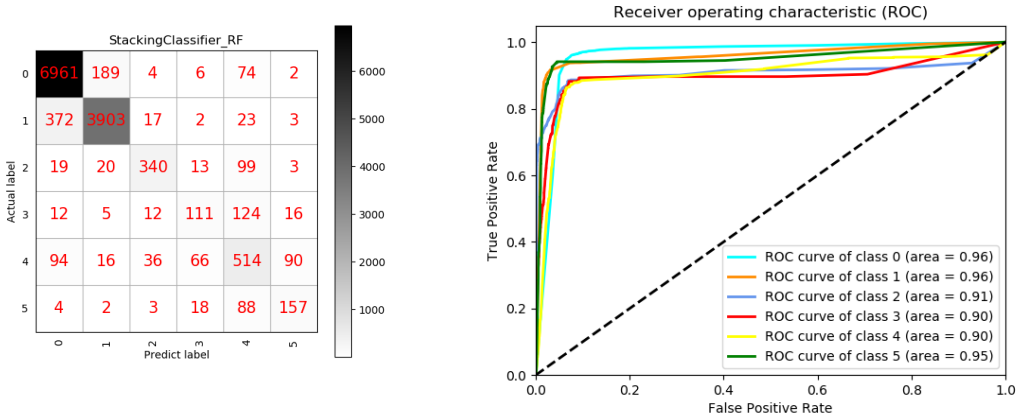
In Table 4.24, The Accuracy of the stacking RF model on **SMOTE oversampled** LSM data-set is up to 97.38% which is 1.65% higher than the RF algorithm. The Precision rate, Recall rate, Specificity, F1-score and F2-score of all classes in the stacking RF model are over 93.00% which is 1.00% higher than the RF, and the mean values are all over 97.99% which is 2.00% greater than the RF. In Figure 4.9 (b), the AUC values of the stacking RF in the oversampled case are almost 1.00 which means this model is highly reliable for this classification.

Table 4.24 : Performance obtained by the stacking RF model on original data-set

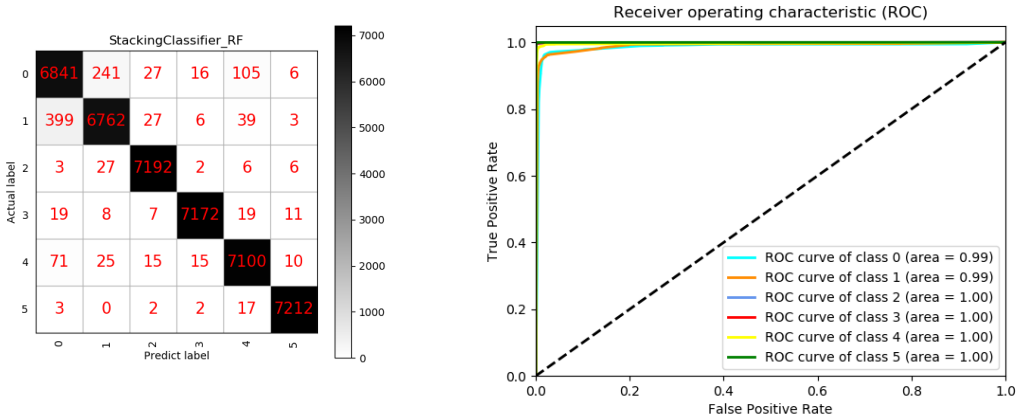
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	93.29	96.20	**91.90**	94.72	95.60
1	94.39	90.35	97.45	92.33	91.13
2	82.52	68.83	99.44	75.06	71.19
3	**51.39**	**39.64**	99.20	**44.76**	**41.54**
4	55.75	62.99	96.76	59.15	61.40
5	57.93	57.72	99.13	57.82	57.76
Mean	72.55	69.29	97.31	70.64	69.77
SD	(+/- 18.03)	(+/- 19.24)	(+/- 2.61)	(+/- 18.42)	(+/- 18.87)
Accuracy			**89.33** (+/- 0.62)		
Weighted AUC			**0.9543**		

Table 4.25 : Performance obtained by the stacking RF model on SMOTE oversampled data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	**93.25**	94.54	**98.63**	93.89	94.28
1	95.74	**93.45**	99.17	**94.58**	**93.90**
2	98.93	99.39	99.78	99.16	99.30
3	99.43	99.12	99.89	99.27	99.18
4	97.45	98.12	99.49	97.98	97.99
5	99.50	99.67	99.90	99.58	99.64
Mean	97.38	97.38	99.48	97.38	97.38
SD	(+/- 2.27)	(+/- 2.46)	(+/- 0.46)	(+/- 2.30)	(+/- 2.38)
Accuracy			**97.38** (+/- 2.27)		
Weighted AUC			**0.9945**		



(a)



(b)

Figure 4.9 : Confusion matrix and ROC curve of the stacking RF model on (a) Original data-set (b) SMOTE oversampled data-set

4.3.3 Stacking NB Model

Comparing the case in Table 4.26 and the oversampled case of NB algorithm, The Accuracy of the stacking NB model is 96.24%, which is 32.30% higher than the NB algorithm. The mean values has improved to 96.47% (Precision), 96.32% (Recall), 99.26%(Specificity), 96.35%(F1-score) and 96.32%(F2-score). Table 4.26 and figure 4.10 show that the average performance of the stacking NB model in evaluations is associated with low variation and high proportion of AUC values of all classes.

Table 4.26 : Performance obtained by the stacking NB model on original data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	93.62	95.55	**92.38**	94.58	95.16
1	95.04	89.98	97.77	92.44	90.95
2	75.92	70.85	99.14	73.30	71.81
3	44.25	**27.50**	99.26	**33.92**	**29.75**
4	50.83	37.75	97.64	43.32	39.80
5	**35.33**	91.18	96.55	50.93	69.28
Mean	65.83	68.80	97.12	64.75	66.13
SD	(+/- 23.62)	(+/- 26.89)	(+/- 2.31)	(+/- 23.56)	(+/- 24.21)
Accuracy			**87.73** (+/- 0.48)		
Weighted AUC			**0.9524**		

Table 4.27 : Performance obtained by the stacking NB model on SMOTE oversampled data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	**90.87**	95.72	**98.08**	**93.23**	94.71
1	97.38	**91.28**	99.51	94.23	**92.44**
2	99.32	98.23	99.86	98.77	98.45
3	99.70	96.95	99.94	98.31	97.49
4	91.88	98.23	98.26	94.95	96.89
5	99.65	97.53	99.93	98.58	97.95
Mean	96.47	96.32	99.26	96.35	96.32
SD	(+/- 3.70)	(+/- 2.41)	(+/- 0.79)	(+/- 2.27)	(+/- 2.10)
Accuracy			**96.24** (+/- 0.12)		
Weighted AUC			**0.9886**		

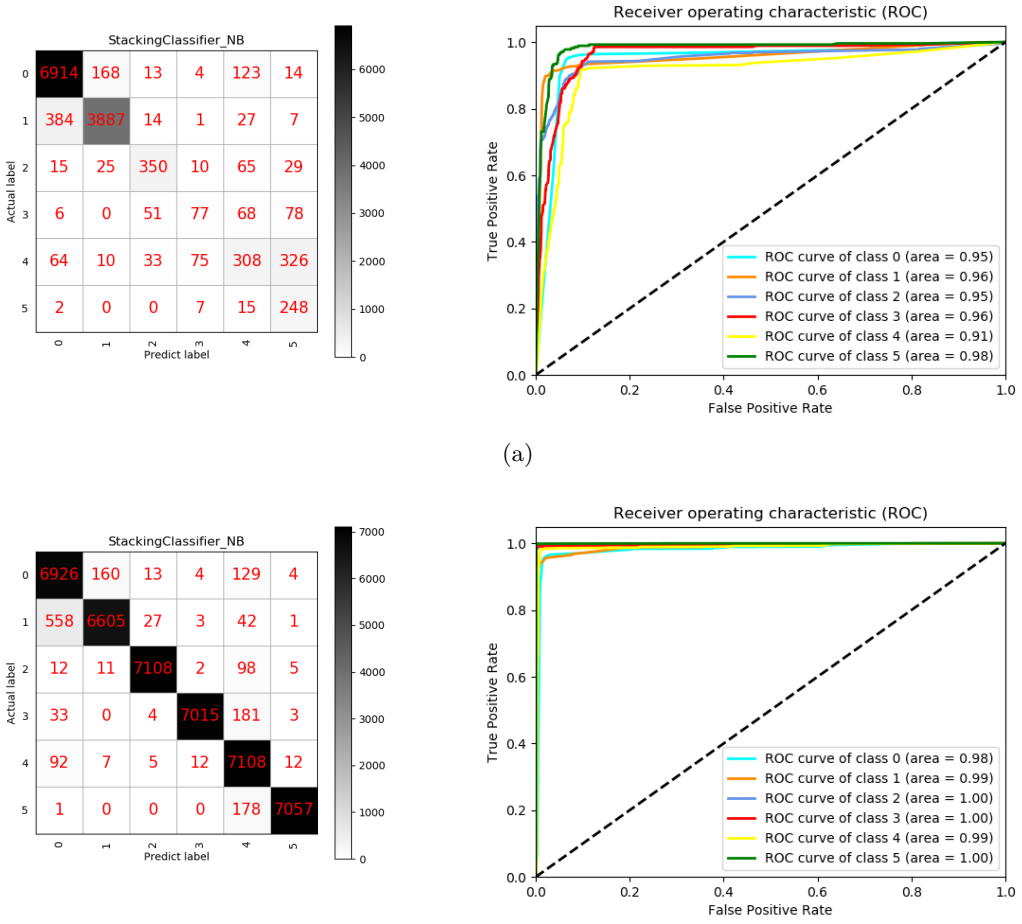


Figure 4.10 : Confusion matrix and ROC curve of the stacking NB model on (a) Original data-set (b) SMOTE oversampled data-set

4.3.4 Stacking LR Model

From Table 4.28, we can see that none of the Class 3 instances were predicted by the stacking LR model on the original data-set, also the instances of Class 2 and Class 5 were rarely classified correctly. Moreover, although the performance of the stacking LR model on the oversampled case is more effective than the original case, the stacking LR model performance is poorer than the performances of the base classifiers.

Table 4.28 : Performance obtained by the stacking LR model on original data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	92.57	96.37	90.94	83.48	95.59
1	74.26	90.07	**85.17**	81.40	86.39
2	20.00	0.40	99.94	0.78	0.50
3	**nan**	**0.00**	1.00	**nan**	**nan**
4	48.56	37.25	97.44	42.17	39.07
5	44.44	0.147	99.96	2.85	1.82
Mean	nan	37.59	95.57	nan	nan
SD	(+/- nan)	(+/- 41.45)	(+/- 5.64)	(+/- nan)	(+/- nan)
Accuracy			**83.16** (+/- 0.23)		
Weighted AUC			**0.9164**		

Table 4.29 : Performance obtained by the stacking LR classifier on SMOTE oversampled data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	92.13	94.13	98.39	93.12	93.72
1	95.74	90.77	99.19	93.19	91.72
2	**88.67**	96.60	**97.53**	92.47	94.90
3	93.62	**89.64**	98.78	**91.59**	**90.41**
4	94.43	92.47	98.91	93.44	92.86
5	99.38	99.79	99.88	99.58	99.71
Mean	94.00	93.90	98.78	93.90	93.89
SD	(+/-3.27)	(+/- 3.46)	(+/- 0.72)	(+/- 2.61)	(+/- 2.97)
Accuracy			93.80 (+/- 0.18)		
Weighted AUC			**0.9718**		

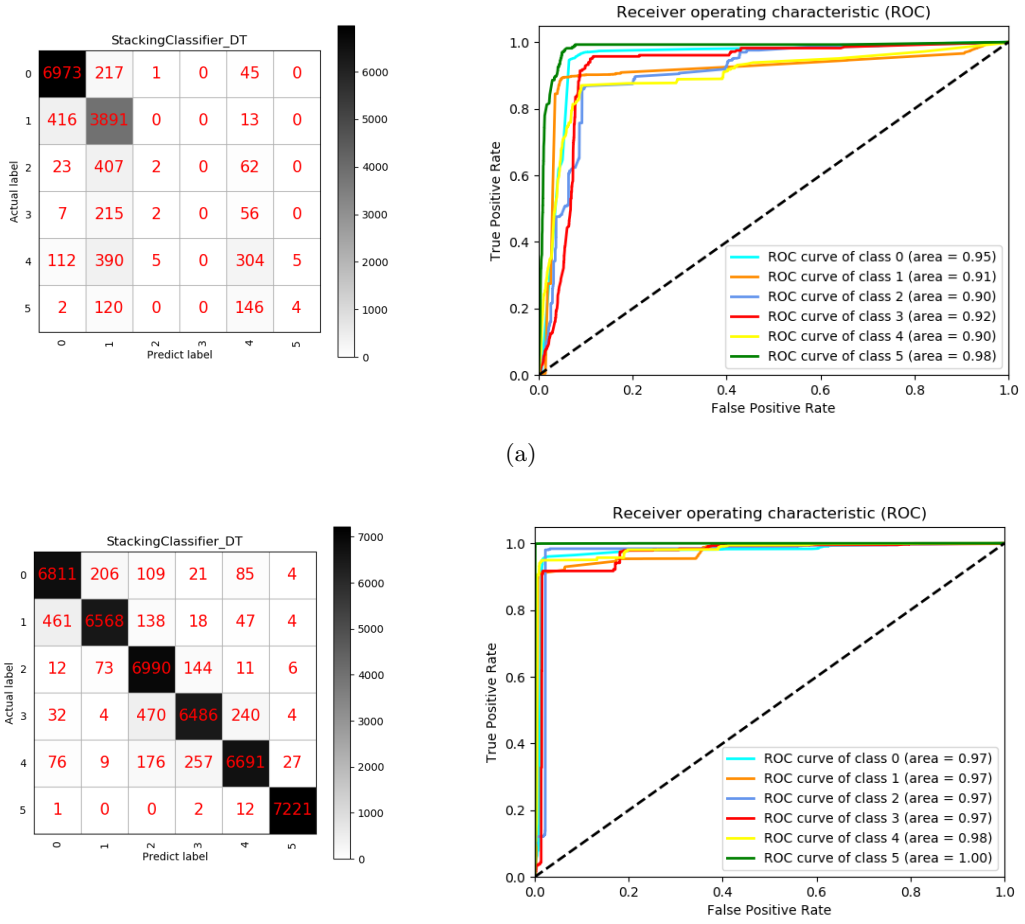


Figure 4.11 : Confusion matrix and ROC curve of the stacking LR model on (a) Original data-set (b) SMOTE oversampled data-set

4.3.5 Stacking DT Model

The stacking DT model performance is similar to the stacking RF model, due to their similar working principle. In Table 4.31, the stacking DT model performs on the over-sampled data-set having the mean value of Precision rate (97.37%), Recall rate (97.36%), Specificity (99.47%), F1-score (97.36%) and F2-score (97.36%), which only have 0.02% difference to the stacking RF model, having same Accuracy 97.38%. However, comparing the deviation of the Accuracy, the stacking DT model is 2.16% less than the stacking RF model, which means the classification performance is more stable.

Table 4.30 : Performance obtained by the stacking DT model on original data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	93.23	96.14	**91.83**	94.66	95.54
1	94.50	90.35	97.50	92.38	91.15
2	80.38	68.83	99.36	74.16	70.87
3	**48.77**	**35.36**	99.21	**41.00**	**37.42**
4	54.38	62.38	96.61	58.11	60.60
5	55.68	54.04	99.11	54.85	54.36
Mean	71.16	67.85	97.27	69.19	68.32
SD	(+/- 18.88)	(+/- 20.75)	(+/- 2.63)	(+/- 19.72)	(+/- 20.32)
Accuracy			**89.23** (+/- 0.32)		
Weighted AUC			**0.9494**		

Table 4.31 : Performance obtained by the stacking DT model on SMOTE oversampled data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	**93.02**	94.71	**98.58**	**93.86**	94.37
1	95.93	**93.46**	99.21	94.68	**93.94**
2	98.93	99.39	99.78	99.16	99.30
3	99.47	98.94	99.89	99.20	99.05
4	97.35	97.35	99.47	97.65	97.83
5	99.50	99.50	99.90	99.60	99.67
Mean	97.37	97.36	99.47	97.36	97.36
SD	(+/- 2.32)	(+/- 2.41)	(+/- 0.47)	(+/- 2.28)	(+/- 2.34)
Accuracy			**97.38** (+/- 0.11)		
Weighted AUC			**0.9930**		

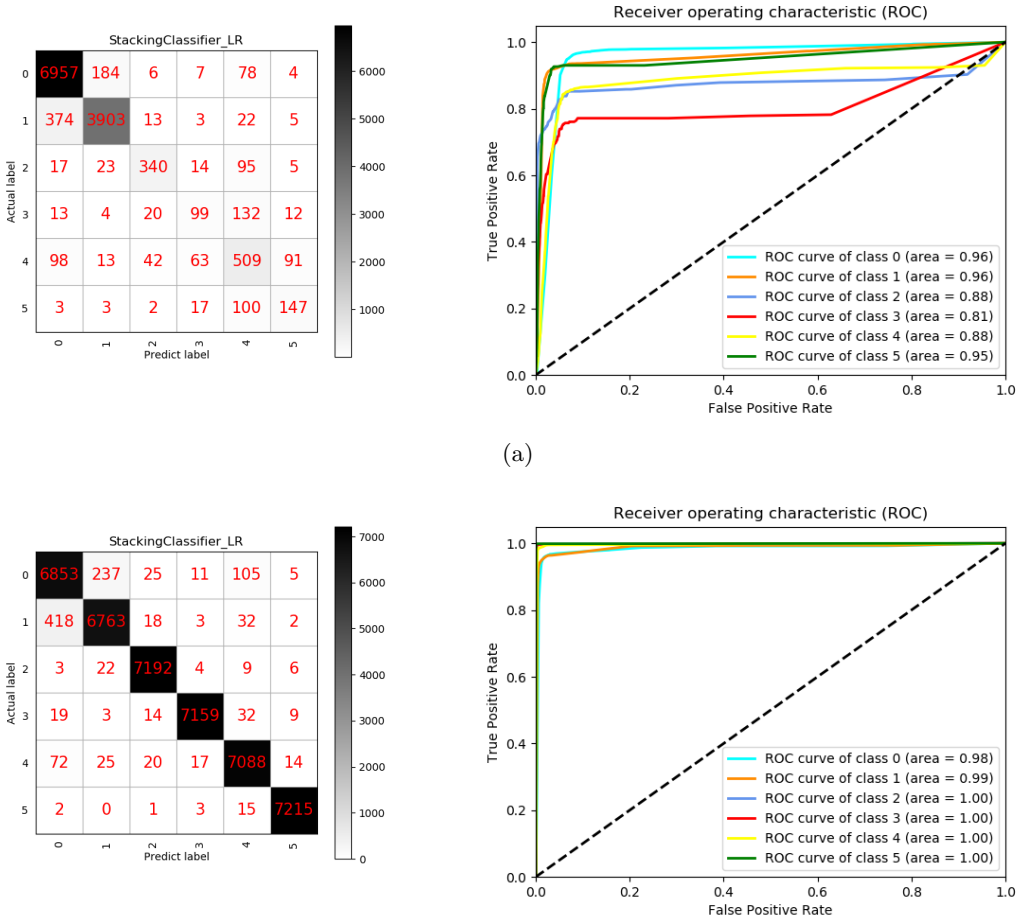


Figure 4.12 : Confusion matrix and ROC curve of the stacking DT model on (a) Original data-set (b) SMOTE oversampled data-set

4.3.6 Stacking SVM Model

In the oversampled case of the stacking SVM model shown in Table 4.33, the mean values of evaluations are similar to the stacking DT model, however, the lowest values are 0.63% (Precision), 0.91% (Recall), 0.15% (Specificity), 0.14% (F1-score) and 0.62% (F2-score) less than the stacking DT model. In addition, the classification time-consumption is much longer than other models.

Table 4.32 : Performance obtained by the stacking SVM model on original data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	93.42	96.12	**92.07**	94.75	95.57
1	94.67	90.37	97.58	92.47	91.20
2	87.96	68.02	99.64	76.72	71.25
3	69.67	**30.36**	99.72	**42.29**	**34.22**
4	**55.20**	68.26	96.41	61.04	65.18
5	58.63	72.43	98.94	64.80	79.17
Mean	76.59	70.93	97.39	72.01	71.10
SD	(+/- 16.16)	(+/- 21.12)	(+/- 2.65)	(+/- 18.32)	(+/- 20.01)
Accuracy			**89.61** (+/- 0.39)		
Weighted AUC			**0.9503**		

Table 4.33 : Performance obtained by the stacking SVM model on oversampled data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	**92.39**	95.08	**98.43**	**93.72**	94.53
1	96.51	**92.55**	99.33	94.49	**93.32**
2	99.13	99.42	99.83	99.27	99.36
3	99.34	99.12	99.87	99.23	99.16
4	97.45	98.20	99.49	97.82	98.05
5	99.46	99.81	99.89	99.63	99.74
Mean	97.38	97.36	99.47	97.36	97.36
SD	(+/- 2.48)	(+/- 2.66)	(+/- 0.51)	(+/- 2.38)	(+/- 2.51)
Accuracy			**97.41** (+/- 0.16)		
Weighted AUC			**0.9941**		

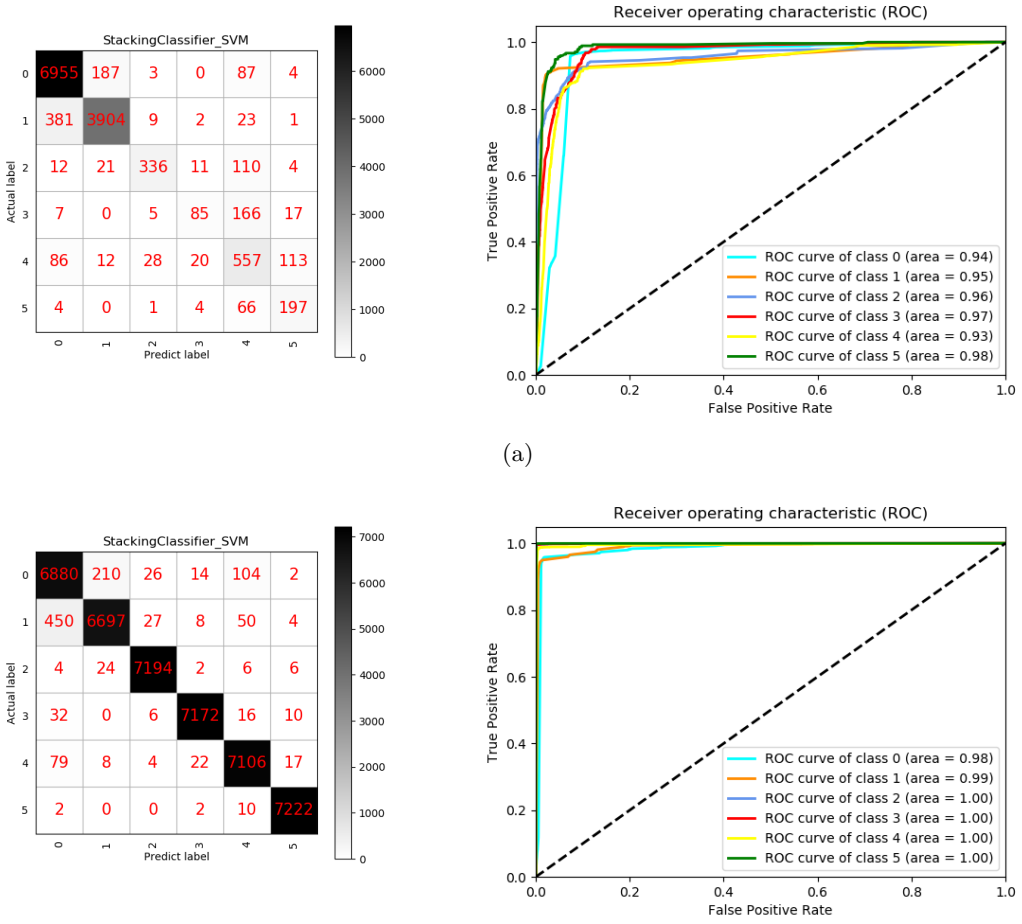


Figure 4.13 : Confusion matrix and ROC curve of the stacking SVM model on (a) Original data-set (b) SMOTE oversampled data-set

4.3.7 Stacking AdaBoost Model

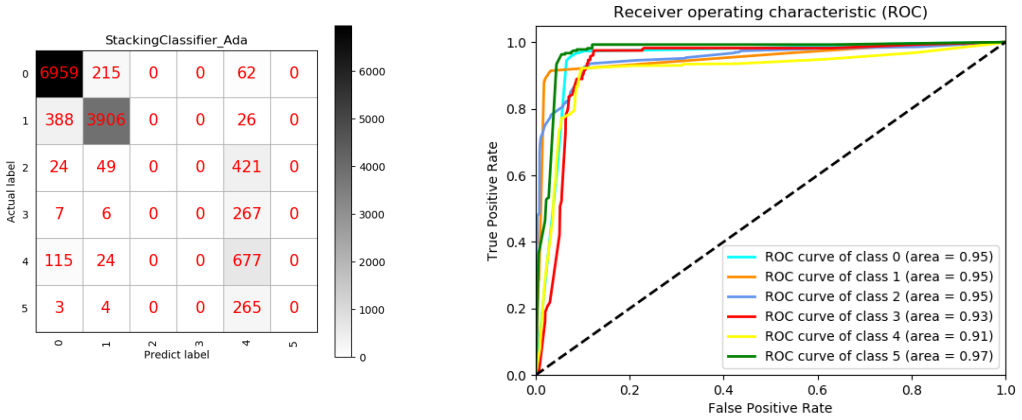
As we can see in Table 4.34, the stacking AdaBoost model has poorly classified the instances in Class 2, Class 3 and Class 5 on the original data-set. In the oversampled case shown in Table 4.35, although the Class 2, Class 3 and Class 5 instances are precisely classified, the Class 1 instances are disregarded.

Table 4.34 : Performance obtained by the stacking AdaBoost model on original data-set

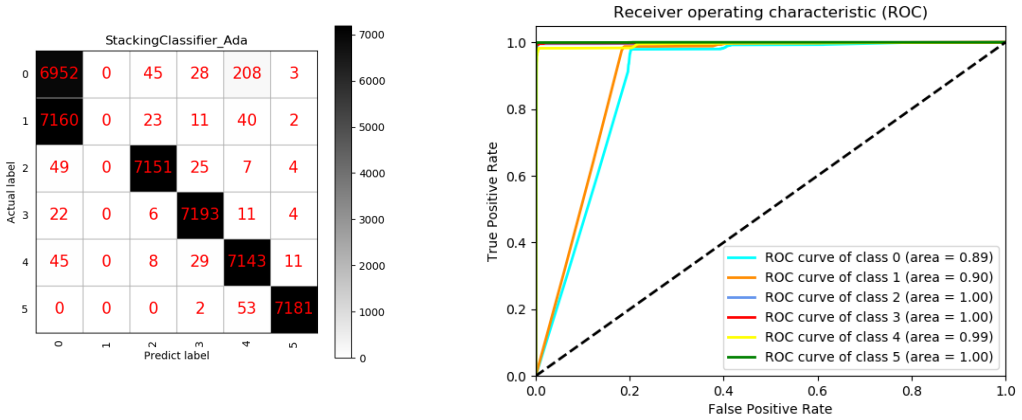
Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	92.84	96.17	**91.31**	94.48	95.49
1	92.91	90.42	96.72	91.65	90.91
2	**nan**	**0.00**	1.00	**nan**	**nan**
3	**nan**	**0.00**	1.00	**nan**	**nan**
4	39.41	82.97	91.74	53.44	67.95
5	**nan**	**0.00**	1.00	**nan**	**nan**
Mean	nan	44.93	96.63	nan	nan
SD	(+/- nan)	(+/- 45.09)	(+/- 3.79)	(+/- nan)	(+/- nan)
Accuracy			**86.15** (+/- 0.16)		
Weighted AUC			**0.9491**		

Table 4.35 : Performance obtained by the stacking AdaBoost model on SMOTE oversampled data-set

Class	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
0	48.86	96.08	**79.89**	64.78	80.52
1	**nan**	**0.00**	1.00	**nan**	**nan**
2	98.87	98.83	99.77	98.85	98.84
3	98.70	99.41	99.74	99.05	99.27
4	95.73	98.71	99.12	97.20	98.10
5	99.67	99.24	99.93	99.45	99.33
Mean	nan	82.05	96.41	nan	nan
SD	(+/- nan)	(+/- 36.71)	(+/- 7.39)	(+/- nan)	(+/- nan)
Accuracy			82.31 (+/- 0.22)		
Weighted AUC			**0.9882**		



(a)



(b)

Figure 4.14 : Confusion matrix and ROC curve of the stacking AdaBoost model on (a) Original data-set (b) SMOTE oversampled data-set

By comparing the stacking models' performance, the stacking RF model, the stacking DT model and the stacking SVM model have similar performance and better than the others. In Table 4.36, a comparison of these three models on SMOTE oversampling data-set is shown. The weighted AUC values are similar but the Accuracy deviation of the stacking DT model is lower than the stacking RF, and the time-consumption of the stacking SVM is long. Therefore, the stacking DT model is the final selected model to predict liver stiffness classification.

Table 4.36 : Performance obtained by selected three stacking models on SMOTE oversampled data-set

Model	Precision(%)	Recall(%)	Specificity(%)	F1-score(%)	F2-score(%)
Stacking RF	97.38	97.38	99.48	97.38	97.38
	(+/-2.27)	(+/-2.46)	(+/-0.46)	(+/-2.30)	(+/-2.38)
Stacking DT	97.37	97.36	99.47	97.36	97.36
	(+/-2.32)	(+/-2.41)	(+/-0.47)	(+/-2.28)	(+/-2.34)
Stacking SVM	97.38	97.36	99.47	97.36	97.36
	(+/-2.48)	(+/-2.66)	(+/-0.51)	(+/-2.38)	(+/-2.51)

Model	Accuracy(%)	Weighted AUC
Stacking RF	97.38 (+/-2.27)	0.9945
Stacking DT	97.38 (+/-0.11)	0.9930
Stacking SVM	97.41 (+/-0.16)	0.9941

Chapter 5

Conclusion and Future Work

Liver cirrhosis is a serious human health issue and advanced notice and appropriate treatments are crucial considerations worldwide. Using FibroScan to exam patients' liver condition must improve success rate to give the most accurate results in the future. To be more convincing with doctors' opinions of FibroScan results, a model with the stacking method was proposed as assistance in this study. In this chapter, we summarize the conclusions about the methods to solve the tasks of doctors' opinion prediction in this thesis. At the end of the thesis, future work will be presented.

5.1 Main Findings

This study has developed an automatic approach to predict the opinions for patients regarding the liver stiffness measurements from FibroScan tests. As we discussed in Chapter 2, two main issues were involved to process the classification task: the imbalanced class distribution of the data-set and the selection of effective classification models. The doctors' opinions were transferred to labels by using Natural Language Processing and the variables were pre-processed by the undersampling method and the oversampling method. Comparative assessments of five popular algorithms (LR, DT, SVM, KNN and NB) and three ensemble methods (Bagging - RF, Boosting - AdaBoost, Stacking) based on four base classifiers selected by first performance comparison was carried out.

5.2 Dissertation Contributions

Instead of targeting for high accuracy of the classification, we were concentrated on the ability of the model to correctly identify the most important instances that are the

least represented and less likely to be predicted in the liver data-set. After the performance comparisons, the selection of the best imbalanced learning method and the selection of the best classification algorithm were obtained.

Selected the imbalanced learning method for liver stiffness classification

The imbalanced class distribution was a significant factor affecting the classifier performance. To select the best imbalanced learning method, we evaluated the results of 7 different algorithms (KNN, RF, LR, NB, DT, SVM and AdaBoost) on EasyEnsemble undersampling method and SMOTE oversampling method, respectively. This evaluation demonstrated how these two methods processing the data and influencing the models' performance. As we can see in Section 4.2, in general, the evaluation scores of EasyEnsemble undersampling method dramatically decreased, in contrary to the oversampling method. All evaluation scores of SMOTE oversampling method increased to a very high level, which leads to conclude that the SMOTE oversampling method was effective to solve the data imbalanced issue.

Selected the best algorithm for liver stiffness classification

Performance comparisons by 7 different algorithms and 7 different stacking models were resulted in Section 4.2. In the first comparison, the evaluation scores of KNN, RF, DT and SVM on SMOTE oversampled data-set were high and similar, therefore they were selected to be the base classifiers for the stacking models. In the second comparison, the best classification performance was provided by the stacking model using DT as meta-classifier.

The best stacking model for liver stiffness classification

After the comparison of the stacking models' performance, the stacking DT model performed on the SMOTE oversampled data-set resulting in the mean values of Precision rate (97.37%), Recall rate (97.36%), Specificity (99.47%), F1-score (97.36%) and F2-score (97.36%) and the highest Accuracy 97.38% with less deviation than the other models, in addition, greater than any of the supervised algorithms in the experiment.

In conclusion, in this study, to solve the main issues of making prediction of doctors'

opinions, SMOTE oversampling method was the best imbalanced learning method and the stacking DT model (using KNN, RF, DT and SVM as base classifiers, DT as meta-classifier) was the best algorithm. The principle of the stacking DT model is shown in Figure 5.1. In clinical practice, doctors can input patients' FibroScan results into the stacking model and obtain the most related opinion based on previous similar cases so that to have a more accurate diagnosis.

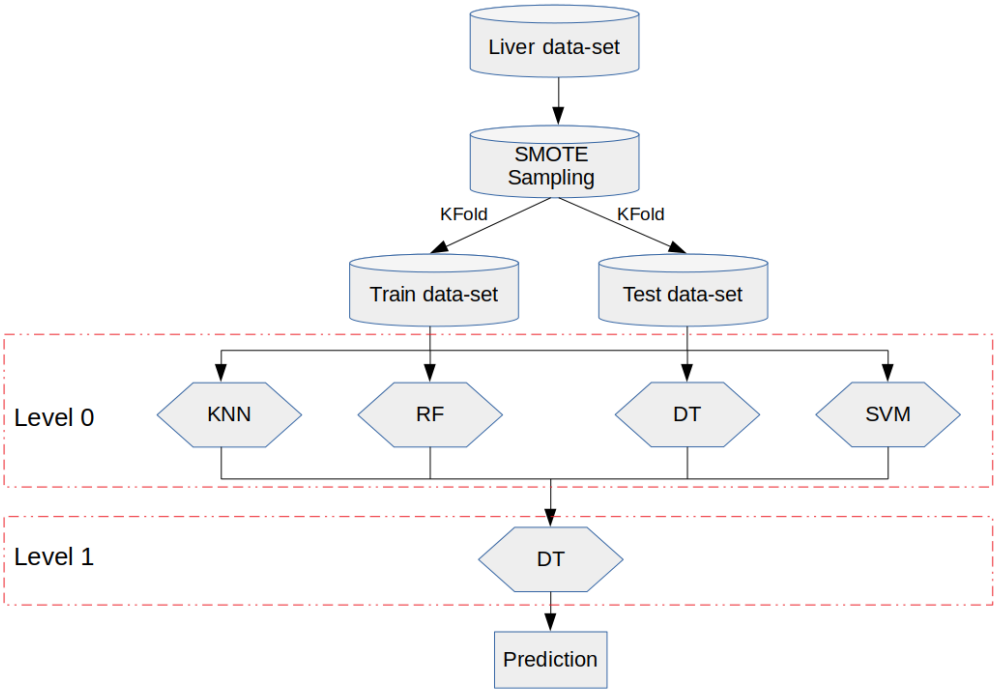


Figure 5.1 : Proposed stacking model (KNN, RF, DT, SVM as base classifiers and RF as meta-classifier)

5.3 Future work

In this section, we present the potential aspects to continue the research and investigate possible enhancements in the methods proposed in this thesis.

New Data-sets Analysis

In this study, we have presented the work on liver stiffness measurements data-set to predict doctors' opinions, which was related to liver cirrhosis disease. Since the data-set is only related to Chinese patients, in order to evaluate the Staking model performance, it

will be interesting to utilize other liver stiffness measurement data-sets from other countries, or different biomedical data-sets to output predictions, or non-biomedical data-sets to observe the model if it can obtain similar performance as this study.

Other sampling methods

The EasyEnsemble undersampling method used in this study cannot stand for all undersampling methods. Although using EasyEnsemble undersampling method to pre-processed the liver data-set failed to improve the classifiers' performance, other undersampling methods can be different. Also, the SMOTE oversampling method is not the unique method to oversample data. Borderline-SMOTE is an optimised method based on SMOTE algorithm. The original SMOTE algorithm treats all the minority samples equally, but in the actual modeling process, it is found that the samples at the boundary position are more likely to be wrongly divided, so using the sample information at the boundary position to generate new samples can bring more promotion. Borderline-SMOTE is an algorithm that combines the original SMOTE algorithm and the border information algorithm.

Evaluation of different algorithms

The Stacking algorithm is one of the ensemble learning methods. In this study, the stacking models compare performances with RF algorithm using Bagging and AdaBoost using Boosting. GBoosting and XGBoosting algorithms are considered to be compared with the stacking model because of their perfect performance in most of the classification cases and possibly combine as base classifiers or meta-classifiers if performing constructive effects. Furthermore, the Bagging algorithm and Boosting algorithm can modify the base learner to observe and compare performance. We will apply different algorithms to Bagging and Boosting methods and make comparisons with the future stacking models.

Feature Selection Methods and Hyper-parameter Optimisation

Besides the above future work, in the future, we will also focus on feature selection and hyper-parameter optimisation of the stacking DT model on the liver stiffness measurement data-set. We will utilize feature selection algorithms to select the most related features

as input for the stacking model in order to improve performance and increase efficiency. For hyper-parameter optimisation, we will use three popular optimisers such as Bayesian optimisation, Grid Search and Random search methods to solve this problem respectively and compare the performance.

Reference

Adams R.P. 2018, 'Linear classification with logistic regression', *Princeton University*.

Afdhal N. 2012, *Fibroscan (transient elastography) for the measurement of liver fibrosis*, viewed 14 September 2019, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3594956/>.

Al Amrani, Y., Lazaar, M., & El Kadiri, K. E. 2018, 'Random forest and support vector machine based hybrid approach to sentiment analysis', *Procedia Computer Science*, vol. 127, pp. 511-520.

Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G. & Spyropoulos, C.D. 2000, 'An evaluation of naive bayesian anti-spam filtering', *Machine Learning in the New Information Age*,pp. 9-17.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. & Spyropoulos, C. Jul 1, 2000, 'An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages', *ACM*, pp. 160-7.

Bhasuran, B., Murugesan, G., Abdulkadhar, S. & Natarajan, J. 2016, 'Stacked ensemble combined with fuzzy matching for biomedical named entity recognition of diseases', *Journal of Biomedical Informatics*, vol. 64, pp. 1-9.

Belgiu, M. & Drăguţ, L. 2016, 'Random forest in remote sensing: a review of applications and future directions', *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24-31.

Bewick, V., Cheek, L. & Ball, J. 2005, 'Statistics review 14: logistic regression', *Critical care*, vol. 9, no. 1, pp. 112-8.

Bhargava, N., Sharma, G., Bhargava, R. & Mathuria, M. 2013, 'Decision tree analysis on j48 algorithm for data mining', *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6.

Biau, G. & Scornet, E. 2016, 'A random forest guided tour', *TEST*, vol. 25, no. 2, pp.

197-227.

Bird R.S. 2018, *Decision Trees-A simple way to visualize a decision*, viewed 24 September 2019, <https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb>.

Boulesteix, A., Janitza, S., Kruppa, J. & König, I.R. 2012, 'Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 6, pp. 493-507.

Boursier, J., Zarski, J., de Ledinghen, V., Rousselet, M., Sturm, N., Lebail, B., Fouchard-Hubert, I., Gallois, Y., Oberti, F., Bertrais, S. & Calès, P. 2013, 'Determination of reliability criteria for liver stiffness evaluation by transient elastography', *Hepatology*, vol. 57, no. 3, pp. 1182-91.

Breiman, L. 2001, 'Random Forests', *Machine Learning*, vol. 45, no. 1, pp. 5-32.

British Liver Trust 2019, *Cirrhosis of the liver*, viewed 10 September 2019, <https://www.britishlivertrust.org.uk/liver-information/liver-conditions/cirrhosis/>.

Cai, D. & Zhao, H. 2016, 'Neural word segmentation learning for Chinese', *arXiv preprint*, arXiv:1606.04300.

Cambria, E. & White, B. 2014, 'Jumping NLP curves: a review of natural language processing research', *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48-57.

Cao, P., Liu, X., Zhang, J., Zhao, D., Huang & M., Zaiane, O. 2016, '$l_{2,1}$ norm regularized multi-kernel based joint nonlinear feature selection and over-sampling for imbalanced data classification', *Neurocomputing*, vol. 234, pp. 38-57.

Cassinotto, C., Lapuyade, B., Mouries, A., Hiriart, J., Vergniol, J., Gaye, D., Castain, C., Le Bail, B., Chermak, F., Foucher, J., Laurent, F., Montaudon, M. & De Ledinghen, V.

2014, 'Non-invasive assessment of liver fibrosis with impulse elastography: comparison of supersonic shear imaging with ARFI and FibroScan', *Journal of Hepatology*, vol. 61, no. 3, pp. 550-7.

Castéra, L., Foucher, J., Bernard, P., Carvalho, F., Allaix, D., Merrouche, W., Couzigou, P. & de Lédinghen, V. 2010, 'Pitfalls of liver stiffness measurement: A 5-year prospective study of 13,369 examinations', *Hepatology (Baltimore, Md.)*, vol. 51, no. 3, pp. 828-35.

Chand, N., Mishra, P., Krishna, C.R., Pilli, E.S. & Govil, M.C. Apr 2016, 'A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection', *IEEE*, pp. 1-6.

Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer, W.P. 2002, 'SMOTE: Synthetic minority over-sampling technique', *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-57.

Ren, J., Zheng, Z., Liu, Q., Wei, Z. & Yan, H. 2019, 'A buffer overflow prediction approach based on software metrics and machine learning', *Hindawi*, vol. 2019.

Chen, X., Shi, Z., Qiu, X. & Huang, X. 2017, 'DAG-based long short-term memory for neural word segmentation', *arXiv preprint*, arXiv:1707.00248.

Chou, C., Chang, C. & Huang, Y. 2016, 'Boosted web named entity recognition via tri-training', *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 16, no. 2, pp. 1-23.

Cleveland Clinic 2018, 'Elevated liver enzymes: possible causes', viewed 28 August 2019, <https://my.clevelandclinic.org/health/symptoms/17679-elevated-liver-enzymes/possible-causes>.

Collins, M., Schapire, R. & Singer, Y. 2002, 'Logistic regression, AdaBoost and Bregman distances', *Machine Learning*, vol. 48, no. 1, pp. 253-85.

Corpechot, C., Gaouar, F., El Naggar, A., Kemgang, A., Wendum, D., Poupon, R., Carrat,

F. & Chazouillères, O. 2014, 'Baseline values and changes in liver stiffness measured by transient elastography are associated with severity of fibrosis and outcomes of patients with primary sclerosing cholangitis', *Gastroenterology*, vol. 146, no. 4, pp. 97,979.e6.

Cunha J.P. 2019, *Cirrhosis*, viewed 10 September 2019, <https://www.medicinenet.com/cirrhosis/article.htm#cirrhosis_of_the_liver_definition_and_facts>.

Dietterich, T. G. 2002, 'Ensemble learning', *The handbook of brain theory and neural networks*, vol. 2, pp. 110-125.

Džeroski, S. & Ženko, B. 2004, 'Is combining classifiers with stacking better than selecting the best one?', *Machine Learning*, vol. 54, no. 3, pp. 255-73.

ECHOSENS 2017,*FibroScan® and the role of liver stiffness*, viewed 10 September 2019, <https://www.echosens.com/en/fibroscan-and-the-role-of-liver-stiffness>.

Elkan, C. 1997, 'Boosting and naive Bayesian learning', *Proceedings of the nternational Conference on Knowledge Discovery and Data Mining.*

Ellis, K., Kerr, J., Godbole, S., Lanckriet, G., Wing, D. & Marshall, S. 2014, 'A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip accelerometers', *Physiological measurement*, vol. 35, no. 11, pp. 2191-203.

Familyfirst Intervention 2019, *Recognizing The Early Symptoms Of Liver Disease –Alcoholic And Non-Alcoholic*, viewed 15 September 2019, <https://family-intervention.com/blog/recognizing-early-symptoms-of-liver-disease/>.

Fan, S., Jia, Y. & Jia, C. 2019, 'A feature selection and classification method for activity recognition based on an inertial sensing unit', *Information*, vol. 10, no, 10, pp. 290.

Fang, W., Jiang, T., Jiang K., Zhang, F., Ding, Y. & Sheng J. 2020, 'A method of automatic text summarisation based on long short-term memory', *International Journal of Computational Science and Engineering (IJCSE)*, vol. 22, no. 1.

Foucher, J., Chanteloup, E., Vergniol, J., Castera, L., Le Bail, B., Adhoute, X., Bertet, J., Couzigou, P. & de Ledinghen, V. 2006, 'Diagnosis of cirrhosis by transient elastography (FibroScan): a prospective study'. *Gut*, vol. 55, no. 3, pp. 403-408.

Friedl, M.A. & Brodley, C.E. 1997, 'Decision tree classification of land cover from remotely sensed data', *Remote Sensing of Environment*, vol. 61, no. 3, pp. 399-409.

Friedrich-Rust, M., Rosenberg, W., Parkes, J., Herrmann, E., Zeuzem, S. & Sarrazin, C. 2010, 'Comparison of ELF, FibroTest and FibroScan for the non-invasive assessment of liver fibrosis', *BMC gastroenterology*, vol. 10, no. 1, pp. 103.

Frohlich, H., Chapelle, O. & Scholkopf, B. 2003, 'Feature selection for support vector machines by means of genetic algorithm', *IEEE*, pp. 142-8.

Gambell, T., & Yang, C. 2006, 'Word segmentation: quick but not dirty', Unpublished manuscript.

Gastrointestinal Society 2019, *Fibroscan-corelatation_GI_Large*, viewed 15 September 2019, <https://badgut.org/information-centre/multimedia/your-liver-cirrhosis/fibroscan-corela- tation_gi_large-2/>.

Guo, Y., Wu, X., Shen, L., Zhang, Z. & Zhang, Y. 2019, 'Method of gait disorders in Parkinson's disease classification based on machine learning algorithms', *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC).*

Guss, D., Sherigar, J. & Mohanty, S. 2018, 'Missed diagnosis of liver cirrhosis Leads to disparities in care for older patients',*Gastroenterology Res.*, vol. 11, no. 5, pp. 333-9.

Hand D.J. & Yu K. 2001, "Idiot's Bayes-not so stupid after all?', *International Statistical Review*, vol. 69, no. 3, pp. 385-98.

Hastie, T., Rosset, S., Zhu, J. & Zou, H. 2009, 'Multi-class AdaBoost', *Statistics and Its Interface*, vol. 2, no. 3, pp. 349-60.

Hepatitis NSW 2015, *Hepatitis C: Fibroscan and liver biopsy*, viewed 15 September 2019, <https://www.mydr.com.au/tests-investigations/hepatitis-c-liver-biopsy-fibroscan>.

Hepatol C. 2018, *High rates of liver stiffness, fibrosis discovered in general population*, viewed 10 September 2019, <https://www.healio.com/hepatology/steatohepatitis-metabolic-liver-disease/news/online/%7B84affc07-d258-4722-b87e-0341795ca5b4%7D/high-rates-of-liver -stiffness-fibrosis-discovered-in-general-population>.

Hirschberg, J., & Manning, C. D. 2015, 'Advances in natural language processing', *Science*, vol. 349, no. 6245, pp. 261-6.

Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. 2013, 'Applied logistic regression', *John Wiley & Sons*, vol. 398.

Huang, P.J. 2015, 'Classification of imbalanced data using synthetic over-sampling techniques', *University of California.*

Imandoust, S.B. & Bolandraftar, M. 2013, 'Application of K-Nearest Neighbor (KNN) approach for predicting economic events: theoretical background', *International Journal of Engineering Research and Applications*, vol. 3, no. 5.

Jabbar, M.A., Deekshatulu, B.L. & Chandra, P. 2013, 'Classification of heart disease using K-Nearest Neighbor and genetic algorithm', *Procedia Technology*, vol. 10, pp. 85-94.

Japkowicz, N. & Shah, M. 2011, 'Evaluating learning algorithms: A classification perspective', *Cambridge University Press.*

Kemp, W. & Roberts, S. 2013, 'FibroScan and transient elastography', *Australian Family Physician*, vol. 42, no. 7, pp. 468-71.

Khalilia, M., Chakraborty, S. & Popescu, M. 2011, 'Predicting disease risks from highly imbalanced data using random forest', *BMC medical informatics and decision making*, vol. 11, no. 1, pp. 51.

Kim, D.Y., Kim, S.U., Ahn, S.H., Park, J.Y., Lee, J.M., Park, Y.N., Yoon, K.T., Paik,

Y.H., Lee, K.S., Chon, C.Y. & Han, K. 2009, 'Usefulness of FibroScan for detection of early compensated liver cirrhosis in chronic hepatitis B', *Digestive diseases and sciences*, vol. 54, no. 8, pp. 1758-63.

Klibansky, D.A., Mehta, S.H., Curry, M., Nasser, I., Challies, T. & Afdhal, N.H. 2012, 'Transient elastography for predicting clinical outcomes in patients with chronic liver disease', *Journal of Viral Hepatitis*, vol. 19, no. 2, pp. e184-93.

Lei, H., Liu, J. & Xian, C. 2019, 'Application of distributed machine learning model in fault diagnosis of air preheater', *IEEE 2019 4th International Conference on System Reliability and Safety (ICSRS).*

Leshem, G. 2005, 'Improvement of Adaboost algorithm by using random forests as weak learner and using this algorithm as statistics machine learning for traffic flow prediction', *Research proposal for a Ph. D. Research proposal for a Ph. D. thesis, the Hebrew university of Jerusalem.*

Li, K., Gu, Y., Zhang, P., Wang, A. & Li, W. 2019, 'Research on KNN algorithm in malicious PDF files classification under adversarial environment', *International Conference on Big Data and Computing*, pp. 156-9.

Liang, P., Li, W. & Hu, J. 2018, 'Oversampling the minority class in a multi-Linear feature space for imbalanced data classification', *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 13, no. 10, pp.1483-91.

Liu, J., Liang, W., Jing, W. & Liu, M. 2019, 'Countdown to 2030: eliminating hepatitis B disease, China', *Bulletin of the World Health Organization*, vol. 97, no. 3, pp. 230-8.

Li, J. & Fong, S. 2018, 'Benchmarking swarm rebalancing algorithm for relieving imbalanced machine learning problems',*Behavior Engineering and Applications*, pp. 1-40.

Li, X., Wang, L. & Sung, E. 2008, 'AdaBoost with SVM-based component classifiers', *Engineering applications of Artificial Intelligence*, vol. 21, no. 5, pp. 785-95.

Li, X.Z. & Kong, J.M. 2014, 'Application of GA–SVM method with parameter optimization for landslide development prediction', *Natural Hazards and Earth System Sciences*, vol. 14, no. 3, pp. 525-33.

Lu, H., Ma X. & Azimi M. 2020, 'US natural gas consumption prediction using an improved kernel-based nonlinear extension of the Arps decline model', *Science Direct Energy*, vol. 194.

Lupascu, C.A., Tegolo, D. & Trucco, E. 2010, 'FABC: retinal vessel segmentation using AdaBoost', *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1267-74.

Ma, Y., Liang, F., Wang, P., Lv, H., Yu, X., Zhang, Y. & Wang, J. 2019, 'An accurate method to distinguish between stationary human and dog targets under through-wall condition using UWB radar', *Remote Sens*, vol. 11, no. 21, pp. 2571.

Malik, R., Lai, M., Sadiq, A., Farnan, R., Mehta, S., Nasser, I., Challies, T., Schuppan, D. & Afdhal, N. 2010, 'Comparison of transient elastography, serum markers and clinical signs for the diagnosis of compensated cirrhosis', *Journal of Gastroenterology and Hepatology*, vol. 25, no. 9, pp. 1562-8.

Meng, Y., Li, X., Sun, X., Han, Q., Yuan, A. & Li, J. 2019, 'Is word segmentation necessary for deep learning of chinese representations?', *arXiv preprint*, arXiv:1905.05526.

Miailhes, P., Pradat, P., Chevallier, M., Lacombe, K., Bailly, F., Cotte, L., Trabaud, M.-, Boibieux, A., Bottero, J., Trepo, C. & Zoulim, F. 2011, 'Proficiency of transient elastography compared to liver biopsy for the assessment of fibrosis in HIV/HBV-coinfected patients', Journal of Viral Hepatitis, vol. 18, no. 1, pp. 61-9.

Millonig, G., Friedrich, S., Adolf, S., Fonouni, H., Golriz, M., Mehrabi, A., Stiefel, P., Pöschl, G., Büchler, M.W., Seitz, H.K. & Mueller, S. 2009, 'Liver stiffness is directly influenced by central venous pressure', *Journal of Hepatology*, vol. 52, no. 2, pp.

206-10.

Morrison, D., Wang, R. & De Silva, L.C. 2007, 'Ensemble methods for spoken emotion recognition in call-centres', *Speech Communication*, vol. 49, no. 2, pp. 98-112.

Mueller, S. 2010, 'Liver stiffness: a novel parameter for the diagnosis of liver disease', *Hepatic Medicine: Evidence and Research*, vol. 2, pp. 49-67.

Myers, R.P., Pomier-Layrargues, G., Kirsch, R., Pollett, A., Duarte-Rojo, A., Wong, D., Beaton, M., Levstik, M., Crotty, P. & Elkashab, M. 2012, 'Feasibility and diagnostic performance of the FibroScan XL probe for liver stiffness measurement in overweight and obese patients', *Hepatology*, vol. 55, no. 1, pp. 199-208.

Mountrakis, G., Im, J. & Ogole, C. 2011, 'Support vector machines in remote sensing: A review', *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, no. 3, pp. 247-59.

Nguyen, C., Wang, Y. & Nguyen, H.N. 2013, 'Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic', *Journal of Biomedical Science and Engineering*, vol. 6, no. 5, pp. 551-60.

Nguyen, H.M., Cooper, E.W. & Kamei, K. 2011, 'Borderline over-sampling for imbalanced data classification', *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, p. 4.

Nie, W., Liu, P., Jia, K., Liao, H. & Huang, X. 2018, 'Taxi license plate block detection based on complex environment', *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 81-85.

Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B. & Kohli, P. 2011, 'Decision tree fields'. *2011 International Conference on Computer Vision*, *IEEE*, pp. 1668-1675.

Orrù, G., Pettersson-Yeo, W., Marquand, A.F., Sartori, G. & Mechelli, A. 2012, 'Using support vector machine to identify imaging biomarkers of neurological and psychiatric

disease: a critical review', *Neuroscience and Biobehavioral Reviews*, vol. 36, no. 4, pp. 1140-52.

Pal, M. 2005, 'Random forest classifier for remote sensing classification', *International Journal of Remote Sensing*, vol. 26, no. 1, pp. 217-22.

Pannier, B. 1985, 'Cirrhosis of the liver', *Revue de l'infirmiere*, vol. 35, no. 13, pp. 31-4.

Pardo, M. & Sberveglieri, G. 2005, 'Classification of electronic nose data with support vector machines', S*ensors* & *Actuators: B. Chemical*, vol. 107, no. 2, pp. 730-7.

Peng, H., Cambria, E. & Hussain, A. 2017, 'A review of sentiment analysis research in Chinese language', *Cognitive Computation*, vol. 9, no. 4, pp. 423-35.

Poorten D., Vongsuvanh R., Zarghom O., Siow W. 2019, *Fibroscan with CAP - the Non-Invasive Liver Biopsy*, viewed 14 September 2019, <http://www.sydneynwgastro.com.au/services/fibroscan>.

Pradhan, B. 2013, 'A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using GIS', *Computers and Geosciences*, vol. 51, pp. 350-65.

Pretorius, A. 2016, 'Advances in random forests with application to classification', *Doctoral dissertation, Stellenbosch: Stellenbosch University.*

Pruengkarn, R., Fung, C.C. & Wong K.W. 2015, 'Using misclassification data to improve classification performance', *2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON).*

Qiu, W. 2019, 'Credit risk prediction in an imbalanced social lending environment based on XGBoost', *2019 5th International Conference on Big Data and Information Analytics (BigDIA).*

Rao, T., & Rajinikanth, T. V. 2014, 'A hybrid random forest based support vector ma-

chine classification supplemented by boosting', *Global Journal of Computer Science and Technology*.

Raut, S. 2017, *A simple introduction to natural language processing*, viewed 1 September 2019, <https://www.digitalistmag.com/digital-economy/2017/02/20/simple-introduction-to-nat ural-language-processing-04906091>.

Rajaraman, S., Candemir S., Xue Z., Alderson P.,Kohli M.,Abuya J.,Thoma G.,Antani S. 2018, 'A novel stacked generalization of models for improved TB detection in chest radiographs', *40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 718-721.

Robic, M.A., Procopet, B., Métivier, S., Péron, J.M., Selves, J., Vinel, J.P. & Bureau, C. 2011, 'Liver stiffness accurately predicts portal hypertension related complications in patients with chronic liver disease: A prospective study', *Journal of Hepatology*, vol. 55, no. 5, pp. 1017-24.

Rodriguez, J.J., Kuncheva, L.I. & Alonso, C.J. 2006, 'Rotation forest: a new classifier ensemble method', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619-30.

Saffran, J.R., Newport, E.L. & Aslin, R.N. 1996, 'Word segmentation: the role of distributional cues', *Journal of Memory and Language*, vol. 35, no. 4, pp. 606-21.

Sardari, S., Eftekhari, M. & Afsari, F. 2017, 'Hesitant fuzzy decision tree approach for highly imbalanced data classification', *Applied Soft Computing*, vol. 61, pp. 727-41.

Sarin S.K. & Mainwall R. 2020, 'Global burden of liver disease: A true burden on health sciences and economies', viewed 26 June 2020, <https://www.worldgastroenterology.org/publications/e-wgn/e-wgn-expert-point-of-view-articles-collection/global-burden-of-liver-disease-a-true-burden-on-health-sciences-and-economies>.

Saini, I., Singh, D. & Khosla, A. 2013, 'QRS detection using K-Nearest Neighbor algorithm

(KNN) and evaluation on standard ECG databases', *Journal of Advanced Research*, vol. 4, no. 4, pp. 331-44.

Schapire, R.E. 2013, 'Explaining AdaBoost', *Empirical Inference*, pp. 37-52.

Schuppan, D. & Afdhal, N.H. 2008, 'Liver cirrhosis', *The Lancet*, vol. 371, no. 9615, pp. 838-51.

Sikora, R. & Al-Laymoun, O. 2015, 'A modified stacking ensemble machine learning algorithm using genetic algorithms', *Handbook of Research on Organizational Transformations through Big Data Analytics*, *IGi Global*, pp. 43-53.

Singh, S., Fujii, L.L., Murad, M.H., Wang, Z., Asrani, S.K., Ehman, R.L., Kamath, P.S. & Talwalkar, J.A. 2013, 'Liver stiffness is associated with risk of decompensation, liver cancer, and death in patients with chronic liver diseases: a systematic review and meta-analysis', *Clinical Gastroenterology and Hepatology*, vol. 11, no. 12, pp. 157,1584.e2.

Song Y.Y. & Lu Y. 2015, *Decision tree methods: applications for classification and prediction*, viewed 24 September 2019, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/>.

Sperandei, S. 2014, 'Understanding logistic regression analysis. *Biochemia medica*, vol. 24, no. 1, pp. 12-18.

Such, M.V., Lohr, R., Beckman, T. & Naessens, J.M. 2017, 'Extent of diagnostic agreement among medical referrals', *Journal of Evaluation in Clinical Practice*.

Suguna, N., & Thanushkodi, K. 2010,' An improved k-nearest neighbor classification using genetic algorithm', *International Journal of Computer Science Issues*, vol. 7, no. 2, pp. 18-21.

Syarif, I., Zaluska, E., Prugel-Bennett, A. & Wills, G. 2012, 'Application of bagging, boosting and stacking to intrusion detection', *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 593-602.

Ting, K. M., & Witten, I. H. 1997, 'Stacked generalization: when does it work?'.

Tsochatzis, E. A., Bosch, J., & Burroughs, A. K. 2014, 'Liver cirrhosis', *The Lancet*, vol. 383, no. 9930, pp. 1749-1761.

Vergniol, J., Foucher, J., Terrebonne, E., Bernard, P., le Bail, B., Merrouche, W., Couzigou, P. & de Ledinghen, V. 2011, 'Noninvasive tests for fibrosis and liver stiffness predict 5-Year outcomes of patients with chronic hepatitis C', *Gastroenterology*, vol. 140, no. 7, pp. 197,1979.e3.

Wade T. 2016, *The clinical significance of compensated versus decompensated cirrhosis*, viewed 10 September 2019, <https://www.tomwademd.net/the-clinical-significance-of-compensat ed-versus-decompensated-cirrhosis/>.

Wang, H., Zhang, W., Zeng, Q., Li, Z., Feng, K. & Liu, L. 2014, 'Extracting important information from Chinese operation notes with natural language processing methods', *Journal of Biomedical Informatics*, vol. 48, pp. 130-6.

Wang, H. & Zhao, T. 2008, 'Identifying named entities in biomedical text based on stacked generalization', *7th World Congress on Intelligent Control and Automation. IEEE*, pp. 160-164.

Wang, H., He, S., Yu, J., Wang, L. & Liu, T. 2020, 'Research and implementation of vehicle target detection and information recognition technology based on NI myRIO', *Sensors*, vol. 20, no. 6, pp. 1765.

Wang, J., Li, J., Zhou, Q., Zhang, D., Bi, Q., Wu, Y. & Huang, W. 2018, 'Liver stiffness measurement predicted liver-related events and all-cause mortality: a systematic review and nonlinear dose–response meta-analysis', *Hepatology Communications*, vol. 2, no. 4, pp. 467-76.

Wang, G., Hao, J., Ma, J., & Jiang, H. 2011, 'A comparative assessment of ensemble learning for credit scoring', E*xpert systems with applications*, vol. 38, no. 1,pp. 223-230.

Wang, Z., Lai, C., Chen, X., Yang, B., Zhao, S., & Bai, X. 2015, 'Flood hazard risk assessment model based on random forest', *Journal of Hydrology*, vol. 527, pp. 1130-1141.

Wong, V.W., Vergniol, J., Wong, G.L., Foucher, J., Chan, A.W., Chermak, F., Choi, P.C., Merrouche, W., Chu, S.H., Pesque, S., Chan, H.L. & de Lédinghen, V. 2012, 'Liver stiffness measurement using XL probe in patients with nonalcoholic fatty liver disease', *The American journal of gastroenterology*, vol. 107, no. 12, pp. 1862-71.

Wong, V.W., Wong, G.L., Yeung, D.K., Lau, T.K., Chan, C.K., Chim, A.M., Abrigo, J.M., Chan, R.S., Woo, J., Tse, Y., Chu, W.C. & Chan, H.L. 2014, 'Incidence of non-alcoholic fatty liver disease in Hong Kong: a population study with paired proton-magnetic resonance spectroscopy', *Journal of Hepatology*, vol. 62, no. 1, pp. 182-9.

Wu, D., Jiang, Z., Xie, X., Wei, X., Yu, W. & Li, R. 2019, 'LSTM learning with bayesian and gaussian processing for anomaly detection in industrial IoT', *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5244-53.

Xiao, H., Shi, M., Xie, Y. & Chi, X. 2017, 'Comparison of diagnostic accuracy of magnetic resonance elastography and Fibroscan for detecting liver fibrosis in chronic hepatitis B patients: A systematic review and meta-analysis', *PloS one*, vol. 12, no. 11, pp. e0186660.

Xie, H., Xie, W., Wu, L., Lin, Q., Liu, M. & Lin, Y. 2019, 'Prediction of short-imminent heavy rainfall based on ECMWF model', *2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom).*

Xie, H., Yang, D., Sun, N., Chen, Z. & Zhang Y. 2018, 'Automated pulmonary nodule detection in CT images using deep convolutional neural networks', *SciencDirect Pattern Recognition*, vol. 85, pp. 109-19.

Xue, N. 2003, 'Chinese word segmentation as character tagging', *International Journal of Computational Linguistics And Chinese Language Processing*, vol. 8, no. 1, pp. 29.

Yang P.Y, Yang Y.H., Zhou B.B. & Zomaya A.Y. 2010, 'A review of ensemble methods in bioinformatics', *Current Bioinformatics*, vol. 5, no. 4, pp. 296-308.

Yu, Z., Haghighat, F., Fung, B.C.M. & Yoshino, H. 2010, 'A decision tree method for building energy demand modeling', *Energy & Buildings*, vol. 42, no. 10, pp. 1637-46.

Zadrozny, B. & Elkan, C. 2001, 'Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers', *Icml*, vol. 1, pp. 609-616.

Zhou, Z. 2012, 'Ensemble methods: foundations and algorithms', *Chapman and Hall/CRC*.

Zhu, B., Yu, Y., Li, C. & Wang, H. 2017, 'Research and implementation of hot topic detection system based on web', *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*.