IDENTIFICATION OF PROTEIN-LIGAND BINDING SITE USING MACHINE LEARNING AND HYBRID PRE-PROCESSING TECHNIQUES

WONG YI KWAN GINNY

Ph.D

The Hong Kong

Polytechnic University

2015

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

IDENTIFICATION OF PROTEIN-LIGAND BINDING SITE USING MACHINE LEARNING AND HYBRID PRE-PROCESSING TECHNIQUES

WONG YI KWAN GINNY

A thesis submitted in partial fulfillment of the requirements

for the Degree of Doctor of Philosophy

January 2015

Certificate of Originality

I herewith declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no materials previously published or written, nor material that has been accepted for the awards of any degree or diploma, except where due acknowledgement has been made in the text.

(Signed)

WONG Yi Kwan Ginny (Name of Student)

Statement of Originality

The following contributions reported in this thesis are claimed to be original.

1. *The predicting method of protein-ligand binding site using Support Vector Machine with protein properties in Chapter 3.* Different protein properties are used as attributes to predict the protein-ligand binding site.

2. *The hybrid pre-processing method based on Synthetic Minority Over-sampling Technique (SMOTE) and CHC in Chapter 4.* The hybrid pre-processing method SMOTE+CHC consists of SMOTE and CHC. SMOTE is first applied to generate new samples of the minority class. CHC is applied to under-sample the synthetic samples and the samples of the majority class. It has the advantages of hybrid methods with a relatively small increase in the size of training sets.

3. *The hybrid pre-processing method based on fuzzy rule base and CHC in Chapter 5.* The hybrid pre-processing method FRB+CHC uses fuzzy rule base to generate new samples of the minority class. CHC is then applied to under-sample the synthetic samples and the samples of the majority class. It gives better performance and improves the robustness.

4. *The under-sampling method based on fuzzy rule base and CHC in Chapter 6.* The under-sampling method uFRB+CHC uses fuzzy rule base to select samples of the majority class. CHC is then applied to further reduce the data size. It improves the performance over large imbalanced datasets.

5. *The predicting method of protein-ligand binding site using FRB +CHC in Chapter 8.* FRB+CHC is applied to solve the imbalanced problem of binding site dataset and gives a significant improvement.

Those who hope in the LORD will renew their strength. They will soar on wings like eagles; they will run and not grow weary, they will walk and not be faint. (Isaiah 40:31)

Abstract

The identification of protein-ligand binding site is an important task in structure-based drug design and docking algorithm. In the past two decades, different approaches have been developed to predict the binding site, such as the geometric, energetic and sequence-based methods. The prediction for these approaches is usually based on some scores, which are defined with a single protein property. Then, a threshold of the scores is set to determine the binding sites. However, it is difficult to set the threshold value even after considering the mean and standard deviation from the practical data.

This thesis investigates the computational prediction of protein-ligand binding sites from the structure and sequence of proteins. The binding site prediction can be formulated as a problem of binary classification: discriminating whether a location is likely to bind the ligand or not. When the scores are calculated from the protein properties, the algorithm for performing classification becomes very important, which affects the prediction results significantly. In this thesis, a Support Vector Machine (SVM) is proposed to classify the pockets that are most likely to bind ligands on considering the attributes of geometric characteristics, interaction potential, offset from protein, conservation score, and properties surrounding the pockets. Different kinds of protein properties are considered to do the classification instead of only one single protein property as used in some published approaches.

First, the grid points near the protein surface are used to represent the locations of binding sites. Our method is compared to eight existing methods on the datasets of LigASite and 198 drug-target complexes. The results show that the proposed method improves

the success rate in terms of F-measure and area under the receiver operating characteristic (AUC). Our method improves the AUC measure from 66 to 81 percent without decreasing the F-measure values, and increases the success rate of locating the binding sites within three largest pockets from 74 to 82 percent. Our method also provides more comprehensive results than the others.

Similar to many datasets in Bioinformatics, the datasets of protein binding sites encounter the problem of being imbalanced and the complexity of doing classification. Re-sampling has become an important step to pre-process the imbalanced data. It aims at balancing the datasets by increasing the samples of the smaller class (the minority class) and/or decreasing the samples of the larger class (the majority class), which are respectively known as over-sampling and under-sampling. Most of the machine learning tools (including SVM) is biased to the majority class, so that the classification of the minority class might not be done satisfactorily. To deal with the imbalanced dataset of binding sites, random under-sampling is used at this stage.

After that, two hybrid pre-processing re-sampling methods and one under-sampling method are proposed. The first one applies Synthesis Minority Over-sampling Technique (SMOTE) to create new samples of the minority class. However, the resulting large sample size will increase the complexity of the classification model. The efficiency of the learning algorithm applied to the classification model will be decreased. Therefore, an evolutionary algorithm (EA) is introduced to further process the synthetic samples and the samples of the majority class for doing under-sampling. The chosen EA is the CHC algorithm. Since the above proposed method is using an existing method (SMOTE) to over-sample the data, the advantages over some previous hybrid methods are not significant. However, it can decrease the over-sampling rate about 50 percent.

Then, the second hybrid pre-processing re-sampling method is proposed, which makes use of fuzzy logic methods to create new samples of the minority class, and CHC as a data cleaning method to the over-sampled dataset. It is found that this pre-processing method can offer an obvious improvement over some previous over-sampling and hybrid methods. From experimental results, our method outperforms the other methods in terms of F-measure and AUC with the lowest over-sampling rate. It also shows its robustness with respect to data complexity.

Large imbalanced datasets have caused many difficulties to the classification problem. Therefore, an under-sampling method is proposed to reduce the data size. It makes use of fuzzy logic to select samples of the majority class, and CHC is employed to further reduce the data size. From experimental results, it can be seen that our proposed method improves both the F-measure and AUC. The complexity of the classification model is also compared. It is found that our proposed method brings the lowest complexity among all methods under comparison.

Finally, a general comparison of the three proposed pre-processing methods is presented. One of the hybrid methods is selected and applied to the datasets for predicting the protein-ligand binding sites. A SVM with the proposed attributes is employed to identify the binding sites. Improvement over our previous method, which does not use the hybrid pre-processing method, is obtained in the testing datasets of 198 drug-target complexes. Improved results over the dataset of 210 bound structures are also obtained. The improvement in success rate is 3 percent and 6 precent respectively. Our method is also compared to five other prediction methods. The results show that our method can have more protein-ligand the binding sites located successfully.

Publication List

Journal Papers

[1] Ginny Y. Wong, Frank H. F. Leung, and Sai-Ho Ling, "Predicting protein-ligand binding site using support vector machine with protein properties," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 6, pp. 1517–1529, Nov. -Dec. 2013.

[2] Ginny Y. Wong, Frank H. F. Leung, and Sai-Ho Ling, "A Hybrid Evolutionary Pre-processing Method for Imbalanced Datasets," *Information Sciences* (Submitted)

Conference Papers

[1] Ginny Y. Wong and Frank H. F. Leung, "Predicting protein-ligand binding site with support vector machine," in *Proceedings of the IEEE Congress on Evolutionary Computation 2010 (CEC 2010)*, pp. 1–5, July 2010.

[2] Ginny Y. Wong, Frank H. F. Leung, and Sai-Ho Ling, "Predicting protein-ligand binding site with differential evolution and support vector machine," in *Proceedings of the International Joint Conference on Neural Networks 2012 (IJCNN 2012)*, pp. 1–6, June 2012.

[3] Ginny Y. Wong, Frank H. F. Leung, and Sai-Ho Ling, "A novel evolutionary pre-processing method based on over-sampling and under-sampling for imbalanced

datasets," in *Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)*, pp. 2354–2359, Nov. 2013.

[4] Ginny Y. Wong, Frank H. F. Leung, and Sai-Ho Ling, "An under-sampling method based on fuzzy logic for large imbalanced dataset," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2014)*, pp. 1248–1252, July 2014.

Acknowledgements

First and foremost, praises and thanks to God, the Almighty, for giving me patience, health, wisdom, and blessing to accomplish my research work.

I would like to express my sincere gratitude to my chief supervisor, Dr. Frank H.F. Leung for his continuous support through my research study. Without his patient and guidance, this work would not have been possible. I would also like to thank my co-supervisor, Dr. Steve S.H. Ling for the helpful advice and discussions during the progress of the research.

My gratitude is extended to my friends, Mr. Benny C.Y. Yeung, Dr. Johnny C.Y. Lai, and Mr. Ivan C.F. Lau for providing help on my work and a delightful life in graduate school.

I am deeply thankful to my grandmother who has passed away three years ago, for her unwavering love and support. I cannot overstate her importance to me. I am extremely grateful to my parents and my brother for their steadfast love, caring, and encouragement. My heartfelt thanks are due to my boyfriend Mr. C.K. Cheung for his understanding, prayers, and continuing support. In addition, I express my thanks to my brothers and sisters in Christ for their support and valuable prayers.

I would also like to show my great appreciation to the staff of the General Office of the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, for their kindness.

The work described in this thesis was substantially supported by a grant from Hong Kong Polytechnic University of the Hong Kong Special Administrative Region (Project Account Code RPKP and G-YL79).

Contents

List of Figures

List of Tables

Nomenclature List

Unless otherwise specified, some commonly-used symbols in the thesis are defined as follow.

C	Regularization parameter of SVM for controlling the tradeoff between training and margin
C_{factor}	Cost-factor of SVM
$K(\cdot)$	Kernel function of SVM
σ	Parameter of the RBF of SVM
FP_{rate}	False positive rate
N_{HIT}	Number of proteins that at least one binding sites can be located correctly
N_P	Total number of proteins
$Rateover$	Over-sampling rate
$Nsampled$	Number of samples in the re-sampled training set
$Noriginal$	Number of samples in the original training set
F_X	F-measure of chromosome X
F_Y	F-measure of chromosome Y
A_X	AUC of chromosome X
A_Y	AUC of chromosome Y
A_β^θ	Fuzzy term
$\mu_{A_\beta^\theta}(\cdot)$	Fuzzy value
w_θ	Fuzzy rule weight

Abbreviations

Unless otherwise specified, some commonly-used abbreviations in the thesis are defined as follow.

3D	3-Dimensional
AUC	Area Under the receiver operating characteristic Curve
CHC	Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation
CNN	Condensed Nearest Neighbor rule
CS	Connolly Surface
EA	Evolutionary Algorithm
ENN	Edited Nearest Neighbor rule
FN	False Negative
FP	False Positive
FRB	Fuzzy Rule Base
IR	Imbalanced Ratio
JSD	Jensen-Shannon Divergence
NCL	Neighborhood Cleaning Rule
NN	Nearest Neighbor
OSS	One-Sided Selection
PASS	Putative Active Site with Spheres
PDB	Protein Data Bank
PSP	Protein-Solvent-Protein
RBF	Radial Basis Function

ROS	Random Over-Sampling
RUS	Random Under-Sampling
SMOTE	Synthetic Minority Over-sampling TEchnique
SVM	Support Vector Machine
TL	Tomek Links
TN	True Negative
TP	True Positive

Chapter 1

Introduction

1.1 An Introduction to Drug Design

1.1.1 General Background

In this thesis, drugs refer to a single or combination of small molecules (e.g. ligands) that activates or inhibits the function of a biomolecule to realize a therapeutic effect. This process of drug design is an expensive and time-consuming activity, where failures are expected. On average, it costs US$800 million and 14 years [1], [2] for bringing a new drug to the market. Therefore, many companies carry out a large amount of projects in the early stages and select only a few to go forward at each stage. The later stages are much more expensive and time-consuming than the early stages; hence, the projects selection and management are important.

The modern process of new drug identification can be divided into two phases: drug discovery and drug development. Drug discovery includes target identification, identification of a compound to bind the target with desired effect, optimizing the affinity and

selectivity of those compounds, and optimizing the drug-like properties with sufficient affinity and appropriate *in vivo* activity. Once the compound meets the required criteria of *in vivo* efficacy, the process of drug development with pre-clinical stage and clinical trials will begin.

The drug discovery process starts with target identification and validation. This operation searches the causes of the phenotype of the disease. Protein plays a critical role in causing the symptoms of a human disease. Activating or inhibiting its function can have a positive effect on the disease [3]. After the relationship between the target (protein) and disease has been found, the next operation of drug discovery is to find a method to modify that target. This consists of protein-protein and protein-ligand (small chemical molecule) interactions.

Traditionally, drug discovery relies on trial-and-error testings of chemical substances and matching the apparent effect to the treatments. Thanks to the rapid growth of computational chemistry, the drug candidate can be determined by testing hundreds of chemical substances or designed by computer virtually, which save time and cost as compared with the traditional way. This process is generally called computer-aided drug design.

1.1.2 Computer-Aided Drug Design

There are mainly two types of computer-aided drug design. One is ligand-based, which relies on the knowledge of ligands and their particular characteristics. This type is suitable if no structural information about the target is available. Another one is structure-based, which relies on the knowledge of the structure of targets. This type is commonly

used since lots of protein structure have been modeled nowadays. Based on the structure of a target protein, different approaches can be used to find the ligand, such as virtual screening, docking, and *de novo* drug design [4].

1.1.3 Structure-Based Drug Design

Taking advantage of the three-dimensional (3D) structure of a given protein, structure-based drug design (SBDD) attempts to contribute to drug discovery [5]. The 3D structure of a protein can be obtained experimentally with x-ray crystallography or Nuclear Magnetic Resonance (NMR) spectroscopy. Another method is to construct the protein based on its amino acid sequence and a similar protein with a known 3D structure. All this information can be found from the Protein Data Bank (PDB) [6] or Protein Quaternary Structure file server (PQS) [7], which show the atomic coordinates and the quaternary structure of proteins respectively. This has made SBDD more and more feasible because the 3D atoms' arrangements of proteins allow the prediction of protein-ligand binding sites, which is an important prerequisite of SBDD [8]. One famous example use of SBDD is the inhibition of the HIV protease. The drug is highly effective against HIV [9].

1.2 Motivation

This thesis focuses on the identification of protein-ligand binding sites, which directly participate in the interaction of target (protein) and molecules (ligand). The identification of protein-ligand binding site is an important step of SBDD. There are three main

representations of the binding sites. They are the grid points surrounding the protein [10], residues contacting the ligand [11], and molecular surface of the protein [12]. The binding site prediction can be formulated as a binary classification problem: discriminating whether grid point / residue / molecular surface is likely to bind the ligand or not.

1.2.1 Problems of Determinate Methods

In this thesis, the binding sites (pockets) are represented by grid points surrounding the protein. As the binding sites are usually found at the clefts on the surface of proteins, many previous works assigned some scores to the grid points based on different protein characteristics, and tried to predict the binding sites using these scores. There are several determinate methods after the corresponding scores of the grid points are calculated. The simplest one is to apply a threshold to the grid point value to determine if the grid point belongs to a pocket [13]. This threshold is set to all proteins and does not consider the difference among them. A poor scenario may cluster most of the grid points as pockets if the threshold is set too low, or the number of pockets is much smaller than that of binding sites if the threshold is set too large.

Another method calculates the mean and standard deviation of the grid points' values to determine the threshold for each protein [14]. Although this approach calculates the threshold for different proteins, the threshold depends on the grid points' values. If the grids embedded in the protein vary, the mean and standard deviation of the grid points' values will be different. That means, the threshold and the number of pockets could be varying for a particular protein used.

In [15], a binary search for the grid threshold is performed. The binary search produces a culled set of pockets, which have specified properties based on the sizes and shapes of the pockets. When the method iterates, the grid points are adjusted until the set of pockets meet all the properties. Although this approach can consider the sizes and shapes of the pockets, all the grid thresholds are set by the users, and we do not know which values of thresholds are suitable for a given protein. To overcome the weakness of the above determinate methods, the Support Vector Machine (SVM) is proposed and will be covered in this thesis.

1.2.2 Problems of Imbalanced Dataset

Like most of the datasets in bioinformatics, the datasets of binding sites have the problem of being imbalanced [16], which is a popular topic in recent years [17]. The imbalanced dataset problem increases the difficulty of determining binding sites with SVM since most of the machine learning tools, including neural networks and SVMs, are originally designed for well-balanced datasets. If the dataset is imbalanced, the performance of the classifier can be poor. The reason for this is apparent. For example, considering a dataset with 99% of data from class A and only 1% of data from class B, the accuracy is 99% if the classifier ignores the data from class B and labels the whole dataset as class A. It is already very hard to achieve an accuracy above 99% by using most of the learning algorithms. However, the minority class in datasets is usually more important and meaningful. For example, there are much less samples of people with a particular disease than those of healthy people in a medical problem. If a classifier is needed to label whether some people are infected or not, it is obvious that the minority class (people with a particular disease) is the more interested class.

Problems with imbalanced datasets can be easily found in the real world, such as the detection of oil spills from satellite images [18], spotting customers for telecommunications management [19], and the identification of power distribution fault causes [20]. There are two main approaches to solve the problems caused by imbalanced datasets. One is the data level approach and the other is the algorithm level approach. The data level approaches in [21]–[23] include balancing the class distribution by over-sampling the minority class or under-sampling the majority class. The algorithm level approaches improve the existing machine learning methods by adjusting the probabilistic estimate [24], modifying the cost per class [25], adding some penalty constants [26], or learning from one class instead of two classes [27].

Many experiments [28] show that re-sampling is a good data level pre-processing approach to handle imbalanced data. Moreover, preprocessing approaches are more flexible since they are independent of the chosen classifier. Therefore, re-sampling approaches are focused in this thesis. There are three main types of strategies for re-sampling data. They are over-sampling, under-sampling, and hybrid methods, which combine the two previous methods (over-sampling and under-sampling methods). The importance of designing sampling strategies has been discussed in [29], which can lead to successful learning of different classes.

1.3 Contributions

This thesis presents the predicting methods of protein-ligand binding sites and the pre-processing methods of imbalanced dataset. Firstly, SVM is employed to predict the binding sites using different protein properties. To solve the imbalanced problem in the

protein datasets, different pre-processing methods are investigated. Then, two hybrid pre-processing methods and one under-sampling methods are proposed. Based on the performance, one of the hybrid methods is applied on the protein datasets and improved results of the prediction of binding sites are shown. The main contributions are given as follows.

Firstly, SVM is proposed to predict the protein-ligand binding sites and handle the problems mentioned in Section 1.2.1. The prediction of binding sites can be expressed as a problem of binary classification. SVM [30]–[33] is one of the supervised learning tools for doing classification. It has shown its high applicability and advantage on classifying high-dimensional and large datasets [34], [35].

The SVM is trained to generate the hyperplane by using 29 proteins' properties, including the geometric characteristics, interaction energy, sequence conservation, distance from protein, and the properties of the surrounding grid points. A radial basis function (RBF) is used as the SVM kernel, which is a common kernel for non-linear classification. Like most of the datasets in bioinformatics, the data of the binding sites have the problem of being imbalanced [16]. Therefore, random under-sampling and filtering are applied to reduce the data size.

Two experiments are used to evaluate our approach. The first one uses LigASite [36] as the dataset which is suggested in ConCavity [15]. Our approach is compared with four other methods. They are LIGSITE [13], PocketFinder [14], ConCavity [15], and SURFNET [37]. The other experiment uses 198 drug-target dataset which is developed in MetaPocket [38]. Only the location of the three largest binding sites are selected as the potential pockets, which are represented by the center points of these pockets,

for the evaluation in this experiment. There are sometimes more than one binding site within a protein or the prediction can identify more than one binding site. In this case, only the correct prediction at the larger pocket is counted when the success rate is calculated. Our approach is compared with six other methods. They are ConCavity [15], SURFNET [37], MetaPocket [38], LIGSITE^{csc} [39], Fpocket [40], and Q-SiteFinder [41]. Two different measurements are applied since the representations of the binding sites are different in these experiments.

On data sampling, three different pre-processing methods are proposed. Two of them are hybrid methods and the other one is an under-sampling method. Both hybrid methods have two stages: over-sampling first and under-sampling next. Their under-sampling stage uses the same approach. The size of training dataset is increased after the first stage of over-sampling, which causes over-generalization easily. It increases the complexity of the classification model, and decreases the efficiency of the learning algorithm. Therefore, an evolutionary algorithm (EA) is applied to both the synthetic samples and majority samples to under-sample the dataset. The chosen EA is the CHC (Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation) algorithm [42] because it can select the smallest and most representative instances among many algorithms as reported in [43]. The main difference between the two proposed hybrid methods is that the first one uses the Synthetic Minority Over-sampling Technique (SMOTE) to generate new samples of the minority class (SMOTE+CHC), while the second one generates the new samples based on a fuzzy rule base (FRB+CHC).

The third proposed pre-processing method using under-sampling aims at solving the problem of large imbalanced dataset. It first selects the samples of the majority class based on fuzzy logic. To further reduce the data size, the evolutionary computational

method of CHC [42] is suggested. This method is named as uFRB+CHC.

Finally, FRB+CHC is selected as the pre-processing method for the dataset of the protein-ligand binding sites because of its robust performance. SVM is employed to train the re-sampled dataset. Two benchmark datasets are used to evaluate this method: the 198 drug-target complexes and 210 bound structures. Both of them are developed in MetaPocket [38]. It is first compared to our previous work, which used random under-sampling as the pre-processing method. Then, this approach is compared with five other methods. They are LIGSITE^{CSC}, PASS [44], SURFNET, Q-SiteFinder, and MetaPocket.

1.4 Outline of the Thesis

The main content of this thesis is organized into eight chapters. In Chapter 2, a literature review on different predicting methods of protein-ligand binding sites and pre-processing methods is given. In Chapter 3, the proposed predicting method of protein-ligand binding site is introduced. It uses an SVM and 29 protein properties to predict the binding sites. Chapter 4, Chapter 5, and Chapter 6 give the details of SMOTE+CHC, FRB+CHC, and uFRB+CHC respectively. They are all proposed pre-processing methods for imbalanced datasets. Chapter 7 shows a general comparison of these three proposed pre-processing methods. In Chapter 8, FRB+CHC is applied on the protein dataset to improve the results in Chapter 3. In the final chapter, a conclusion is drawn and some potential directions for further work are also given.

Chapter 2

Literature Review

A review on the previous works on protein-ligand binding sites prediction and re-sampling methods is given in this chapter.

2.1 Prediction of Protein-Ligand Binding Sites

The protein-ligand binding sites are commonly located in the clefts on the surface of proteins. However, not all the clefts are identified as the potential binding sites (pockets), as we need to examine the pockets' size, the interaction energy of the surrounding protein atoms, and the sequence conservation of the contacted residues. The following sections are divided into three parts. The first part describes the studies that mainly use the geometric characteristics of a protein to predict the pockets. The second part describes the studies that use the energy criteria by calculating the van der Waals interaction potential to do the prediction. The last part describes the studies that integrate the sequence conservation with the structural information of a protein to identify the potential pockets.

2.1.1 Geometry-Based Methods

POCKET [45] is one of the geometry-based methods to define the binding sites. Firstly, a 3D grid is generated. Secondly, a distance check is applied on the grid to make sure the atoms of protein do not overlap with the grid point. All the grid points, which do not overlap with the atoms of protein, are labeled as solvent. If the grid points outside the protein are enclosed by the protein surface in opposite directions of the same axis (i.e. the grid points are enclosed by pairs of atoms within the protein), it is called a protein-solvent-protein (PSP) event (Fig. 2.1).

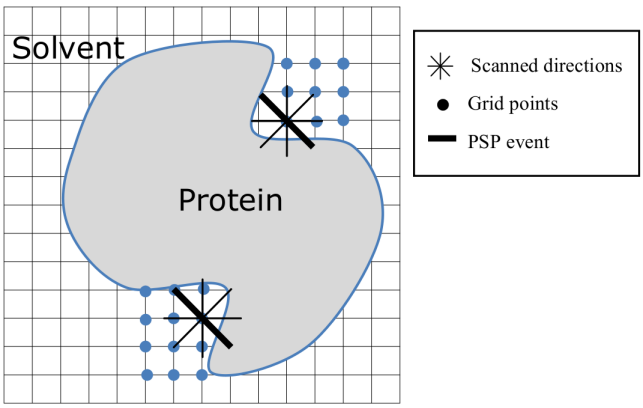


Figure 2.1: PSP event used to describe the geometric feature of a grid point. It counts the number scanning directions that pairs of protein atoms can enclose the grid point. For the POCKET method, the maximum number of PSP event is three while it is seven for the LIGSITE method.

LIGSITE [13] is an extension of POCKET with more scanning directions. Both of them considered the identification of PSP events on the basis of atom coordinates. LIGSITE scans for the pockets along three axes and four cubic diagonals while POCKET only scans along three axes. The number of PSP events occurred in the scanning directions are counted and the value is assigned to each grid point of the protein. The grid points with higher values are more likely to be the pockets. Therefore, when the grid point's

value exceeds a certain threshold, it can be considered as a pocket. Fig. 2.1 shows the PSP events of two enclosed grid points.

SURFNET [37] is another geometry-based method to define the binding sites. Like LIGSITE, a 3D grid is generated first. The grid values of SURFNET are calculated by counting the number of constructed spheres. Firstly, pairs of relevant atoms are taken within the protein. Then, testing spheres are formed between the pairs. If the sphere overlaps with other atoms, the radius decreases until no overlapping occurs (Fig. 2.2). Only the distance between two atoms within 10 Å is considered. The sphere of radius smaller than 1.5 Å is also ignored. If the grid points are out of the pockets, the distances between pairs of atoms are very large or cannot be found. On the contrary, if the grid points are inside the pockets, more than one sphere can be formed.

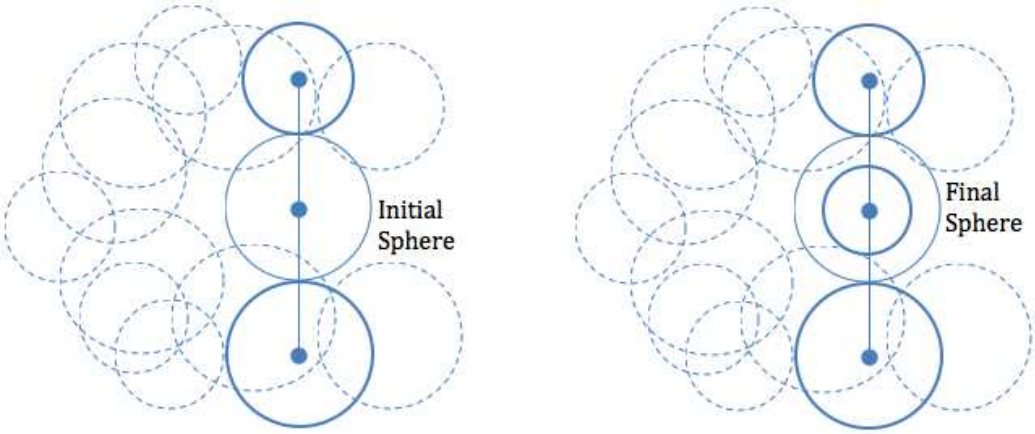
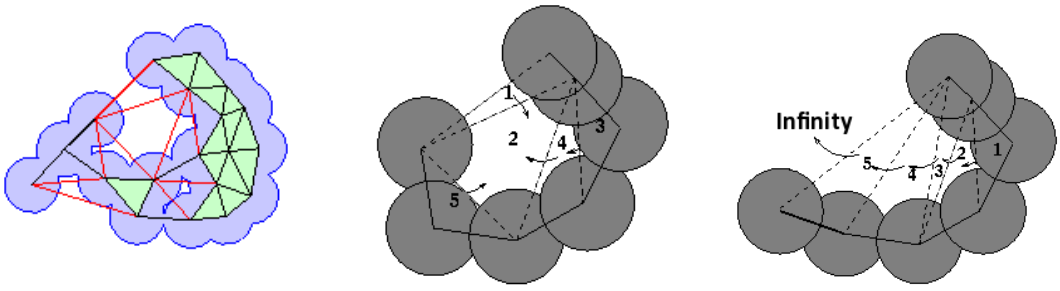


Figure 2.2: SURFNET. There are three solid line circles and several dotted line circles in each graph. The top and bottom solid line circles represent the pair of relevant atoms and the middle one shows the constructed sphere of a grid point. The dotted line circles represent the other atoms that surround the grid point under testing. The initial sphere in the graph on the left overlaps with other atoms; therefore, its radius decreases until no overlapping occurs to form the final sphere in the graph on the right.

CASTp (Computed Atlas of Surface Topography of proteins) [46] is based on Delaunay triangulation, alpha shape, and discrete flow [47]–[49]. Delaunay triangulation is used

to represent the surface of a protein and a convex hull is formed containing all atom centers inside. To obtain the alpha shape, the edges of Delaunay triangulation that are outside the protein are omitted. A triangle with one or more edges omitted is called an "empty" triangle. Fig. 2.3(a) shows the alpha shape of the protein and the solid line in red color is the omitted edges of Delaunay triangulation.

The discrete flow is applied to determine the pockets and defined only for the empty triangles. An obtuse empty triangle flows to its adjacent triangle, whereas an acute triangle collects the flow from adjacent triangles. Fig. 2.3(b) shows a pockets formed by five empty triangles. All the triangles are obtuse, except triangle 2. Therefore, triangle 2 collects all the flows from other triangles and this cleft is identified as the potential binding site. If a cleft consists of obtuse empty triangles only, the triangles will flow sequentially to infinity. This type of cleft is not identified as a binding site. Fig. 2.3(c) shows an example of this kind of cleft, which is formed by five obtuse empty triangles. They flow from triangle 1 to triangle 5 and then to the infinity.



(a) Alpha Shape.

(b) A pocket is composed by five empty triangles.

(c) This type of cleft is not identified as a binding site.

Figure 2.3: CASTp

Fpocket [40] uses the concept of alpha sphere, which is a sphere contacting with four protein atoms on its boundary and not containing any atoms inside. The size of these alpha spheres is used to determine the property of the sphere locations. Small spheres

are located inside the protein, whereas large spheres are located outside the protein. For the clefts or pockets, the size of the spheres is intermediate. Therefore, a filtered collection of spheres are identified by measuring the radii of the alpha spheres. Fpocket then clusters the spheres close together and identifies them as potential pockets.

PASS (Putative Active Site with Spheres) [44] looks at all unique triplets of protein atoms. If they are close together, two possible probe spheres, which lie tangential to all three protein atoms, will be calculated. It results in an initial layer of spherical probes coating the protein. These probes are filtered if they clash with any protein atoms and two probes lie too close to each other. In addition, the burial count of each probe is calculated by measuring the number of protein atoms found within 8Å radius of the probe. The probes with a low burial count are most likely outside the pockets. Therefore, a burial count threshold is applied to eliminate the probes outside the pockets.

After the initial layer of probes is formed, additional layers of spheres are accreted onto the existing probes and the filtration to the new probes as described above is followed. These cycles of addition and filtration are repeated until the number of probes bounded to the protein is no longer changed. PASS then assigns a weight to each probe and identify the central probes in regions that contain many spheres with high burial count (Fig. 2.4).

2.1.2 Energy-Based Methods

PocketFinder [14] is an energy-based method for ligand binding site prediction. It uses the van der Waals interaction energy between the protein atoms and the simple atomic

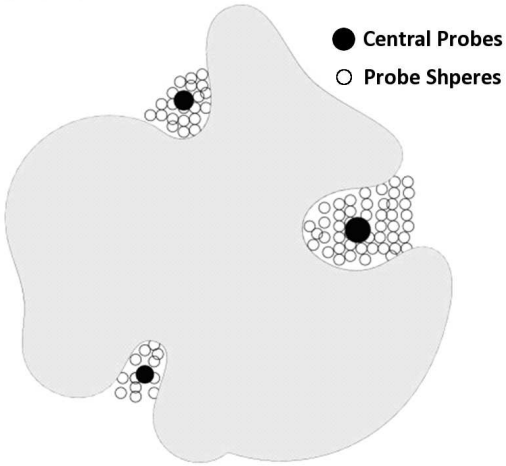


Figure 2.4: PASS. The black dots represent the the central probes of each potential pockets and the empty circles surrounding are the retained spheres after the repeated cycles of addition and filtration.

probes to locate the binding sites with high energy. A 3D grid potential map is generated first. The potential of each grid point p is calculated by the Lennard-Jones formula:

$$V(p) = \sum_{i=1}^{N} (\frac{C_{12}^{i}}{r_{pi}^{12}} - \frac{C_{6}^{i}}{r_{pi}^{6}}) \tag{2.1}$$

where C_{12}^{i} and C_{6}^{i} are constants, which are the typical 12-6 Lennard-Jones parameters used to model the van der Waals interaction energy between a carbon atom placed at the grid point p and the protein atom i; N is the total number of protein atoms. r_{pi}^{12} and r_{pi}^{6} are the 12-th and 6-th powers of r_{pi} respectively, where r_{pi} is the distance between the grid point p and the protein atom i. The first term describes the repulsion between atoms when they are very close to each other. The second term describes the attraction between atoms at long distance.

The grid points are then filtered with a threshold, which is determined by the mean and standard deviation of the grid points' values. Those filtered grid points are clustered into many groups and the potential pockets are identified as those groups with a large size.

Q-SiteFinder [41] uses a concept similar to PocketFinder (described above) to define the potential pockets. It uses methyl (–CH3) probes, which are initialized on the protein surface, to calculate the van der Waals interaction energy between the protein atoms and the probes. Those probes with high interaction energy are retained and clustered into many groups. The clusters with a large size and high energy are considered as potential pockets.

2.1.3 Sequence Conservation

Residues in a protein are the individual organic compounds called amino acids. As not all residues in a protein are equally important, conservation analysis becomes a very useful method to indicate those functionally important residues in the protein sequence [50]–[52]. Conservation analysis has been shown to be strongly correlated with the prediction of ligand binding sites [53], [54]. It is a scoring method that involves the conservation of sequentially adjacent residues. Therefore, some studies suggested combining the sequence conservation and the structure of protein to predict the protein ligand binding sites through weighting every pair of protein atoms [15], [39].

LIGSITE^{CSC} [39] is an extension of LIGSITE. It uses the Connolly surface [55] and defines surface-solvent-surface events, instead of PSP events defined in POCKET and LIGSITE. In the Connolly algorithm, a probe sphere of typical 1.4Å radius rolls over the protein. The Connolly surface consists of the van der Waals surface of the protein, which is touched by the probe sphere, and the probe sphere surface when it is in contact with more than one protein atom. Fig. 2.5 shows the difference between the Connolly surface and the van der Waals surface. LIGSITE^{CSC} scans the three axes and four cubic

diagonals for the surface-solvent-surface events. The pockets with high values of such events are re-ranked by the degree of conservation of the involved surface residues.

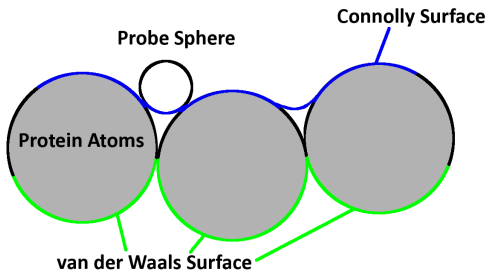


Figure 2.5: The van der Waals surface in green is the outer surface of a protein. It separates the inner space from the outer space. The Connolly surface in blue is composed of two parts. One is the van der Waals surface of the protein, which contacts with the probe sphere. The other one is the probe sphere surface when it contacts with more than one protein atom.

ConCavity [15] consists of three steps to predict the binding sites: grid creation, pocket extraction, and residue mapping. Like LIGSITE, SURFNET, or PocketFinder, a 3D grid surrounding a given protein is created first. The structural properties and sequence conservation of the protein are used to assign a value to each grid point. Those grid points with high values are retained and clustered into many groups. A binary search is then performed and produces a culled set of pockets, which have specified properties based on the sizes and shapes of the pockets. After several searches, the grid points are adjusted with the set of pockets having all the properties. Finally, the retained grid points are mapped to the surface of the protein and scored the protein residues.

MetaPocket [38], [56] is a combination of eight predictors, including ConCavity [15], SURFNET [37], LIGSITE^{CSC} [39], Fpocket [40], Q-SiteFinder [41], PASS [44], GHE-

COM [57], and POCASA [58]. A given protein is first sent to the predictors to identify the pocket sites. A z-score, which is a statistical measurement to find out the probability of a score occurring within normal distribution, is calculated separately for each pocket in different methods in order to compare the ranking scores of the pockets. Then, only the three highest-scored pockets of each method are selected. Totally 24 retained pockets are clustered according to their spatial similarity. The total z-score of each cluster is calculated and used to re-rank the final pockets. Finally, the potential ligand binding residues are figured out based on the final pockets.

2.2 Re-sampling Methods

Re-sampling is a common approach to handle the imbalanced dataset. There are three main strategies of re-sampling. The first one is over-sampling, which generates some new instances of the minority class, and the second one is under-sampling, which eliminates some samples of the majority class. The above two strategies artificially re-balance the class distribution. However, this kind of re-balancing may not solve the problems of some imbalanced datasets.Therefore, hybrid methods, which combine the two previous methods (over-sampling and under-sampling methods), are also considered.

2.2.1 Over-sampling Methods

Random over-sampling (ROS) is a non-heuristic method that replicates samples of the original minority class to generate the new instances. This method causes over-fitting easily since the new instances copy exactly from the original minority class.

Synthetic Minority by Over-sampling TEchnique (SMOTE) [22] creates the new instances by interpolating several minority samples that join together. This method makes use of each minority class sample and inserts synthetic samples along the line segments joining any/all of the k minority class nearest neighbors to over-sample the minority class. The synthetic samples are randomly chosen among the neighbors from the k nearest neighbors, depending upon the degree of over-sampling required. An example is shown in Fig. 2.6. Five nearest neighbors are used in it, where x_i is a selected sample of minority class, x_{i1} to x_{i5} are the 5 nearest neighbors of x_i and s_1 to s_5 are the synthetic samples created by interpolation. If the degree of over-sampling required is 300%, three synthetic examples are selected randomly from s_1 to s_5.

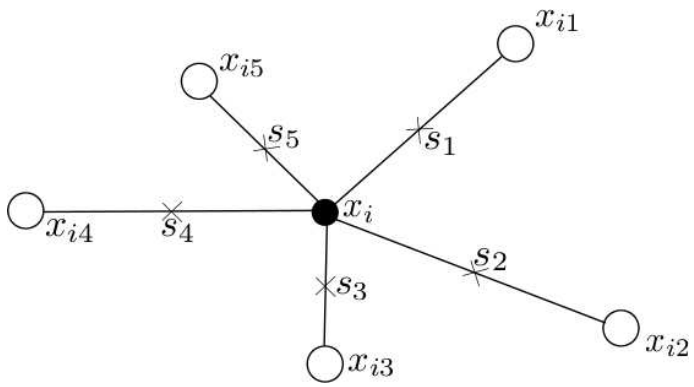


Figure 2.6: Example of SMOTE with 5 nearest neighbors.

Each sample can be represented by a feature vector. Synthetic samples are generated in the following steps. Firstly, the difference between the feature vector of the selected sample and that of its neatest neighbor is calculated. Then, this difference is multiplied by a random number between 0 to 1. Finally, this value is added to the feature vector of the selected sample. Therefore, the final value should lie along the line segment at a random position between 2 specific vectors. Since the synthetic samples provide a less specific and larger decision regions, the over-fitting problem can be reduced. However,

this method may introduce more minority synthetic samples in the area of majority class where the minority class is very sparse with respect to the majority class. This causes the problem of overgeneralization [59], which means the decision boundary is very narrow or there is large overlapping area between the majority class and minority class.

2.2.2 Under-sampling Methods

Random under-sampling (RUS) is a non-heuristic method that aims to balance the datasets by randomly removing samples of the majority class. This method may easily remove some useful data.

Condensed nearest neighbor rule (CNN) [60] eliminate the majority class samples that are distant from the decision border since these samples can be considered as less relevant for learning. First, a majority class sample is randomly drawn and formed a subset with all the minority class samples. Then, 1-NN is used over this subset to classify the other majority class samples. Every misclassified majority sample is selected to form the re-sampled majority dataset.

Tomek links (TL) [61] is opposite to CNN. It edits out noisy and borderline majority class samples. Borderline samples can be treated as unsafe samples since only small changes can cause them to be assigned to a wrong class. The process can be described as follows. First, each sample is used to find another sample which has the minimum distance between them. If these two samples are in different classes, the sample of majority class will be removed. This method can remove the noisy and borderline majority class samples to increase the area of decision border. However, some useful data, which

is important for the classification, may also be discarded.

One-sided selection (OSS) [62] applies TL followed by CNN. By combining the advantages of both methods, the remainder majority samples are safe and more relevant for learning.

Neighborhood Cleaning Rule (NCL) [63] is modified the Wilson's Edited Nearest Neighbor Rule (ENN) [64] to remove majority class samples. Firstly, three nearest neighbors of each sample in the training set are found. If the selected sample belongs to the majority class but the three nearest neighbors classify it wrongly, the selected sample will be removed. If the selected sample belongs to the minority class but the three nearest neighbors classify it wrongly, the nearest neighbors belonging to the majority class will be removed.

CHC [42] is a kind of EAs that combines a selection strategy with a highly disruptive recombination operator. To avoid premature convergence and maintain diversity, incest prevention and cataclysmic mutation are introduced. The process of CHC can be described as follows. Firstly, a population set of chromosomes P is created. Each chromosome $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ is an n-dimensional vector, which is a set of genes, where p_{ij} is the jth gene value ($j = 1, 2, \ldots, n$) of the ith chromosome in the population ($i = 1, 2, \ldots, m$); m is the population size and n is the number of genes.

Secondly, the chromosomes are evaluated by a defined fitness function. The form of fitness function depends on the application. Thirdly, an intermediate population set of chromosomes C, which is of the same size as P is generated by copying all members of P in a random order.

Then, a uniform crossover (HUX) operator is applied on C to form C'. HUX exchanges half of the genes randomly between the parents. CHC also uses an additional method for incest prevention. Before applying HUX to the parents, the Hamming distance between them is calculated. If half of that distance is larger than a difference threshold d, HUX is applied; otherwise these two parents are deleted from C. The initial threshold d is set at $n/4$. After C' has formed, it is evaluated by the fitness function and an elitist selection is taken. Only the best chromosomes from both P and C' are selected to form the offspring population in the next generation. If the offspring population is the same as P, the difference threshold d is decreased by one.

CHC is different from the traditional genetic algorithm. Mutation is not performed at the recombination stage. CHC performs partial reinitialization (divergence) when the search becomes trapped (i.e., the difference threshold d becomes zero and no new offspring population is formed for several generations). The population is reinitialized, based on the best chromosome, by changing the elements' values randomly with a user-defined divergence rate D_{rate}. For example, if $D_{rate} = 0.35$, the values of 35% elements will be changed randomly. The search is then resumed with a new difference threshold $d = D_{rate} * (1 - D_{rate}) * n$. This process is called cataclysmic mutation.

CHC has shown the ability of selecting the most representative instances among the other algorithms studied in [43]. Therefore, it is chosen as the under-sampling algorithm [65]–[66].

2.2.3 Hybrid Methods

Although both over-sampling and under-sampling can balance the class distribution, different kinds of drawbacks are also introduced, including overgeneralization and removal of useful data. Therefore, hybrid methods are introduced such that both over-sampling and under-sampling are used to tackle the problems.

SMOTE+Tomek links (sTL) [21] is one of the hybrid methods. As discussed above, although SMOTE can reduce the problem of over-fitting and balance the class distribution, it may introduce synthetic samples too deeply in the area of majority class. Therefore, this method applies TL to the over-sampled training set as a data cleaning method. Instead of removing only majority class samples, TL used here removes samples of both classes.

SMOTE+ENN (sENN) [21] is similar to sTL. This method applies ENN as the data cleaning method. ENN used in this method is different from NCL mentioned previously. Instead of removing only majority class samples, ENN removes samples of both classes. Therefore, any sample that is misclassified by its three nearest neighbors is removed.

SMOTE+Rough Set (sRST) [67] applies the rough set theory (RST) as the data cleaning method to include the original samples and the synthetic minority samples that belong to the lower approximation for their class in the final training set.

Borderline-SMOTE (sBorder) [68] is a modified version of SMOTE. It only oversamples or strengthens the borderline minority samples, which can be found by using the

following method. Firstly, m nearest neighbors of each minority sample in the training set are calculated. The number of majority samples among these m nearest neighbors is denoted by m', where $0 \leq m' \leq m$. If $m = m'$, the selected minority sample will be considered as noise. If $m/2 \leq m' < m$, which means that the number of majority samples among the nearest neighbor is larger than that of minority samples, the selected sample will be considered as being misclassified easily, and it will become one of the borderline minority samples. If $0 \leq m' < m/2$, the selected sample will be considered as safe. After all the borderline minority samples have been found, only these samples will undergo the SMOTE process continuously to generate the synthetic samples.

Safe-Level-SMOTE (sSafe) [69] is also a modified version of SMOTE. It assigns each minority sample its safe level before the SMOTE process. Firstly, m nearest neighbors of each minority sample in the training set are calculated. The safe level of the selected sample can be calculated by counting the number of minority samples among these m nearest neighbors. If the safe level of a sample is close to 0, it will consider as noise. If it is close to m, it will be considered to be safe. The synthetic samples are only generated in safe regions.

Chapter 3

Predicting Protein-Ligand Binding Site using Support Vector Machine with Protein Properties

3.1 Introduction

Identification of protein-ligand binding site is an important task in structure-based drug design and docking algorithms. In the past two decades, different approaches have been developed to predict the binding site, including the geometric, energetic and sequence-based methods. While scores are calculated from these methods, the algorithm for doing classification becomes very important and can affect the prediction results greatly. In this chapter, the Support Vector Machine (SVM) is used to cluster the pockets that are most likely to bind ligands under the consideration of attributes of geometric characteristics, interaction potential, offset from protein, conservation score and properties surrounding the pockets.

The binding sites predictions of the previous methods were based on different scores, which were calculated from some protein characteristics. The simplest method was setting a threshold to help determining the binding sites [13]. If the score of a point was greater than the threshold, that point would be identified as the binding site. In [14], mean and standard deviation of the scores were considered on finding the threshold. The results of these approaches are easily affected by the grid format and the threshold has to be set carefully; otherwise the results would not be satisfactory. Machine learning techniques have been widely applied in bioinformatics and have shown satisfactory performance in binding site prediction [70]–[73]. In this chapter, support vector machine (SVM) is proposed for tackling this problem. SVM [30]–[33] has shown its high applicability and advantage on classifying high-dimensional and large datasets [34], [35].

The prediction of binding sites can be formulated as a problem of binary classification to determine a location for binding the ligand. SVM is one of the supervised learning tools and it mainly applies two techniques to do the classification: the formulation of a large-margin hyperplane and the use of a kernel function. SVM can construct an $(n-1)$-dimensional hyperplane in an n-dimensional space to separate the data, where each datum is represented by an n-dimensional vector.

We train the SVM to generate the hyperplane by using 29 proteins' attributes, including the geometric characteristics, interaction energy, sequence conservation, distance from protein, and the properties of the surrounding grid points. A radial basis function (RBF) is used as the SVM kernel since a non-linear classification model is needed and RBF is a common kernel to handle this problem. Like most of the datasets in bioinformatics, the data of the binding sites have the problem of being imbalanced and in large data scales [16]. Therefore, down sampling and filtering are also applied to reduce the data size.

Two experiments are conducted to evaluate our approach. The first one uses LigASite [36] as the dataset which is suggested in ConCavity. The predicted binding sites are represented as grid points in this experiment. Our approach is compared with four other methods. They are LIGSITE, SURFNET, PocketFinder and ConCavity. The other experiment uses 198 drug-target dataset which is developed in MetaPocket [38]. Only the location of the top three largest binding sites are predicted, and each site is represented as one center point in this experiment. Our approach is compared with six other methods. They are LIGSITE^{CSC}, SURFNET, Fpocket, Q-SiteFinder, ConCavity, and MetaPocket. Two different measurements are applied since the representations of the binding sites are different in these experiments.

3.2 Methodology

This section explains the datasets and attributes used in this chapter, and the details of the SVM classifier. The overall flow of our method is then described.

3.2.1 Datasets

We have used two sets of proteins to evaluate our method. The first one is the non-redundant LigASite (v9.4) dataset [36], which was suggested in [15]. The other one is the 198 drug-target complexes, which was discussed in [38]. For the dataset of LigASite, only six main classes of enzyme (categorized for 272 protein complexes) from the dataset are selected. They are transferase, hydrolase, oxidoreductase, lyase, ligase and isomerase, which occupy around 70% of LigASite. Fig. 3.1 shows the percentages of the

number of proteins distributed among these six enzyme classes. For reference, Fig. 3.2 shows the number of chains distributed in the selected proteins of LigASite.

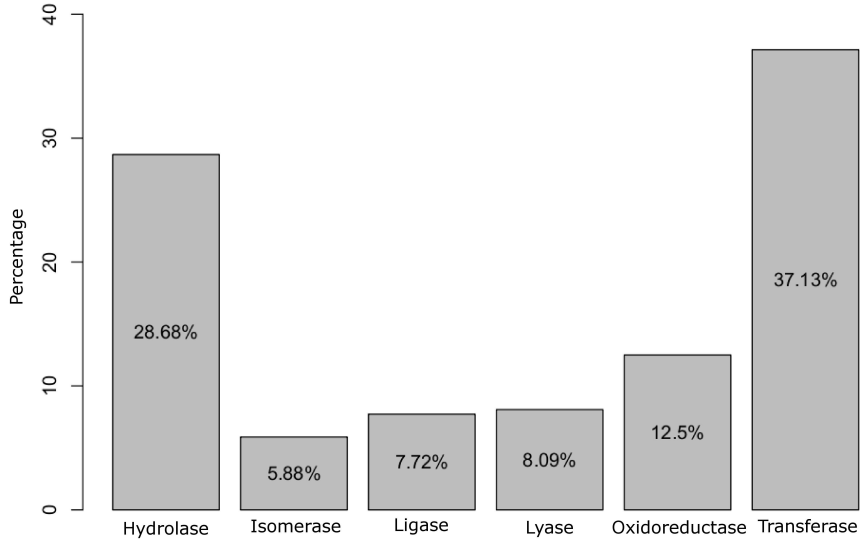


Figure 3.1: Percentages of distribution among six enzyme classes. Most of the proteins in LigASite belong to the transferase class. The second is hydrolase. The contribution of the other classes are almost the same.

3.2.2 Protein Properties Used for Training and Testing

The structure of proteins with bound ligands are obtained from the Protein Data Bank (PDB) [6], which is a collection of atomic coordinates and other information describing proteins and other important biological macromolecules. Structural biologists use methods such as X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy to determine the location of each atom relative to each other in the molecule.

After the structure of each protein is retrieved, a 3D grid is generated by covering the free-space surrounding the proteins. The program is based on the source of ConCavity, which is available on its website. The attributes of each grid point used in the SVM are

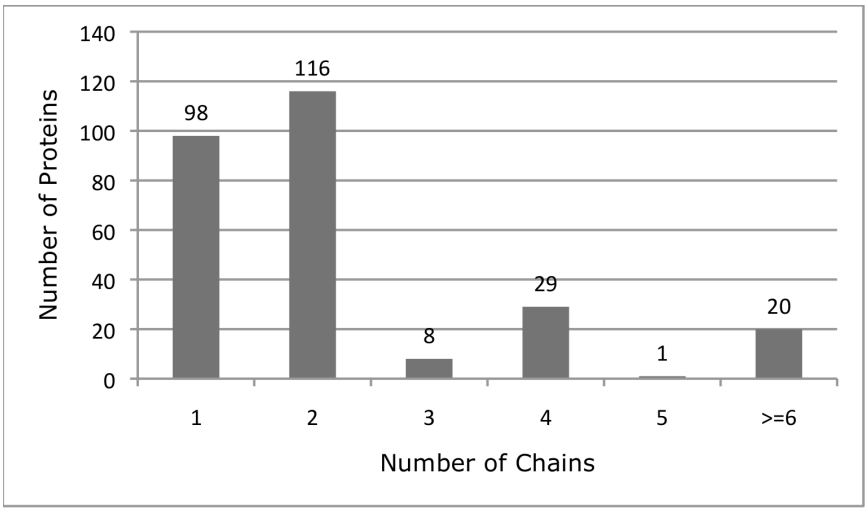


Figure 3.2: Distribution of the number of chains in the selected proteins of LigASite. Most of the proteins have less than three chains.

calculated based on the protein properties in the following:

3.2.2.1 Grid values

These are the two values of each grid point that are calculated by LIGSITE and SURFNET. They represent the binding site preference based on geometric characteristics.

3.2.2.2 Interaction potential

This is the van der Waals interaction potential of an atomic probe with the protein [14]. The calculation is done by the PocketFinder method. The Lennard-Jones formula (3.1) is used to estimate the interaction potential between the protein and a carbon atom placed at the grid point:

$$V(p) = \sum_{i=1}^{N} (\frac{C_{12}^i}{r_{pi}^{12}} - \frac{C_6^i}{r_{pi}^6}) \tag{3.1}$$

where C_{12}^i and C_6^i are constants, which are the typical 12-6 Lennard-Jones parameters used to model the van der Waals interaction energy between the carbon atom placed at the grid point p and the protein atom i; N is the total number of protein atoms. r_{pi}^{12} and r_{pi}^6 are the powers 12 and 6 of r_{pi} respectively, where r_{pi} is the distance between the grid point p and the protein atom i. The first term describes the repulsion between atoms when they are very close to each other. The second term describes the attraction between atoms at long distance.

3.2.2.3 Conservation score

Conservation score is obtained from a residue-level analysis to identify which residues in a protein are responsible for its function. The score of each grid point is the conservation score of the nearest residue. The Jensen-Shannon divergence (JSD) method is used to calculate the score since it has been shown to provide an outstanding performance in identifying residues near bound ligands. It is an open source program which is freely available on its webpage [52].

3.2.2.4 Distance from protein

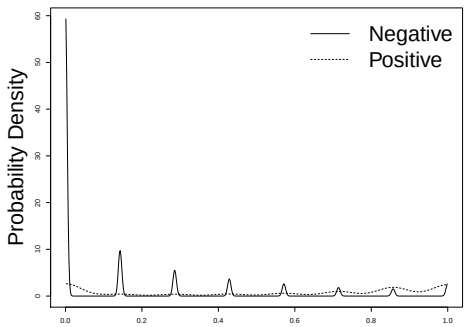
The squared distance from each grid point to the closest point on the van der Waals surface of the protein is calculated. When the grid points are too far from the atoms, they are not likely to be a pocket. In the experiment, almost 90% of ligand atoms are located within 5Å of the protein's van der Waals surface. Hence, the grid points with

the squared distance larger that 5Å are filtered out in order to reduce the huge data size.

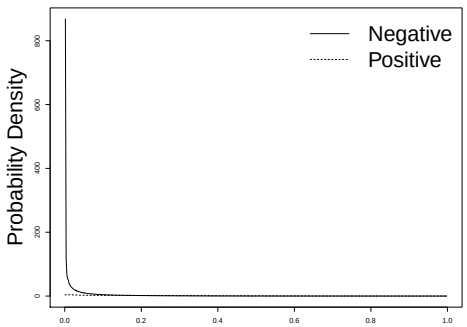
To explain the relationship between the binding sites and the selected attributes, graphs of probability density for the normalized attribute values are shown in Fig. 3.3. The solid line represents the corresponding probability density for non-binding sites (negative class) and the dotted line represents the corresponding probability density for binding sites (positive class). Some of the attributes, such as LIGSITE values, SURFNET values, and interaction potential, show a very high density of small values when the grid points are located at non-binding sites. On the other hand, these attributes show a small difference on the density when the grid points are located at binding sites. This difference is shown more clearly in Fig. 3.4. Hence, we can see that the values of these attributes are relevant to the location of the binding sites. However, it may be inadequate to use only one property to classify the binding sites. Therefore, we propose to use all of them as the features of the training set for an SVM.

3.2.2.5 Properties of surrounding grid points

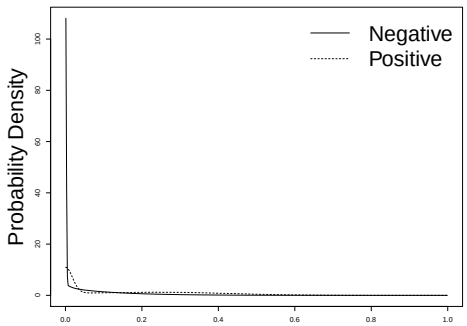
All the binding sites are formed by many grid points (the distance between two grid points is 1Å [15]), so the properties of the grid points nearby are also relevant features to the prediction. Six connected points (as shown in Fig. 3.5) are selected and their properties as described in Sections 3.2.2.1–3.2.2.3 above are used as the attributes. The point in the middle of the cube in Fig. 3.5 is the selected grid point to be classified. There are totally 29 features assigned as the SVM attributes.



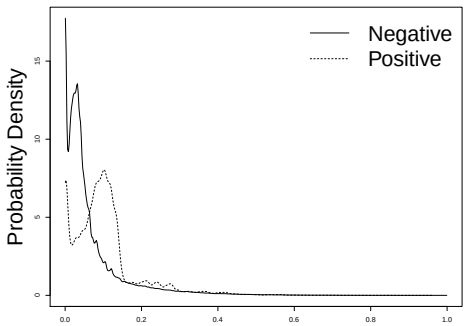
(a) Normalized grid values calculated by LIGSITE.



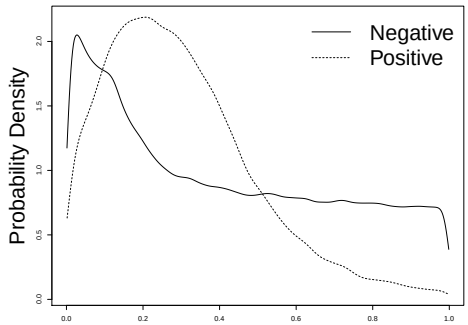
(b) Normalized grid values calculated by SURFNET.



(c) Normalized interaction potential.



(d) Normalized conservation score.

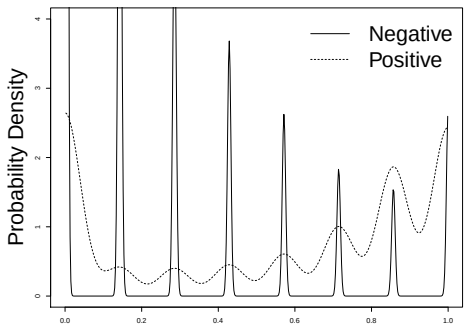


(e) Normalized distance from protein.

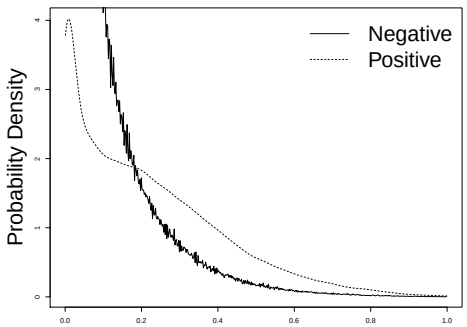
Figure 3.3: Probability density functions of protein properties.

3.2.3 Classification with Support Vector Machine

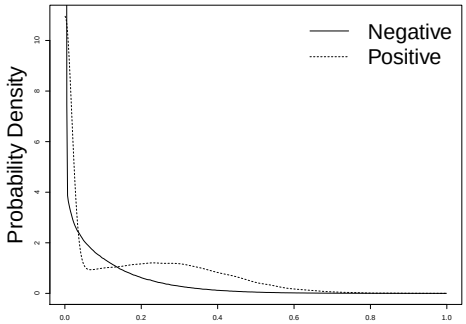
Machine learning methods have been applied to predict catalytic sites [70], [74]. In this chapter, one of the machine learning tools, the support vector machine (SVM), is



(a) Normalized grid values calculated by LIGSITE.



(b) Normalized grid values calculated by SURFNET.



(c) Normalized interaction potential.

Figure 3.4: A zoom-in version of Fig. 3.3: (a) LIGSITE values, (b) SURFNET values, and (c) Interaction potential.

employed to predict the protein-ligand binding sites. It uses the radial basis function as the kernel to construct a non-linear hyperplane. The program called SVM^{light} is used, which is available on its website [75].

SVM basically is a binary classifier. Let a vector $\mathbf{x}$ be denoted by $[x_j]$, $j = 1, \ldots, m$, where m is the number of attributes and $[x_j]$ is a point in an m-dimensional vector space. The notation $\mathbf{x}_i$ is the i-th vector in a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $y_i \in \{-1, 1\}$ is the condition label for a binary classification problem and n is the number of examples (grid points). To construct the SVM, all training samples are first mapped to a feature space by a non-linear function $\phi(\mathbf{x})$. A separating hyperplane in the feature space can

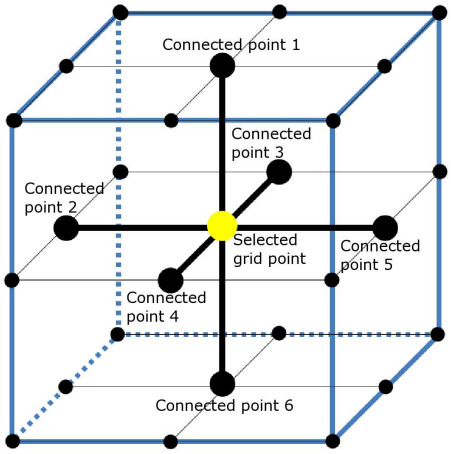


Figure 3.5: Six connected grid points of a selected grid point. All the black spots in the graph represent the grid points. The middle one is the selected grid point to be classified and the larger black spots are the connected grid points: their properties are also used as the attributes of the classification.

be expressed as

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b \\ &= \sum_{j=1}^{m} w_j \phi(x_j) + b \end{aligned} \tag{3.2}$$

where $\mathbf{w}$ is the weight vector and b is the bias.

The optimal separating hyperplane is defined as a linear classifier which can separate the two classes of training samples with the largest marginal width, and the solution $\boldsymbol{\alpha} = [\alpha_i]$ is obtained by maximizing the following function:

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{3.3}$$

subject to:

$$\sum_{i=1}^{n} y_i \alpha_i = 0, \quad i = 1, \ldots, n, \tag{3.4}$$

where $0 \leq \alpha_i \leq C * C_{factor}$ (for positive samples) and $0 \leq \alpha_i \leq C$ (for negative samples). C is the regularization parameter controlling the tradeoff between training error and margin. The larger the value of C, the larger penalty is assigned to errors. C_{factor} is a cost-factor, which makes the training errors on positive samples outweigh the errors on negative samples [76].

In the above optimization problem, only those items with $\alpha_i > 0$ can remain. The samples $\mathbf{x}_i$ that lie along or within the margins of the decision boundary (by Kuhn-Tucker theorem) are called the support vectors. The weight vector in (3.2) can be expressed in terms of $\mathbf{x}_i$ and the solutions α_i of the optimization function (3.3):

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i) \tag{3.5}$$

where $\alpha_i \geq 0$.

Then, the separating hyperplane in (3.2) becomes

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b \tag{3.6}$$

To avoid the computation of the inner product $\langle \phi(\mathbf{x}_i),\ \phi(\mathbf{x}) \rangle$ in the high dimensional space during the optimization of (3.3), the kernel function that can satisfy the Mercer's condition is introduced:

$$K(\mathbf{x}_i,\ \mathbf{x}) = \langle \phi(\mathbf{x}_i),\ \phi(\mathbf{x}) \rangle \tag{3.7}$$

The kernel function can be computed efficiently and solve the problem of mapping the samples to the potentially high dimensional feature space.

Table 3.1: Training Dataset.

1pkj	3gd9	1lf3	3lem	1llo
1ybu	4tpi	3h72	2j4e	1rn8
2v8l	1x2b	1g97	2zhz	3a0t
1o26	1rzu	1znz	1ojz	1sqf
2gga	3gh6	3d1g	2jgv	1dy3
1jyl	2e1t	2ywm	1kwc	2g28
3d4p	2wyw	2dtt	1tjw	2za1
2art	1u7z	3gid	1i1h	2w1a

Radial basis function is used as the kernel in this chapter, which is defined by

$$K(\mathbf{x}_i,\, \mathbf{x}) = exp(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}\|^2) \tag{3.8}$$

where $\sigma > 0$ is the parameter to determine the width of the radial basis function. It controls the flexibility of the classifier. When σ decreases, the flexibility of the resulting classifier in fitting the training data increases, and this might lead to over-fitting.

Around 15% of the proteins in LigASite (40 proteins) are selected as the training set of SVM since we find that the results are only slightly different when more proteins are used as training data. As the number of grid points of each protein is very large, more proteins will cause the training time to increase greatly. Based on the consideration of interpretability, only 15% of the proteins are selected to form the training set. (The rest 85% are for testing.) The training data are selected randomly with the same distribution of enzyme type as that of the whole dataset (as shown in Fig. 3.1). The proteins used in the training set are shown in Table 3.1.

Like most of the datasets in bioinformatics, the dataset used in this chapter encounters the problem of being imbalanced, i.e. the number of positive samples (the grid points

of binding site) is much smaller than the number of negative samples (the other grid points). Random under-sampling is applied to reduce this problem. Normalization is also applied to even the contribution of each attribute. After several experiments, it was found that one proper proportion between the negative samples and the positive samples is 2:1. Therefore, the negative samples are selected randomly to get to this ratio in the training set.

As a summary, the flowchart for the prediction of protein-ligand binding site is shown in Fig. 3.6. The training dataset is built with the 29 attributes of each grid point by using the ConCavity program, and the 3D grid space is set as 1 Å. The training set undergoes random under-sampling, so that the ratio 2:1 for the negative to positive samples can be obtained. SVM is applied on the re-sampled training set to form the classification model. This model is used later to classify the grid points of the testing proteins. The prediction datasets for testing are also built with the 29 attributes by using the ConCavity program. Both the learning and classifying processes of SVM use the SVM^{light} program.

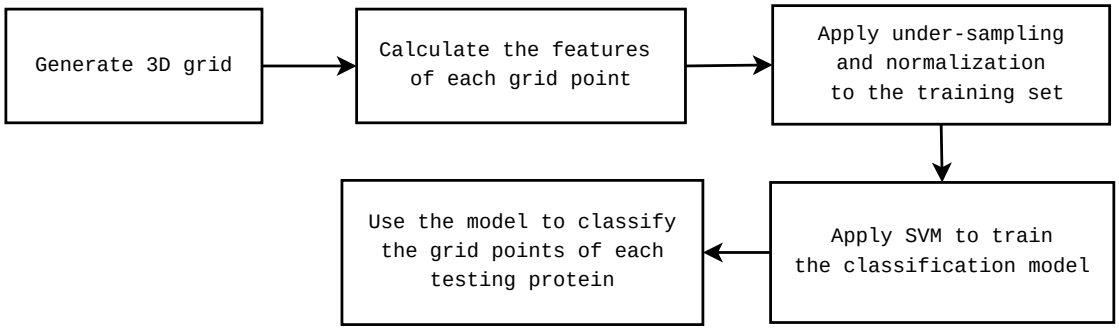


Figure 3.6: Flowchart for the prediction of protein-ligand binding site.

3.3 Evaluation

To evaluate and compare our method to the other methods, the same performance measurement should be used. We apply two different measurements on different methods and datasets.

3.3.1 Dataset of LigAsite

For this dataset, grid points are used to represent the potential binding sites. If a grid point is clustered as not suitable for binding ligands, a zero value will be assigned to it. Therefore, the prediction of ligand binding sites can be represented by non-zero values of the grid point, which represent the potential of being binding sites. The prediction can be validated by computing the difference with the grid points of known ligands. We define the grid points of the ligand atoms calculated from PDB as the positive samples and the other grid points as the negative samples.

The terms of precision and recall are introduced [59] to help showing the evaluation metric for imbalanced problems. The definitions of precision and recall are given as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.9}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.10}$$

where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives. The high value of precision indicates that the predicted

positive samples are most likely relevant. The high value of recall indicates that most of the positive samples can be predicted correctly.

Another term called $F - measure$ [59], which is a function of precision and recall, is introduced. It is a popular evaluation metric for imbalanced problems. In principle, $F - measure$ represents a harmonic mean between precision and recall. A high value of $F - measure$ means both the precision and recall values are high and do not differ very much. It is defined as follows:

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{3.11}$$

The area under the receiver operating characteristic curve (AUC) is also commonly used to measure the performance of classification. The AUC metric [66] is the probability of correctly identifying a random sample and can be defined as:

$$AUC = \frac{1 + Recall - FP_{rate}}{2} \tag{3.12}$$

where $Recall$ is defined in (3.10), $FP_{rate} = \frac{FP}{FP+TN}$ and TN is the number of true negatives. FP_{rate} defines the percentage of true negatives cases misclassified as positives. A high value of AUC implies small values of FN and FP, meaning that the corresponding classifier is very effective.

3.3.2 198 Drug-target Complexes for Testing

For this dataset, the center points of the three largest pockets are used to represent the potential binding sites. [38] proved that most of ligands bind to large pockets. Therefore, they suggested an evaluation method for comparing the top three largest sites only. In our experiment, after the grid points of potential binding sites are predicted by SVM, the top three largest sites [38] are selected and each site is represented by a grid point in the center of it.

If the center grid points are located at the real pocket sites (i.e. the distance between the center grid points and any atoms of the ligand is within 4Å), the prediction will count as a hit, which means the predicted binding site is identified correctly. There are sometimes more than one binding site within a protein and the prediction may identify more than one binding site correctly at the same time. In this case, only one hit in the larger cluster is counted. The success rate is calculated by the following equation to compare the performance of different methods:

$$success\_rate = \frac{N_{HIT}}{N_P} \tag{3.13}$$

where N_{HIT} is the number of proteins that at least one binding sites can be located correctly and N_P is the total number of proteins in the dataset.

Table 3.2: Performance under different parameters of the SVM classifier.

C	C_{factor}	σ	F-measure	AUC	Number of support vectors
0.3817	0.5	0.5	0.3089	0.6891	53,651
0.3817	0.5	1	0.3053	0.6883	54,951
0.3817	0.5	2	0.2991	0.6866	58,939
0.3817	1	0.5	0.2756	0.7289	52,327
0.3817	1	1	0.2733	0.7295	53,279
0.3817	1	2	0.2687	0.7316	56,854
0.3817	2	0.5	0.2358	0.7420	54,662
0.3817	2	1	0.2359	0.7416	55,655
0.3817	2	2	0.2345	0.7409	59,362
0.7635	0.5	0.5	0.3118	0.6928	52,106
0.7635	0.5	1	0.3081	0.6934	53,322
0.7635	0.5	2	0.3009	0.6933	56,987
0.7635	1	0.5	0.2784	0.7275	50,899
0.7635	**1**	**1**	0.2767	0.7269	51,936
0.7635	1	2	0.2731	0.7277	55,510
0.7635	2	0.5	0.2397	0.7410	53,599
0.7635	2	1	0.2409	0.7397	54,674
0.7635	2	2	0.2413	0.7377	58,366
1.1452	0.5	0.5	0.3125	0.6943	51,386
1.1452	0.5	1	0.3075	0.6940	52,574
1.1452	0.5	2	0.3014	0.6955	56,243
1.1452	1	0.5	0.2797	0.7260	50,260
1.1452	1	1	0.2781	0.7251	51,367
1.1452	1	2	0.2749	0.7247	54,960
1.1452	2	0.5	0.2416	0.7398	53,092
1.1452	2	1	0.2430	0.7370	54,149
1.1452	2	2	0.2443	0.7343	57,834

3.4 Results

In this chapter, the value of σ in (3.8) is set to the usually chosen value of 1, the value of C_{factor} for α_i in (3.4) is set to 1, and the value of C in (3.4) is equal to $\frac{1}{avg\_\mathbf{x} \cdot avg\_\mathbf{x}} = 0.7635$, where $avg\_\mathbf{x} = \frac{\sum_{i=1}^{n} \mathbf{x}_i}{n}$, $\mathbf{x}_i$ is the i-th vector in the training dataset and n is the number of samples in the dataset. These SVM parameters are set as the default values of SVM$^{\text{light}}$ program. Table 3.2 shows the validation results for different parameters of

the SVM classifier. Six random proteins from different enzyme classes are chosen to generate the validation dataset. They are 2cwh, 1g6c, 3p0x, 1wxg, 3kco, and 1k54. In the experiment, the values of σ and C_{factor} differ from 0.5 to 2. The default value of C is 0.7635 and it differs from a half to a double of the default value. The results show that an increase of $F - measure$ may lead to the decrease of AUC and the difference brought by the parameters is not significant. Therefore, the default values of each parameter are used to get a balance between $F - measure$ and AUC. The number of support vectors after the training is about 52,000.

3.4.1 Dataset of LigASite

In the first experiment, six enzyme classes are selected to compare our method with four other methods. They are LIGSITE, SURFNET, PocketFinder and ConCavity. Both LIGSITE and SURFNET used geometric characteristics to predict the ligand binding site. PocketFinder used energy criteria and ConCavity used both geometric and sequence conservation properties to do the prediction. For the grid points determination, LIGSITE applied a threshold with the value of 5.5, SURFNET and PocketFinder determine the threshold value by considering the mean and standard deviation of the grid values. ConCavity applied a binary search to the grid points. The search was made by considering different specified properties based on the sizes and shapes of the pockets. Only the grid points, which met all the properties, were selected.

The performance is calculated in terms of $F - measure$ in (3.11) and AUC in (3.12). The $F - measure$ and AUC of the training data set are shown in Table 3.3. Both results of sampled and non-sampled training data are given. The results of sampled training

data are the classification outcome of the training set that is used to learn the classification model of SVM. As mentioned before, random under-sampling is applied before the SVM training in order to tackle the problem of imbalanced dataset. The results of non-sampled training data are the classification results of the training set provided by the trained SVM without applying any under-sampling. The other 85% of the selected proteins are then used as testing data to test the performance of our method.

Table 3.3: Performance of Training Data.

Dataset	F-measure	AUC
Sampled Training Data	0.8150	0.8585
Non-sampled Training Data	0.3360	0.8417

From previous studies, ligands are not likely to be bound in small cavities. Therefore, after the grid points are classified by the SVM model, the cavities with volume small than 100 Å^3 are ignored. Table 3.4 shows the classification results of the testing dataset and our method can classify the grid points correctly with a high value of AUC. The other methods define the pockets with low AUC because the thresholds of the grid points are not always suitable to the proteins and only one property of protein is considered. The thresholds may be wrongly set by the user. On the contrary, we do not define any threshold for our method. We use SVM to train the system and cluster the grid points which are most likely to bind with ligands. The results also show that the success rate is not sensitive to the enzyme classes the proteins belong to. Both $F-measure$ and AUC show a small difference of values (around 10%) among the six enzyme classes.

Table 3.5 shows the $F-measure$ and AUC of testing datasets with different numbers of chains in the proteins. The results can be interpreted by separating them into groups. The first group has 1 or 2 chains, which has the largest values of $F-measure$. The second group has 3 or 4 chains, where the values of $F-measure$ are 0.2803 and

Table 3.4: Performance of Testing Data in Six Enzyme Classes.

Type	Method	F-measure	AUC
Transferase	Our Method	**0.3338**	**0.8162**
	LIGSITE	0.1622	0.6615
	SURFNET	0.2806	0.6516
	PocketFinder	0.08970	0.6353
	ConCavity	0.3195	0.6588
Hydrolase	Our Method	**0.3376**	**0.7548**
	LIGSITE	0.0982	0.6026
	SURFNET	0.2577	0.6332
	PocketFinder	0.07476	0.6132
	ConCavity	0.2963	0.6562
Oxidoreductase	Our Method	**0.3895**	**0.8208**
	LIGSITE	0.2044	0.6705
	SURFNET	0.3142	0.6467
	PocketFinder	0.1255	0.6396
	ConCavity	0.3314	0.6441
Lyase	Our Method	0.3025	**0.8464**
	LIGSITE	0.1507	0.7101
	SURFNET	0.2709	0.6698
	PocketFinder	0.06788	0.6349
	ConCavity	**0.3292**	0.6933
Ligase	Our Method	0.3453	**0.8407**
	LIGSITE	0.1540	0.6831
	SURFNET	0.2823	0.6612
	PocketFinder	0.07515	0.63915
	ConCavity	**0.3750**	0.6988
Isomerase	Our Method	**0.3442**	**0.7839**
	LIGSITE	0.1758	0.6685
	SURFNET	0.2497	0.6341
	PocketFinder	0.1205	0.6236
	ConCavity	0.2519	0.6177
Overall	Our Method	**0.3422**	**0.8105**
	LIGSITE	0.1576	0.7993
	SURFNET	0.2759	0.6494
	PocketFinder	0.07133	0.6310
	ConCavity	0.3172	0.6615

0.2933 respectively. The third group has 6 or more chains, which has the lowest values of $F-measure$. Generally, from the results of these three groups, F-measure decreases when the number of chains increases, except when the number of chains is 5. Fig. 3.2 shows that there is only one protein with 5 chains. Therefore, the result of the case of 5 chains is not sufficient to reflect the trend. The values of AUC is insensitive to the number of chains. The reason is that more chains in a protein means a more complicated protein structure, and the number of potential pockets on the protein's surface increases. The method predicts some extra pockets which are not true binding sites.

Table 3.5: Performance of Testing Data with Different Numbers of Chains in the Proteins.

No. of Chains	F-measure	AUC
1	0.3427	0.7950
2	0.3674	0.8057
3	0.2803	0.7976
4	0.2933	0.8105
5	0.4416	0.8989
$>= 6$	0.2575	0.7956

The grid points classified as binding sites are subject to evaluation, which is carried out by computing the difference with the known bound ligands. The $F-measure$ of all methods cannot reach a very high rate, since the size of predicted binding sites is much larger than that of ligands. Also, some predicted binding sites may be useful to look for some new ligands in the further research. Therefore, the comprehensive results are more important. After the binding sites are predicted, docking process and many medical experiments are needed to find a correct ligand to bind to the protein.

3.4.2 198 Drug-target Complexes for Testing

In the second experiment, 198 drug-target protein complexes are used and our method is compared with six other approaches, based on the evaluation of top three largest binding sites. The six other approaches are LIGSITE^{CSC}, SURFNET, Fpocket, Q-SiteFinder, ConCavity, and MetaPocket. LIGSITE and PocketFinder are not applied in this experiment since LIGSITE^{CSC} and Q-SiteFinder are the extension of them respectively. LIGSITE^{CSC}, SURFNET, and Fpocket use geometric characteristics to predict the ligand binding site. Q-SiteFinder uses energy criteria and ConCavity uses both geometric and sequence conservation properties to do the prediction. MetaPocket predicts the binding site by combining eight other approaches. Fig. 3.7 shows an example of binding sites prediction for the protein 1p5j. The real ligand is shown in red sticks at the center and the predicted pockets by all the seven approaches are shown in spheres with different colors.

The success rate of this experiment is calculated by (3.13). The prediction results of top 1 to top 3 binding sites for all approaches are evaluated separately. Table 3.6 shows the prediction results of our method and the other six approaches on the 198 drug-target dataset. Our method can achieve the highest success rate among all the methods. Table 3.7 shows the number of hit proteins among the seven methods on the drug-target dataset. There are 122 proteins that can have the binding sites correctly identified as the top 1 predictions. There are 30 and 10 proteins that can have the binding sites correctly identified as the top 2 and top 3 predictions respectively. There are 36 proteins that no associated binding sites can be identified correctly in the top 3 predictions. Our method can locate the highest number of binding sites among all methods.

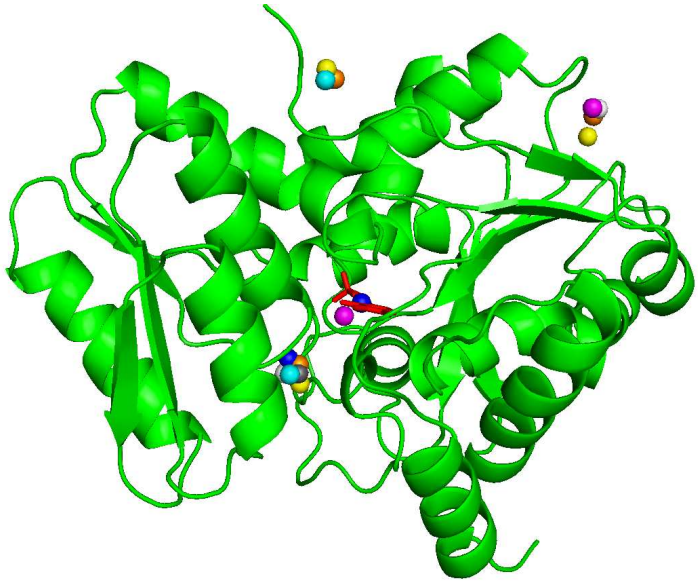


Figure 3.7: The real ligand (red) binding site and the predicted pockets for protein 1p5j. The pockets sites of MetaPocket (orange), LIGSITE^{CSC} (white), SURFNET (yellow), Fpocket (cyan), Q-SiteFinder (magenta), ConCavity (grey), and our method (blue) are shown in spheres.

The reason why our method can outperform the other methods is that no threshold is set to the grid points to identify the binding sites. Our method forms a training set with 29 different properties of some proteins first, and then applies an SVM to train a classification model. Finally, this model is used to predict the binding sites of other proteins. Besides, we have applied many different properties of protein, such as the geometric characteristics, interaction energy between the protein and a carbon probe, and sequence conservation score, to do the predictions; while some other methods use only one property to locate the binding sites.

Our method still has some limitations. In the drug-target dataset, 36 proteins cannot have the binding sites located correctly. From these cases, we conclude with three limitations of our method. The first one is that ligands may bind to a flat region. Since our method tends to predict the binding sites inside a cavity or pocket, the sites in a flat region

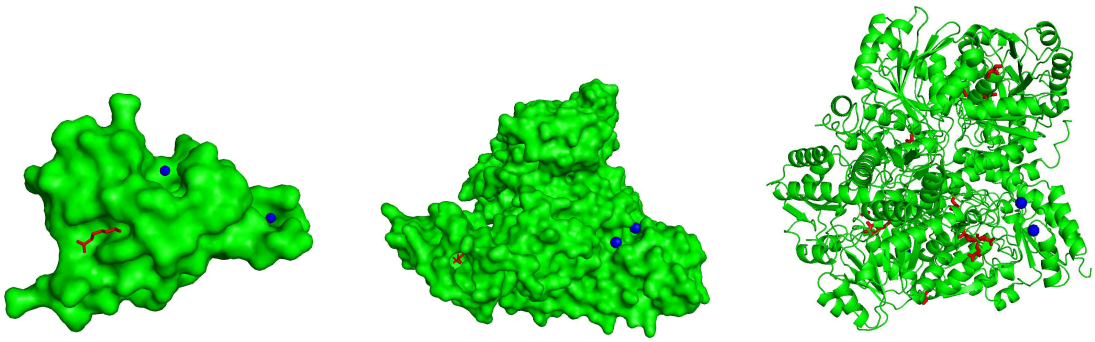
Table 3.6: Success Rate (%) of Top 3 Binding Sites Predictions on 198 Drug-Target Dataset.

Method	Top 1	Top 1-2	Top 1-3
Our Method	**61.6**	**76.8**	**81.8**
MetaPocket	61	70	74
LIGSITE^{CSC}	48	57	61
SURFNET	24	30	34
Fpocket	31	48	57
Q-SiteFinder	40	54	62
ConCavity	47	53	56

Table 3.7: Number of Hit Proteins on 198 Drug-Target Dataset.

Method	Top 1	Top 2	Top 3	None
Our Method	122	30	10	36
MetaPocket	121	17	9	51
LIGSITE^{CSC}	95	18	7	78
SURFNET	46	11	8	133
Fpocket	61	34	17	86
Q-SiteFinder	79	28	16	75
ConCavity	93	12	6	87

are difficult to locate. There are16 cases in the drug-target dataset belonging to this category. The second limitation is that ligands may bind to small cavities. Since only the top three largest binding sites are considered in the drug-target dataset, sites in small cavities cannot be selected . There are 17 cases in the drug-target dataset belonging to this category. The third limitation is that the binding sites may be inside the proteins while only the pockets on the protein surface can be detected. There are three cases in the drug-target dataset belonging to this category. Fig. 3.8 shows three examples of the difficult structures mentioned above. The real ligands are shown in red sticks. The predicted binding sites of our method are shown in blue spheres.



(a) 1pk2. (b) 1e7a. (c) 3cog.

Figure 3.8: Examples of the three limitations of our method. (a) The ligand binds to a flat region. (b) The ligands bind to small cavities. (c) The binding sites are inside the protein.

3.5 Conclusion

The determination of binding sites (pockets) is the prerequisite for protein-ligand docking and an important step of structure-based drug design. The prediction of the protein-ligand binding site has been investigated in this chapter. SVM is employed to distinguish the binding sites. It makes use of the properties of geometric characteristics, interaction potential, distance from protein, conservation score and the grid points nearby to identify the binding sites. Threshold assignment is no longer needed to determine the pockets. Distance filter and random under-sampling are also employed to reduce the effects of large data size and imbalanced data respectively.

Our approach is compared to LIGSITE, LIGSITE^{CSC}, SURFNET, Fpocket, PocketFinder, Q-SiteFinder, ConCavity, and MetaPocket on the datasets of LigASite and 198 drug-target protein complexes. For the LigASite dataset, the binding sites are represented as grid points and our approach gets better results than the other approaches. The sites are

predicted correctly with 35 % and 80 % of $F - measure$ and AUC respectively. The proposed method is shown to offer more comprehensive results than the others since more proteins fail to have the binding sites located when other approaches are used. For the 198 drug-target dataset, only the top three largest binding sites are considered and represented as one center point of each site. The results show that our approach performs better than the other approaches and predicts the binding sites correctly in 62% at top 1 prediction, 77% at top 1–2 prediction, and 82% at top 1–3 prediction. This study of binding sites identification can be further developed in the application of ligands finding by virtual screening, docking or de novo drug design.

Chapter 4

An Evolutionary Preprocessing Method Based on Over-sampling and Under-sampling for Imbalanced Datasets (SMOTE+CHC)

4.1 Introduction

Imbalanced datasets are commonly encountered in real-world classification problems. However, many machine learning algorithms are originally designed for well-balanced datasets. Re-sampling has become an important step to pre-process imbalanced dataset. It aims at balancing the datasets by increasing the sample size of the smaller class or decreasing the sample size of the larger class, which are known as over-sampling and under-sampling respectively. In this chapter, a sampling method based on both over-sampling and under-sampling is proposed, in which the new samples of the smaller class are created by the Synthetic Minority Over-sampling Technique (SMOTE) [21].

The improvement of the datasets is done by the evolutionary computational method of CHC [42] (Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation) that works on both the synthetic samples and the samples of the majority class.

In [21], it was reported that over-sampling and hybrid approaches provided better results than the under-sampling approaches. However, the re-sampled training dataset has a larger size and causes an increase in the complexity of the classification model. It also decreases the efficiency of the learning algorithm. Therefore, a hybrid method that combines SMOTE and Evolutionary Algorithm (EA) is proposed, which shows a good balance between the over-sampling rate and the accuracy. The well-known over-sampling method SMOTE is applied first to generate the new samples of the minority class. In [59], the problem of over-generalization was mentioned. This means that the synthetic samples of the minority class have occupied the area of the majority class. Therefore, hybrid methods are introduced to overcome this problem. An EA approach is used to under-sample both the synthetic samples and the samples of the majority class to improve the performance of SMOTE in this chapter. The chosen EA is the CHC algorithm since it can select the smallest and most representative instances among many algorithms as reported in [43].

Experiments were then carried out to show the performance of our proposed approach, which are compared to random under-sampling (RUS), Tomek links (TL) [61], random over-sampling (ROS), SMOTE [22], and SMOTE+Tomek links (SMOTE+TL) [21] using 22 imbalanced datasets from UCI Repository [77]. (These methods have already been introduced in Chapter 2.) C4.5 [78] is used as the learning algorithm for obtaining a classification model from each re-sampled dataset, so as to evaluate the corresponding

preprocessing method.

This chapter is organized as follows. Section 4.2 introduces the details of the novel sampling strategy and the evaluation method of this study. To show the effectiveness of our proposed method, the results and comparisons with other methods are discussed in Section 4.3. A conclusion is drawn in Section 4.4.

4.2 Methodology

In this section, the details of the proposed hybrid data preprocessing method is discussed. The data preprocessing method involves two stages: the first stage is SMOTE (over-sampling). The samples in the minority class of the training sets are firstly over-sampled with SMOTE. After applying SMOTE, the size of the minority class are the same as that of the majority class. CHC is then implemented to reduce the numbers of both the synthetic samples and the samples in the majority class. CHC has been shown to be able to reach a good balance between convergence rate and diversity of results among the EAs in [43].

There are two important issues that need to be addressed clearly before CHC is employed: the representation of each chromosome and the definition of fitness function.

4.2.1 Chromosome Representation

CHC is used to eliminate the synthetic samples as well as the majority class samples. Therefore, the chromosomes need to represent the subsets of these samples, which can be done by a binary representation. Each chromosome is an n-dimensional vector, which is a set of genes, where n is the number of genes. In this study, n is the number of synthetic samples plus the samples in the majority class. Each gene shows whether the corresponding sample exists in the training set or not. Therefore, there are two possible values for each gene: 0 and 1. If the gene value is 1, the corresponding sample is included in the subset of the training set. If the gene value is 0, the corresponding sample is excluded from the subset.

4.2.2 Fitness function

In this study, the k-Nearest Neighbor (k-NN) classifier is used as the evaluation method of CHC to obtain the subset with the highest classification rate. Normally, accuracy (ratio of correctly classified samples to total number of samples) would be used as the measure of classification rate. However, it may cause difficulty for imbalanced datasets since the corrected classification rate of the majority class affects the accuracy more seriously than that of the minority class. This problem is more obvious if the ratio of the size of majority class to that of minority class is large. The worst case could be that even all the minority class samples are misclassified, the accuracy is still very high. Therefore, the measures used in Chapter 3 are used here instead. They are $F-measure$ in (3.11) and AUC in (3.12).

Since both $F-measure$ and AUC are important measures on imbalanced datasets, a multi-objective fitness function is used here. If a chromosome X as compared with chromosome Y has a higher value of $F-measure$ ($F_X > F_Y$) and a lower value of AUC ($A_X < A_Y$), the difference between the chromosomes' $F-measure$ ($|F_X - F_Y|$) and the difference between the chromosomes' AUC ($|A_X - A_Y|$) will be compared. If $|F_X - F_Y| > |A_X - A_Y|$, chromosome X will be regarded as a better one; otherwise chromosome Y will be regarded as a better one.

4.3 Experimental Study

In this section, experiments are carried out to compare our proposed method, which is called SMOTE+CHC, with the others methods, including RUS, TL, ROS, SMOTE, and SMOTE+TL. To measure the performance of the preprocessing method, the same classification algorithm should be used among all the methods and C4.5 is selected to obtain the classification model from the re-sampled training set. The program of all testing algorithms and the learning algorithm are based on KEEL, which is an open source software tool available at the Web [79].

In this chapter, the geometric mean of $F-measure$ and AUC ($GMFA$) in (4.1) is used as the measure to analyze the results of the experiments. $GMFA$ is defined as follows:

$$GMFA = \sqrt{F-measure \cdot AUC} \tag{4.1}$$

The over-sampling rate of ROS, SMOTE, SMOTE+TL, and SMOTE+CHC will also be

compared. The over-sampling rate is defined as follows:

$$Rate_{over} = \frac{(N_{sampled} - N_{original})}{N_{original}} * 100\% \tag{4.2}$$

where $N_{sampled}$ is the number of samples in the re-sampled training set and $N_{original}$ is the number of samples in the original training set.

4.3.1 Datasets

To evaluate the algorithms' performance, 22 datasets with different imbalanced ratio (IR) from UCI Repository [77] are used. IR is the ratio of the size of majority class to that of minority class. Table 4.1 shows the details of the selected datasets, where the number of samples ($N_{samp.}$), the number of attributes ($N_{attr.}$), the percentage distribution of the minority and majority classes, and the IR for each dataset can be found.

Table 4.1: Descriptions of the Imbalanced Datasets.

Dataset	$N_{samp.}$	$N_{attr.}$	Min., Maj.(%)	IR
yeast2vs4	514	8	(9.92, 90.08)	9.08
yeast05679vs4	528	8	(9.66, 90.34)	9.35
vowel0	988	13	(9.01, 90.99)	9.98
glass016vs2	192	9	(8.85, 91.15)	10.29
glass2	214	9	(7.94, 92.06)	11.59
shuttlec0vsc4	1829	9	(6.72, 93.28)	13.87
yeast1vs7	459	7	(6.53, 93.47)	14.3
glass4	214	9	(6.07, 93.93)	15.47
ecoli4	336	7	(5.95, 94.05)	15.8
page_blocks13vs4	472	10	(5.93, 94.07)	15.86
abalone918	731	8	(5.65, 94.25)	16.4
glass016vs5	184	9	(4.89, 95.11)	19.44
shuttlec2vsc4	129	9	(4.65, 95.35)	20.5
yeast1458vs7	693	8	(4.33, 95.67)	22.1
glass5	214	9	(4.2, 95.8)	22.78
yeast2vs8	482	8	(4.15, 95.85)	23.1
yeast4	1484	8	(3.43, 96.57)	28.1

Dataset	$N_{samp.}$	$N_{attr.}$	Min., Maj.(%)	IR
yeast1289vs7	947	8	(3.16, 96.84)	30.57
yeast5	1484	8	(2.96, 97.04)	32.73
ecoli0137vs26	281	7	(2.49, 97.51)	39.14
yeast6	1484	8	(2.36, 97.64)	41.4
abalone19	4174	8	(0.77, 99.23)	129.44

4.3.2 Setup of Experiments

In this chapter, C4.5 with pruning is used to evaluate the influence of each preprocessing method. For CHC, which is applied to improve the performance of SMOTE, the basic settings of the parameters are as follows:

- Population size: 50.
- Divergence rate: 0.35.
- k of k-NN classifier used for evaluation: 1.
- Number of Evaluations: 10,000.
- Maximum number of best fitness: 1,000.

Two criteria are set to end the process of CHC. The first one is the number of evaluations. The second one is the maximum number of best fitness. If the best fitness value does not increase for 1,000 evaluations, the convergence to the global optimum is assumed. The process of CHC will then be ended. We adopt a 5-fold cross validation model to conduct the experiments, i.e., dividing the whole dataset into five parts randomly, and combining four of them for training and the remaining part for testing. All the algorithms except TL involve some random parameters, so ten experiments are carried out for each method and the average results are obtained. Table 4.2 shows the $GMFA$ for each sampling method on the 22 datasets. The results of the original datasets are also shown in the

first column and the best $GMFA$ value for each dataset is highlighted in bold. The last row shows the average $GMFA$ value of each sampling method for the datasets. The over-sampling and hybrid approaches can obtain better results than the under-sampling methods. RUS performs the worst among the seven preprocessing methods and it even performs worse than the original datasets on average. The performance of the over-sampling and hybrid methods is very similar and the difference among them is around 3%. This finding can be seen in Table 4.3 as well, which shows the average of AUC and $F-measure$ values of all the datasets. In Table 4.3, all the preprocessing methods work better than the original datasets in terms of AUC value. However, RUS gets a poor result in $F-measure$. This is because many samples of the majority class are predicted wrongly as the minority class. The over-sampling and hybrid approaches get a balance between AUC and $F-measure$ value. The results can re-confirm that data preprocessing is an important step to deal with imbalanced datasets.

Although the results of both over-sampling and hybrid approaches are better than that of under-sampling approaches, the number of training samples are increased greatly. If the value of IR is large, the over-sampling rate in (4.2) will become almost 100%. A large training set increases the complexity of the classification model. SMOTE+CHC can use less training samples and the performance does not lead to a great drop. Table 4.4 shows the over-sampling rate of different approaches. For the over-sampling methods, new minority samples are produced to balance the datasets. Therefore, the over-sampling rate of SMOTE and ROS are the same and it depends on the IR value. When the IR value is increased, the over-sampling rate is increased as well. SMOTE+CHC can obtain the lowest over-sampling rate for all the datasets and it does not depend on the IR value. CHC only selects the samples to increase the performance of the datasets, but not considering the locations of the samples.

Table 4.2: GMFA of Testing Datasets under Different Sampling Approaches.

Dataset	Original	RUS	TL	ROS	SMOTE	SMOTE+TL	SMOTE+CHC
yeast2vs4	0.7701	0.7881	**0.7886**	0.7661	0.7750	0.7706	0.7832
yeast05679vs4	0.5425	0.5577	0.5523	0.5503	0.5989	0.5902	**0.6091**
vowel0	**0.9628**	0.8421	**0.9628**	0.9483	0.9600	0.9581	0.9399
glass016vs2	0.4051	0.3789	0.3761	0.4477	0.4568	0.4561	**0.4574**
glass2	**0.5897**	0.4193	0.5208	0.5179	0.5450	0.5327	0.5821
shuttlec0vsc4	0.9978	**1.0000**	0.9978	0.9979	0.9977	0.9953	0.9979
yeast1vs7	**0.4832**	0.2959	0.3743	0.4725	0.4250	0.4258	0.4449
glass4	0.6510	0.6470	0.7749	**0.8319**	0.7282	0.6898	0.6829
ecoli4	0.7884	0.5771	0.7884	**0.7977**	0.7541	0.7523	0.7783
page_blocks13vs4	0.9815	0.7949	0.9815	**0.9841**	0.8956	0.8857	0.9359
abalone918	0.4065	0.4184	0.4554	0.6011	**0.6292**	0.6208	0.5004
glass016vs5	**0.8288**	0.6980	0.7415	0.7270	0.7385	0.7320	0.7258
shuttlec2vsc4	0.9129	0.9581	0.9129	**1.0000**	0.9904	0.9962	0.9700
yeast1458vs7	0.0000	0.2304	0.0000	0.2464	0.2646	0.2683	**0.2760**
glass5	**0.8544**	0.6632	0.8439	0.6930	0.7503	0.7081	0.6509
yeast2vs8	0.2236	0.4374	0.0000	**0.7072**	0.6621	0.5996	0.5922
yeast4	0.4312	0.4153	**0.5049**	0.4853	0.5041	0.4912	0.4793
yeast1289vs7	**0.4701**	0.2959	0.2724	0.3735	0.3109	0.3060	0.3486
yeast5	**0.8164**	0.6231	0.7978	0.7815	0.7975	0.7879	0.7845
ecoli0137vs26	0.6234	0.3837	**0.8111**	0.5613	0.4712	0.5457	0.5458
yeast6	0.6067	0.4328	**0.6959**	0.5801	0.5600	0.5499	0.5796
abalone19	0.0000	0.1236	0.0000	0.1535	**0.1594**	0.1585	0.1254
Mean	0.6066	0.5446	0.5979	**0.6466**	0.6352	0.6282	0.6268

Table 4.3: Mean of AUC and F-measure.

	Original	**RUS**	**TL**	**ROS**	**SMOTE**	**SMOTE+TL**	**SMOTE+CHC**
Mean of AUC	0.7440	0.7990	0.7555	0.7760	0.8137	**0.8157**	0.8091
Mean of F-measure	0.5386	0.3991	0.5306	**0.5590**	0.5167	0.5051	0.5070

Table 4.4: Over-Sampling Rate(%) of Different Over-Sampling and Hybrid Approaches.

Dataset	ROS	SMOTE	SMOTE+TL	SMOTE+CHC
yeast2vs4	80.16	80.16	76.85	26.31
yeast05679vs4	80.68	80.68	74.48	26.70
vowel0	81.78	81.78	81.78	33.93
glass016vs2	82.29	82.29	73.18	21.35
glass2	84.11	84.11	75.93	17.99
shuttlec0vsc4	86.55	86.55	86.50	36.13
yeast1vs7	86.93	86.93	81.64	31.64
glass4	87.85	87.85	83.64	26.64
ecoli4	88.10	88.10	86.46	31.55
page_blocks13vs4	88.14	88.14	84.85	33.05
abalone918	88.58	88.58	83.28	34.54
glass016vs5	90.22	90.22	88.45	22.69
shuttlec2vsc4	90.70	90.70	89.92	23.06
yeast1458vs7	91.34	91.34	86.33	37.27
glass5	91.59	91.59	89.37	26.52
yeast2vs8	91.70	91.70	89.83	35.22
yeast4	93.13	93.13	90.09	41.00
yeast1289vs7	93.66	93.66	90.05	41.00
yeast5	94.07	94.07	92.62	42.03
ecoli0137vs26	95.02	95.02	93.24	36.48
yeast6	95.28	95.28	93.36	43.14
abalone19	98.47	98.47	97.32	47.86

4.4 Conclusion

A hybrid re-sampling method developed based on both over-sampling and under-sampling has been proposed. First, SMOTE is employed to generate new synthetic samples of the minority class. The problem of over-generalization can occur if only SMOTE is used. Therefore, CHC is employed over the synthetic samples and the samples of the majority class to reduce the problem.

The proposed sampling method (SMOTE+CHC) is compared to RUS, TL, ROS, SMOTE, and SMOTE+TL on 22 datasets. To evaluate the performance of these seven sampling

approaches, the same classifier (C4.5 algorithm) has been used in the experimental verification. The performance of all over-sampling and hybrid methods is better than that of under-sampling in practice. Additionally, the results of the four approaches (ROS, SMOTE, SMOTE+TL, and SMOTE+CHC) are very similar and their difference is only around 3%. Although the over-sampling and hybrid methods outperform the under-sampling methods, an increased size of the training sets will lead to low efficiency of the classification model. Therefore, the over-sampling rates of them are also compared. SMOTE+CHC can obtain the lowest over-sampling rates among the four approaches for all 22 testing datasets. It can be seen that the proposed algorithm has the advantages of hybrid sampling methods with a relatively small increase in the size of training sets.

Chapter 5

An Evolutionary Hybrid Preprocessing Method Based on Regular Membership Functions for Imbalanced Datasets (FRB+CHC)

5.1 Introduction

Another hybrid sampling method is proposed in this chapter, in which the synthetic samples of the minority class are generated based on fuzzy logic, which is a useful tool to treat imbalanced datasets [28]. However, a large re-sampled training dataset will increase the complexity of the classification model and decrease the efficiency of the classification. It may cause over-generalization, which leads to a narrow decision boundary between two classes. Therefore, an evolutionary algorithm (EA) is applied on the synthetic samples and the samples of the majority class to under-sample the datasets and as a cleaning method to solve the over-generalization problem. The chosen EA is the CHC

algorithm [42] (Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation) since it shows the ability of selecting the smallest group of most representative instances among many algorithms studied in [43].

Experiments are then carried out to show the performance of our proposed method, which is compared to SMOTE and different hybrid re-sampling methods, including SMOTE+Tomek Links [21], SMOTE+ENN [21], Borderline-SMOTE [68], Safe-Level-SMOTE [69], and SMOTE+Rough Set [67]. 44 imbalanced datasets from UCI Repository [77] are used as the datasets. The Support Vector Machine (SVM) [80] is used as the tool for reaching a classification model from each re-sampled dataset, so as to evaluate the corresponding preprocessing method. The evaluation measures are based on the functions of precision and recall. Since the size of dataset is related to the complexity of classification model, the over-sampling rate and number of support vectors used are also compared. Data complexity measure, which was suggested in [81], is also used to compare the performance of different preprocessing methods.

This chapter is organized as follows: Section 5.2 introduces the details of the proposed re-sampling method and the evaluation measures of this study. To show the effectiveness of our proposed method, the comparisons and results are discussed in Section 5.3. A conclusion is drawn in Section 5.4.

5.2 Methodology

In this section, the details of the proposed hybrid preprocessing method are discussed. The proposed method involves two stages. The minority samples of the training sets

are firstly over-sampled based on fuzzy logic. To decrease the size of training sets and prevent over-generalization, CHC is then implemented to reduce the size of synthetic samples and samples of the majority class.

5.2.1 Fuzzy Logic

In this chapter, let the $positive$ class be the minority class and only λ training samples of positive class are considered. A training sample of positive class is denoted by X_α, where $X_\alpha = (x_{\alpha 1}, \ldots, x_{\alpha\gamma})$ is a γ-dimensional vector, $\alpha = 1, 2, \ldots, \lambda$ and $x_{\alpha\beta}$ is the βth attribute value $(\beta = 1, 2, \ldots, \gamma)$ of the αth training sample. The θth fuzzy if-then rule can be written as follows:

$$\begin{aligned} &\text{Rule } \theta : \text{IF } z_1 \text{ is } A_1^\theta \text{ AND } \ldots \text{ AND } z_\gamma \text{ is } A_\gamma^\theta \\ &\qquad\qquad \text{THEN class = positive with } w_\theta \end{aligned} \tag{5.1}$$

where A_β^θ is a fuzzy term of the θth rule corresponding to the attribute z_β, $\beta = (1, 2, \ldots, \gamma)$ and $z = (z_1, z_2, \ldots, z_\gamma)$ is a γ-dimensional attribute vector, and w_θ is the θth rule weight. The regular triangular membership functions are used for the fuzzy terms.

The fuzzy rules are generated based on the samples of positive class. For each sample, the label of each attribute with the highest membership value is selected to form the corresponding rule. Therefore, the number of rules must not greater than λ. The maximum number of rules depends on the numbers of labels and attributes and is equal to L^γ, where L is the number of labels.

The rule weight w_θ is used to reflect the degree of matching of each fuzzy rule over all the positive samples, so that the importance of each rule can be evaluated. First, the fuzzy value of each sample is calculated. The fuzzy value of X_α for the θth fuzzy rule is defined as follows:

$$\mu_{A^\theta}(X_\alpha) = T(\mu_{A_1^\theta}(x_{\alpha 1}), \ldots, \mu_{A_\gamma^\theta}(x_{\alpha\gamma})), \tag{5.2}$$

where $\mu_{A_\beta^\theta}(x_{\alpha\beta})$ is the fuzzy value of the βth attribute for the θth fuzzy rule and the product T-norm is used. The rule weight (w_θ) is calculated by adding the fuzzy values of all samples. In this way, w_θ reflects the effect of a rule towards the whole positive samples dataset.

$$w_\theta = \sum_{\alpha=1}^{\lambda} (\mu_{A^\theta}(X_\alpha)). \tag{5.3}$$

After the rule base of the positive class is generated, the rules are randomly drawn based on the rule weight. The rule with a higher rule weight will have a higher probability to be chosen. Then, a new sample is generated within the area of the selected rule. These processes are repeated until the number of positive samples is the same as the negative samples.

To illustrate the idea more clearly, Fig. 5.1 shows the original distribution of two classes with two attributes as an example of the formulation of fuzzy rules. The x-axis and y-axis govern the values of the two different attributes and regular triangular membership functions with five labels are used. The circle dots represent the negative samples and the square dots represent the positive samples. The dashed lines show the minimum or maximum value of the corresponding attribute of the positive samples. As only the attribute vectors of the positive class are considered to generate fuzzy rules, totally ten

rules can be formed in this example:

Rule 1: IF $Attr.1$ is L_1 AND $Attr.2$ is L_4. THEN class = positive with 0.897

Rule 2: IF $Attr.1$ is L_2 AND $Attr.2$ is L_3. THEN class = positive with 1.147

Rule 3: IF $Attr.1$ is L_2 AND $Attr.2$ is L_4. THEN class = positive with 1.508

Rule 4: IF $Attr.1$ is L_3 AND $Attr.2$ is L_3. THEN class = positive with 1.230

Rule 5: IF $Attr.1$ is L_3 AND $Attr.2$ is L_4. THEN class = positive with 2.344

Rule 6: IF $Attr.1$ is L_3 AND $Attr.2$ is L_5. THEN class = positive with 1.607

Rule 7: IF $Attr.1$ is L_4 AND $Attr.2$ is L_1. THEN class = positive with 0.727

Rule 8: IF $Attr.1$ is L_4 AND $Attr.2$ is L_4. THEN class = positive with 1.319

Rule 9: IF $Attr.1$ is L_4 AND $Attr.2$ is L_5. THEN class = positive with 1.731

Rule 10: IF $Attr.1$ is L_5 AND $Attr.2$ is L_4. THEN class = positive with 1.399

where $Attr.1$ and $Attr.2$ represent Attribute 1 and Attribute 2 for the x-axis and y-axis in Fig. 5.1 respectively. Rule 5 has the highest rule weight and rule 7 has the lowest rule weight in this example.

For generating the synthetic samples, a rule out of these ten rules is selected with the probability of selection depending on the rule weight. Then, this rule sets up the criteria with the highest and lowest value of each attribute. The new sample is generated randomly within these criteria. This process is repeated until the number of the positive samples is the same as that of the negative samples. Fig. 5.2 shows the distribution of samples after over-sampling. The triangle dots represent the synthetic samples. It is found that the spread of the synthetic samples is similar to that of the original positive samples (shown as the square dots). The synthetic samples in Fig. 5.2 are dense in the

area of rule 5.

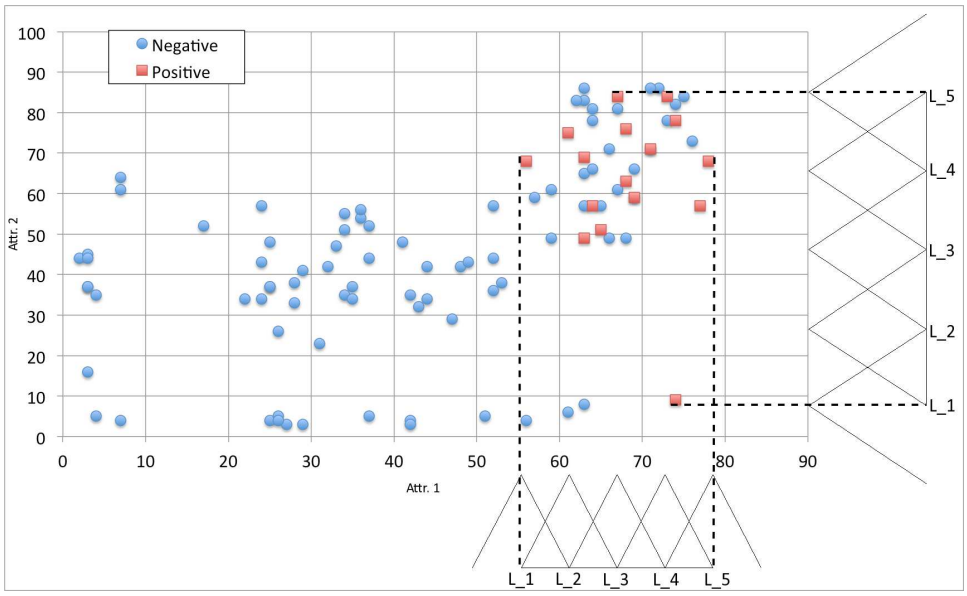


Figure 5.1: Example of the distribution of imbalanced dataset. The y-axis represents the values of $Attr.2$ and x-axis represents the value of $Attr.1$.

5.2.2 Setting of CHC

After the over-sampling, the number of minority class samples is the same as that of majority class and CHC is then applied. There are two important issues that need to be addressed clearly before the algorithm is employed: the representation of each chromosome and the definition of fitness function.

5.2.2.1 Chromosome Representation

CHC is used to reduce the synthetic samples as well as the majority class samples. The chromosomes are used to represent the subsets of the dataset. The details have been discussed when we presented the method of SMOTE+CHC, which can be found in

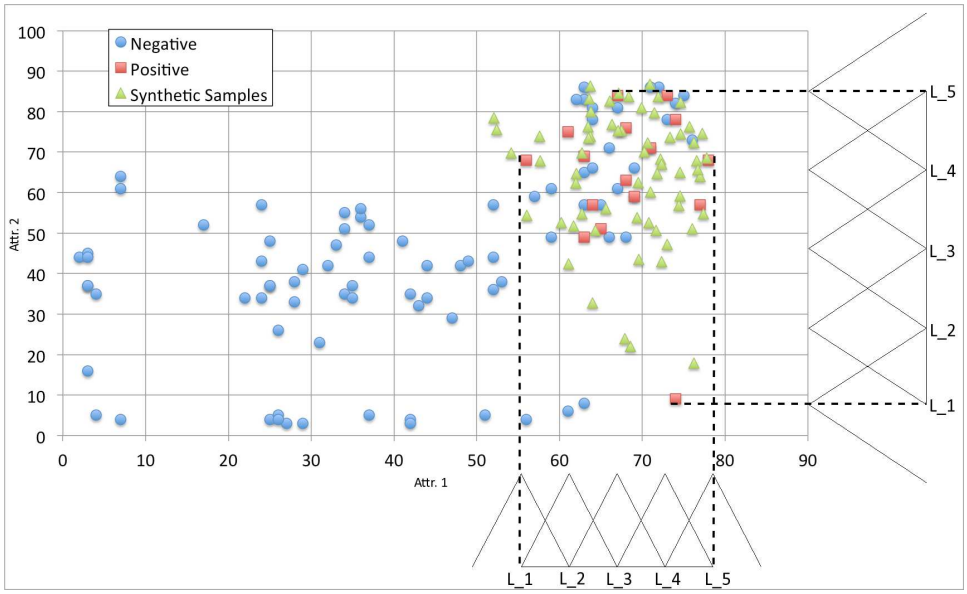


Figure 5.2: Distribution of the samples after over-sampling. The y-axis represents the values of $Attr.2$ and x-axis represents the value of $Attr.1$.

Section 4.2.1 of Chapter 4.

5.2.2.2 Fitness function

In this study, the k-Nearest Neighbor (k-NN) classifier is used again as the evaluation method of CHC to obtain the subset with the highest classification rate. The fitness function is formed from $F-measure$ and AUC, which has been introduced in Section 3.3.1 of Chapter 3.

Since both $F-measure$ and AUC are important measures on imbalanced datasets, a multi-objective fitness function is used here. If a chromosome X as compared with chromosome Y has a higher value of $F-measure$ ($F_X > F_Y$) and a lower value of AUC ($A_X < A_Y$), the difference between the chromosomes' $F-measure$ ($|F_X - F_Y|$) and the difference between the chromosomes' AUC ($|A_X - A_Y|$) will be compared. If

$|F_X - F_Y| > |A_X - A_Y|$, chromosome X will be regarded as a better one; otherwise chromosome Y will be regarded as a better one.

5.3 Experimental Study

In this section, we present the experiments that are carried out to compare our proposed method with other hybrid sampling methods. The datasets used can be found in UCI Repository [77].

The experiments involve SMOTE, sTL, sENN, sRST, sBorder, sSafe, and our proposed method, which is named as Fuzzy Rule Base+CHC (FRB+CHC). To measure the performance of the preprocessing method, the same learning tool should be used among all the experimental methods. This tool is a Support Vector Machine (SVM) that attempts to obtain the classification model from the re-sampled training set. The program of all testing methods and the learning tool are based on KEEL, which is an open source software available at the Web [79]. $F-measure$ and AUC are used as measures to analyze the results. The average value of these measures of each method will be calculated. As the large re-sampled training datasets will increase the complexity of the classification model, the over-sampling rate and the number of support vectors formed from SVM will also be compared. Finally, the behavior of each method in data complexity measures will be analyzed.

5.3.1 Setup of Experiment

For over-sampling, the rules of the minority class samples are formed from regular triangular membership functions with five labels. For CHC, the basic setting of the parameters are:

- Population size: 50.
- Divergence rate: 0.35.
- Threshold decreasing rate: 0.001.
- k of k-NN classifier used as evaluation: 1.
- Number of Evaluations: 5,000.

It was found that the above values have no significant effect to the performance of CHC. In this chapter, SVM is used to weigh the influence of each preprocessing methods. A radial basis function (RBF) is used as the SVM kernel since a non-linear classification model is needed and RBF is a common kernel to handle this problem. The RBF is defined as follows:

$$RBF = exp(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}\|^2) \tag{5.4}$$

where $\sigma > 0$ is a parameter to determine the width of the radial basis function. It controls the flexibility of the classifier. When σ decreases, the flexibility of the resulting classifier in fitting the training data increases, and this might lead to over-fitting easily. The value of σ is set as 0.01. The tradeoff between training error and margin of SVM is set as 100. The above values are chosen through experiments.

We consider a 5-fold cross validation model to develop the experiments, i.e., dividing the whole dataset into five parts randomly, and combining four of them for training and the remaining part for testing. All the methods involve some random parameters, so five experiments are carried out for each 5-fold cross validation model and the average value are calculated as the results, i.e. totally 25 experiments are done for each method.

5.3.2 Evaluation Method

To show the performance of our proposed method, $F-measure$ and AUC are used. As mentioned before, the main drawback of over-sampling or hybrid sampling methods is to increase the complexity of the learning model. Therefore, the over-sampling rates of different methods are also compared. It is defined as follows:

$$Rate_{over} = \frac{(N_{sampled} - N_{original})}{N_{original}} * 100\% \tag{5.5}$$

where $N_{sampled}$ is the number of samples in the re-sampled training set and $N_{original}$ is the number of samples in the original training set. The over-sampling rate shows the increase rate of the number of training samples. The increase rate of the support vectors used to form the classification model is also shown to evaluate the complexity of the learning model. This rate is calculated based on the support vectors generated.

$$Rate_{SV} = \frac{(SV_{sampled} - SV_{original})}{SV_{original}} \tag{5.6}$$

where $SV_{sampled}$ is the number of support vectors trained by the re-sampled training set and $SV_{original}$ is the number of support vectors trained by the original training set.

In [81], the performance of different preprocessing methods has been compared by means of data complexity measures. It has been proved that these measures are useful to evaluate the behavior of re-sampling approaches, and three kinds of data complexity measures (F1, L3, and N4) were suggested under the imbalanced framework. F1 is the maximum Fisher's discriminant ratio, which focuses on the effectiveness of the single feature dimension of different classes. It is calculated by the means and variances of each feature to investigate the overlap between different classes. First, define:

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{5.7}$$

where μ_1, μ_2, σ_1, and σ_2 are the means and variances of the two classes, respectively, for that feature. The f value of each feature is calculated and the maximum value of f is taken as F1. A small value of F1 means the overlapping area between two different classes is large. Therefore, a larger value of F1 represents a lower complexity of the dataset. L3 is the nonlinearity based on a linear classifier by linear programming. A measure of the nonlinearity of a classifier with respect to a given dataset is suggested in [82]. This measure has been modified to study the nonlinearity of the class boundary in [83]. First, a test set is created by linear interpolation between random sample pairs from the same class. Then, the error rate on this test set is measured. In this study, the error rate, L3, is defined as $(1 - AUC)$. A Support Vector Machine with a linear kernel is used as the linear classifier in this case. N4 is the nonlinearity of a 1 Nearest-Neighbor (1NN) classifier. This measure follows the same procedure of L3 to obtain the value. The only difference is that the error rate is calculated for a 1NN classifier in this case. Therefore, a larger value of L3 or N4 represents a higher complexity of the dataset. The three measures of F1, L3, and N4 are used to evaluate the behavior of different hybrid methods of re-sampling by comparing the data complexity of the training set and testing

set.

5.3.3 Datasets

To study the methods on different datasets, 44 datasets with different imbalanced ratio (IR) are chosen. IR is the ratio of the number of majority class to the number of minority class. Table 5.1 shows the details of the selected datasets, where the number of samples ($N_{samp.}$), the number of attributes ($N_{attr.}$), the percentage of minority class against majority class, IR, and the data complexity measures (F1, L3, and N4) for each dataset can be found.

Table 5.1: Descriptions of the Selected Imbalanced Datasets.

Dataset	$N_{samp.}$	$N_{attr.}$	Min., Maj.(%)	IR	F1	L3	N4
ecoli034vs5	200	7	(10, 90)	9	1.6323	0.1083	0.1300
yeast2vs4	514	8	(9.92, 90.08)	9.08	1.5793	0.4843	0.1733
ecoli067vs35	222	7	(9.91, 90.09)	9.09	0.9205	0.1760	0.1956
ecoli0234vs5	202	7	(9.9, 90.1)	9.1	1.6180	0.1094	0.0947
glass015vs2	172	9	(9.88, 90.12)	9.12	0.1375	0.5	0.4403
yeast0359vs78	506	8	(9.88, 90.12)	9.12	0.3113	0.4046	0.3758
yeast0256vs3789	1004	8	(9.86, 90.14)	9.14	0.6939	0.4506	0.3047
yeast02579vs368	1004	8	(9.86, 90.14)	9.14	1.6349	0.3510	0.1793
ecoli046vs5	203	6	(9.85, 90.15)	9.15	1.6030	0.1341	0.0988
ecoli01vs235	244	7	(9.83, 90.17)	9.17	1.1028	0.1080	0.1897
ecoli0267vs35	224	7	(9.82, 90.18)	9.18	0.9129	0.1798	0.1870
glass04vs5	92	9	(9.78, 90.22)	9.22	1.5422	0.1833	0.0400
ecoli0346vs5	205	7	(9.76, 90.24)	9.25	1.5952	0.1393	0.1221
ecoli0347vs56	257	7	(9.73, 90.27)	9.28	1.1296	0.1137	0.1261
yeast05679vs4	528	8	(9.66, 90.34)	9.35	1.0507	0.5	0.3248
vowel0	988	13	(9.01, 90.99)	9.98	2.4579	0.0855	0.1078
ecoli067vs5	220	6	(9.09, 90.91)	10	1.6922	0.1355	0.1573
glass016vs2	192	9	(8.85, 91.15)	10.29	0.2692	0.5	0.4311

Dataset	$N_{samp.}$	$N_{attr.}$	Min., Maj.(%)	IR	F1	L3	N4
ecoli0147vs2356	336	7	(8.63, 91.37)	10.59	0.5275	0.1780	0.1585
led7digit02456789vs1	443	7	(8.35, 91.65)	10.97	1.9568	0.1010	0.3684
glass06vs5	108	9	(8.33, 91.67)	11	1.0487	0.3833	0.1621
ecoli01vs5	240	6	(8.33, 91.67)	11	1.3898	0.1309	0.0670
glass0146vs2	205	9	(8.29, 91.71)	11.06	0.3487	0.5	0.4399
glass2	214	9	(7.94, 92.06)	11.59	0.3952	0.5	0.4352
ecoli0147vs56	332	6	(7.53, 92.47)	12.28	0.9124	0.1088	0.1149
cleveland0vs4	177	13	(7.34, 92.66)	12.62	1.3442	0.1835	0.4511
ecoli0146vs5	280	6	(7.14, 92.86)	13	1.3399	0.1777	0.0896
shuttlec0vsc4	1829	9	(6.72, 93.28)	13.87	12.9723	0	0.0102
yeast1vs7	459	7	(6.53, 93.47)	14.3	0.3534	0.5	0.4228
glass4	214	9	(6.07, 93.93)	15.47	1.4693	0.4325	0.0808
ecoli4	336	7	(5.95, 94.05)	15.8	3.2474	0.5	0.0969
page_blocks13vs4	472	10	(5.93, 94.07)	15.86	1.5470	0.1462	0.2245
abalone918	731	8	(5.65, 94.25)	16.4	0.6320	0.5	0.3630
glass016vs5	184	9	(4.89, 95.11)	19.44	1.8505	0.5	0.1926
shuttlec2vsc4	129	9	(4.65, 95.35)	20.5	12.1322	0	0
yeast1458vs7	693	8	(4.33, 95.67)	22.1	0.1757	0.5	0.4598
glass5	214	9	(4.2, 95.8)	22.78	1.0185	0.5	0.1980
yeast2vs8	482	8	(4.15, 95.85)	23.1	1.1424	0.2284	0.3194
yeast4	1484	8	(3.43, 96.57)	28.1	0.7411	0.5	0.2787
yeast1289vs7	947	8	(3.16, 96.84)	30.57	0.3660	0.5	0.4306
yeast5	1484	8	(2.96, 97.04)	32.73	4.1976	0.5	0.2121
ecoli0137vs26	281	7	(2.49, 97.51)	39.14	2.3018	0.1738	0.2031
yeast6	1484	8	(2.36, 97.64)	41.4	1.9675	0.5	0.2373
abalone19	4174	8	(0.77, 99.23)	129.44	0.5295	0.5	0.4909

5.3.4 Results

Tables 5.2 and 5.3 show the results on $F - measure$ and AUC for each sampling method on the 44 datasets respectively. The results of the original datasets are shown in the first column and the best value for each dataset are highlighted in bold. The last

row shows the average value of each sampling method for the datasets. FRB+CHC can obtain the highest average values of both $F - measure$ and AUC among all the methods. The performance of sTL and sENN are similar. All the preprocessing methods can perform better than the original datasets. This is expected and can re-confirm that the preprocessing is an important step to deal with imbalanced datasets.

The main drawback of over-sampling or hybrid sampling methods is that the number of training samples are increased greatly. If IR of the dataset is large, the size of the re-sampled training set is almost a double of that of the original one (an over-sampling rate of nearly 100%). This drawback will cause the increase of complexity of the learning model. Table 5.4 shows the over-sampling rate (5.5) of different methods on each dataset and the mean rate of each method. FRB+CHC can obtain the best over-sampling rate of all datasets while the rates of the other methods are similar. A negative value is shown in many datasets. It means the size of the re-sampled dataset is smaller than the original one. The difference between FRB+CHC and the other methods is significant. This shows that FRB+CHC can use less training samples to achieve high performance.

Table 5.5 shows the increased rate of the number of support vectors used to form the classification model (5.6). The number of support vectors can reflect the complexity of the classification model formed by SVM. When the number of support vectors is smaller, the classification model is more easily applied. Some negative values can be found since the number of support vectors for the re-sampled datasets is less than that of original datasets. FRB+CHC gives the best rate in the overall performance. The average number of support vectors are only increased by around 0.9 times of the original datasets; while most of the other methods have been increased by over 2 times.

Table 5.2: F-measure of Testing Datasets among Different Sampling Methods.

Dataset	Original	SMOTE	sTL	sENN	sBorder	sSafe	sRST	FRB+CHC
ecoli034vs5	0	0.5629	**0.5901**	0.5522	0.2678	0.5578	0.5007	0.5829
yeast2vs4	0.6384	0.6824	0.6683	0.6963	**0.7090**	0.6824	0.6787	0.7015
ecoli067vs35	0	0.4540	**0.5122**	0.4661	0.4977	0.4609	0.4447	0.4308
ecoli0234vs5	0	0.5176	0.5240	0.5307	0.2958	0.5012	0.4734	**0.6142**
glass015vs2	0	0.3094	0.3103	0.3143	0.2427	0.3301	**0.3419**	0.2049
yeast0359vs78	0.3481	0.3541	0.3379	0.3637	**0.3912**	0.3580	0.3529	0.3470
yeast0256vs3789	0.1782	0.5282	0.5206	0.5371	0.5363	0.5286	0.5325	**0.5899**
yeast02579vs368	**0.8152**	0.7199	0.7179	0.7304	0.7275	0.7189	0.7201	0.7747
ecoli046vs5	0	0.3901	0.3958	0.3732	0.2014	0.4084	0.4214	**0.5225**
ecoli01vs235	0	0.4325	**0.4396**	0.4201	0.2869	0.4352	0.4264	0.4224
ecoli0267vs35	0	0.3158	0.3257	0.3002	0.3091	0.2902	0.3253	**0.4592**
glass04vs5	**1**	0.8793	0.8747	0.8950	**1**	0.9228	0.9209	0.9631
ecoli0346vs5	0	0.5446	0.6397	0.6147	0.2870	0.5741	0.5642	**0.6766**
ecoli0347vs56	0	**0.5743**	0.5628	0.5547	0.5189	0.5576	0.5104	0.5176
yeast05679vs4	0	0.4327	0.4282	0.4387	0.4736	0.4333	0.4250	**0.4786**
vowel0	**1**	0.9936	0.9905	0.9899	0.9984	0.9890	0.9816	0.9060
ecoli067vs5	0	0.3260	0.3463	0.3745	0.1910	0.3444	0.3225	**0.6173**
glass016vs2	0	**0.3196**	0.2686	0.2955	0.3195	0.3048	0.2963	0.2001
ecoli0147vs2356	0	0.4230	**0.4960**	0.4445	0.3897	0.4354	0.4435	0.4043
led7digit02456789vs1	**0.7748**	0.5707	0.5226	0.4746	0.7389	0.5766	0.5156	0.6746
ecoli01vs5	0	0.4138	0.4482	0.4437	0.2873	0.4103	0.4946	**0.6843**
glass06vs5	**1**	0.9057	0.8953	0.9102	0.9899	0.8857	0.9083	0.9783
glass0146vs2	0	0.2463	0.2247	0.2560	**0.2836**	0.2473	0.2814	0.2597
glass2	0	0.2477	0.2329	0.2590	0.2486	0.2478	**0.2988**	0.2019
ecoli0147vs56	0	0.5757	0.6288	0.5805	0.4192	0.6022	0.5103	**0.6762**
cleveland0vs4	0	0.1539	0.1560	0.1497	0.1169	0.1263	0.1600	**0.1687**
ecoli0146vs5	0	0.4280	0.4112	0.4264	0.2263	0.4356	0.4422	**0.7456**
shuttlec0vsc4	0.9490	0.9740	0.9749	0.9757	0.9753	0.9740	**0.9817**	0.7964
yeast1vs7	0	0.2926	0.2865	0.2998	0.2760	0.2939	0.2738	**0.3161**
glass4	0.8560	0.6633	0.6590	0.6371	**0.8657**	0.6613	0.6463	0.7273
ecoli4	0.7500	0.6352	0.6354	0.6399	**0.8075**	0.6389	0.6491	0.7356
page_blocks13vs4	**0.2270**	0.2033	0.2010	0.2010	0.1949	0.2034	0.1894	0.1816
abalone918	0.0444	0.4522	0.4206	0.4507	0.5671	0.4474	0.4570	**0.5732**
glass016vs5	0.6650	0.5674	0.5601	0.5172	0.7211	0.5668	0.6551	**0.7694**
shuttlec2vsc4	0.4000	0.7152	0.7152	0.7152	0.7152	0.7152	**0.7288**	0.6126
yeast1458vs7	0	0.1318	0.1260	0.1345	0.1569	0.1323	0.1344	**0.1557**
glass5	0.7000	0.5937	0.5495	0.5684	0.7000	0.5932	0.4838	**0.7533**
yeast2vs8	**0.6967**	0.5972	0.5905	0.6035	0.6128	0.5989	0.5984	**0.6967**
yeast4	0	0.2703	0.2648	0.2713	0.3435	0.2715	0.2711	**0.3533**
yeast1289vs7	0	0.1395	0.1357	0.1429	0.1663	0.1397	0.1308	**0.1967**
yeast5	0	0.4843	0.4742	0.4836	**0.5111**	0.4818	0.4751	0.4476
ecoli0137vs26	0	0.3976	**0.4681**	0.4378	0.4208	0.4292	0.3636	0.3465
yeast6	0	0.2698	0.2606	0.2702	**0.3546**	0.2705	0.2670	0.3288
abalone19	0	0.0408	0.0403	0.0409	0.0457	0.0409	**0.0486**	0.0482
Mean	0.2510	0.4711	0.4734	0.4723	0.4634	0.4733	0.4693	**0.5179**

Table 5.3: AUC of Testing Datasets among Different Sampling Methods.

Dataset	Original	SMOTE	sTL	sENN	sBorder	sSafe	sRST	FRB+CHC
ecoli034vs5	0.4972	0.7069	0.7236	0.7014	0.5922	0.7047	0.6799	**0.8217**
yeast2vs4	0.7362	0.8924	0.8900	0.8929	0.8804	**0.8931**	0.8892	0.8757
ecoli067vs35	0.5	0.6860	0.7063	0.6800	0.6818	0.6790	0.6700	**0.7860**
ecoli0234vs5	0.4972	0.6978	0.7081	0.7084	0.6006	0.6943	0.6820	**0.8289**
glass015vs2	0.5	0.7152	0.7284	0.7082	0.5982	0.7376	**0.7496**	0.5530
yeast0359vs78	0.6067	0.7344	0.7281	0.7407	**0.7572**	0.7391	0.7334	0.6062
yeast0256vs3789	0.5486	0.7960	0.7972	0.8015	0.7943	0.7965	**0.7993**	0.7691
yeast02579vs368	0.8695	0.9057	0.9085	0.9100	0.9120	0.9035	0.9071	**0.9125**
ecoli046vs5	0.4973	0.6496	0.6488	0.6421	0.5626	0.6574	0.6696	**0.7880**
ecoli01vs235	0.4955	0.6606	0.6628	0.6571	0.6023	0.6598	0.6616	**0.7866**
ecoli0267vs35	0.5	0.6073	0.6093	0.6038	0.6078	0.6020	0.6113	**0.8176**
glass04vs5	**1**	0.9754	0.9728	0.9771	**1**	0.9842	0.9830	0.9732
ecoli0346vs5	0.4973	0.6974	0.7421	0.7274	0.5898	0.7124	0.7127	**0.8459**
ecoli0347vs56	0.5	0.7569	0.7594	0.7477	0.6899	0.7444	0.7294	**0.7888**
yeast05679vs4	0.5	0.7869	0.7862	0.7892	**0.7950**	0.7861	0.7797	0.7899
vowel0	**1**	0.9993	0.9990	0.9989	0.9998	0.9988	0.9981	0.9892
ecoli067vs5	0.5	0.6103	0.6155	0.6295	0.5608	0.6175	0.6106	**0.8125**
glass016vs2	0.5	**0.7529**	0.7106	0.7242	0.7084	0.7464	0.7322	0.6114
ecoli0147vs2356	0.4984	0.6509	0.6920	0.6625	0.6358	0.6580	0.6629	**0.8054**
led7digit02456789vs1	0.8788	0.8819	0.8799	0.8791	0.8684	**0.8856**	0.8650	0.8844
ecoli01vs5	0.4977	0.6602	0.6786	0.6705	0.5980	0.6566	0.6875	**0.8159**
glass06vs5	**1**	0.9774	0.9574	0.9669	0.9900	0.9629	0.9436	0.9840
glass0146vs2	0.5	0.6823	0.6594	0.6946	0.6833	0.6821	**0.7142**	0.6336
glass2	0.5	0.7132	0.6938	0.7253	0.6623	0.7127	**0.7607**	0.6078
ecoli0147vs56	0.5	0.7160	0.7460	0.7215	0.6382	0.7335	0.7053	**0.8578**
cleveland0vs4	0.4969	0.5622	0.5526	0.5508	0.5304	0.5321	0.5421	**0.5857**
ecoli0146vs5	0.4981	0.6440	0.6394	0.6444	0.5831	0.6467	0.6558	**0.8371**
shuttlec0vsc4	0.9515	0.9747	0.9755	0.9763	0.9759	0.9747	**0.9845**	0.9812
yeast1vs7	0.5	0.7583	**0.7632**	0.7625	0.6757	0.7602	0.7500	0.6932
glass4	0.9092	0.9148	0.9113	0.9128	**0.9393**	0.9143	0.9163	0.9230
ecoli4	0.8000	0.9101	0.9143	0.9171	0.9326	0.9171	**0.9426**	0.9368
page_blocks13vs4	0.5700	0.7528	0.7493	0.7495	0.7388	**0.7531**	0.7298	0.7141
abalone918	0.5125	**0.8961**	0.8863	0.8859	0.8674	0.8939	0.8916	0.8597
glass016vs5	0.8443	0.8856	0.8791	0.8810	0.8893	0.8853	**0.9221**	0.9186
shuttlec2vsc4	0.7000	0.9548	0.9548	0.9548	0.9548	0.9548	**0.9590**	0.9493
yeast1458vs7	0.5	0.6427	0.6396	0.6434	0.6302	0.6444	**0.6539**	0.5958
glass5	0.8451	0.8760	0.8807	0.8816	0.8451	0.8845	0.8515	**0.8967**
yeast2vs8	0.7739	0.7628	0.7614	0.7642	0.6964	0.7633	**0.7770**	0.7739
yeast4	0.5	0.8156	**0.8227**	0.8136	0.8203	0.8160	0.8124	0.7991
yeast1289vs7	0.5	0.7141	0.7133	**0.7166**	0.6761	0.7109	0.6968	0.6990
yeast5	0.5	0.9668	0.9655	0.9667	**0.9703**	0.9665	0.9655	0.9621
ecoli0137vs26	0.5	0.7118	0.7390	0.7425	**0.7434**	0.7413	0.6909	0.6655
yeast6	0.5	0.8742	0.8716	0.8743	0.8624	0.8744	0.8736	**0.8879**
abalone19	0.5	0.7177	0.7163	0.7183	0.6882	0.7180	**0.7715**	0.7016
Mean	0.6141	0.7784	0.7805	0.7799	0.7507	0.7795	0.7801	**0.8028**

Table 5.4: Over-sampling Rate (%) of Training Sets among Different Sampling Methods.

Dataset	SMOTE	sTL	sENN	sBorder	sSafe	sRST	FRB+CHC
ecoli0347vs56	80.53	77.63	77.02	80.53	80.53	100.77	-4.54
yeast2vs4	80.16	76.85	76.17	80.16	80.16	80.16	-4.23
ecoli067vs35	80.18	77.14	76.02	80.18	80.18	94.13	-4.95
ecoli0234vs5	80.20	77.10	77.47	80.20	80.20	89.36	-6.44
glass015vs2	80.23	70.79	70.06	80.23	80.23	80.23	0.44
yeast0359vs78	80.24	71.49	70.60	80.24	80.24	80.34	-3.26
yeast0256vs3789	80.28	74.25	73.58	80.28	80.28	80.73	0.60
yeast02579vs368	80.28	76.97	76.77	80.28	80.28	80.28	-2.57
ecoli046vs5	80.30	77.09	76.72	80.30	80.30	112.06	-2.10
ecoli0147vs2356	84.01	80.02	80.57	84.01	84.01	119.57	3.01
ecoli0267vs35	80.36	76.56	75.67	80.36	80.36	94.65	-0.22
glass04vs5	80.44	77.18	75.81	80.44	80.44	96.76	-3.56
ecoli034vs5	80.20	77.72	77.59	80.20	80.20	86.63	-2.76
ecoli0346vs5	80.39	78.18	78.55	80.39	80.39	87.98	-1.60
yeast05679vs4	80.68	74.48	72.16	80.68	80.68	80.68	-1.94
vowel0	81.78	81.78	81.65	81.78	81.78	84.56	-3.09
ecoli067vs5	81.82	75.68	73.86	81.82	81.82	87.27	-0.57
glass016vs2	82.29	73.18	71.88	82.29	82.29	82.29	-5.60
ecoli0137vs26	90.48	87.65	87.21	90.48	90.48	169.85	-3.86
led7digit02456789vs1	83.30	78.39	49.55	83.30	83.30	94.02	-5.59
ecoli0147vs56	83.97	79.91	79.46	83.97	83.97	137.03	0.81
glass06vs5	83.34	80.79	79.17	83.34	83.34	91.22	3.24
glass0146vs2	83.41	74.63	73.66	83.41	83.41	83.41	-2.56
glass2	84.11	75.93	75.70	84.11	84.11	84.11	1.16
ecoli0146vs5	89.30	86.00	86.17	89.30	89.30	139.65	-0.22
cleveland0vs4	84.97	80.35	79.19	84.97	84.97	205.49	-0.29
ecoli01vs5	92.75	90.61	90.49	92.75	92.75	186.47	-2.23
shuttlec0vsc4	86.55	86.50	86.50	86.55	86.55	136.58	-2.43
yeast1458vs7	91.81	87.16	87.11	91.81	91.81	91.81	-2.16
glass4	87.85	83.65	82.48	87.85	87.85	112.84	-3.04
ecoli4	88.10	86.46	86.61	88.10	88.10	88.39	-3.20
page_blocks13vs4	88.14	86.60	85.86	88.14	88.14	157.10	-1.59
abalone918	88.58	83.38	83.28	88.58	88.58	88.58	-1.47
glass016vs5	90.22	88.45	87.10	90.22	90.22	94.57	-3.26
shuttlec2vsc4	90.70	89.92	87.98	90.70	90.70	113.19	-9.30
yeast1289vs7	92.32	88.44	88.30	92.32	92.32	92.32	-0.41
glass5	91.59	89.37	88.44	91.59	91.59	92.76	-1.64
yeast2vs8	91.70	89.83	89.63	91.70	91.70	98.44	-1.45
yeast4	93.13	90.09	89.56	93.13	93.13	93.13	-1.23
yeast1vs7	89.62	85.04	84.63	89.62	89.62	89.62	-3.36
yeast5	94.07	92.62	92.59	94.07	94.07	94.07	-2.91
ecoli01vs235	82.59	78.95	77.60	82.59	82.59	106.04	0.98
yeast6	95.28	93.36	93.33	95.28	95.28	95.28	-2.06
abalone19	98.47	97.32	97.66	98.47	98.47	98.47	-0.76
Mean	80.70	81.94	80.71	85.70	85.70	103.47	**-2.09**

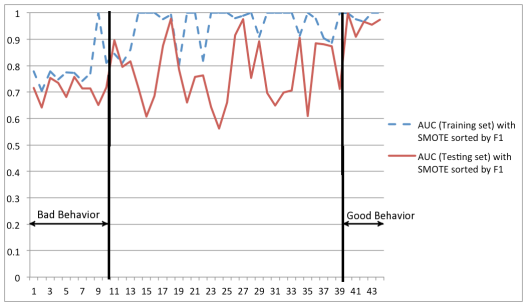
Table 5.5: The Increased Rate of Number of Support Vectors of the Classification Model formed by SVM.

Dataset	SMOTE	sTL	sENN	sBorder	sSafe	sRST	FRB+CHC
ecoli0347vs56	0.245	0.418	0.412	0.194	0.143	0.476	0.020
yeast2vs4	2.662	2.507	2.439	2.050	2.981	2.698	1.001
ecoli067vs35	0.252	0.468	0.452	0.255	0.249	0.465	0.010
ecoli0234vs5	0.272	0.472	0.462	0.144	0.122	0.473	-0.004
glass015vs2	3.731	3.193	3.456	3.111	3.996	3.583	1.331
yeast0359vs78	1.384	1.149	1.159	0.917	1.434	1.418	0.116
yeast0256vs3789	3.555	3.240	3.064	3.033	3.669	3.577	0.928
yeast02579vs368	2.142	1.919	1.771	1.799	2.251	2.188	0.505
ecoli046vs5	0.263	0.460	0.457	0.142	0.125	0.490	0.040
ecoli0147vs2356	0.229	0.435	0.431	0.188	0.170	0.494	0.080
ecoli0267vs35	0.279	0.490	0.438	0.283	0.234	0.467	0.063
glass04vs5	1.839	1.467	1.333	-0.015	4.104	1.242	0.329
ecoli034vs5	0.269	0.456	0.452	0.118	0.135	0.477	0.038
ecoli0346vs5	0.286	0.488	0.475	0.120	0.140	0.497	0.040
yeast05679vs4	3.500	3.113	3.033	2.375	3.561	3.575	1.422
vowel0	1.116	0.752	0.822	0.045	2.311	0.972	1.582
ecoli067vs5	0.227	0.391	0.398	0.211	0.212	0.438	0.058
glass016vs2	3.868	3.296	3.571	2.965	4.209	3.660	1.405
ecoli0137vs26	0.175	0.338	0.333	0.141	0.089	0.439	0.016
led7digit02456789vs1	3.332	2.864	0.043	0.344	4.139	2.838	0.408
ecoli0147vs56	0.218	0.391	0.386	0.204	0.121	0.522	-0.112
glass06vs5	1.494	1.124	1.249	0.034	2.924	1.204	0.331
glass0146vs2	3.967	3.396	3.711	3.117	4.271	3.725	1.605
glass2	3.820	3.221	3.691	2.934	4.036	3.671	1.689
ecoli0146vs5	0.189	0.361	0.351	0.170	0.105	0.394	0.053
cleveland0vs4	0.540	0.712	0.721	0.485	0.247	0.505	0.019
ecoli01vs5	0.148	0.309	0.310	0.164	0.082	0.417	-0.041
shuttlec0vsc4	0.221	0.264	0.263	0.212	0.001	0.450	1.359
yeast1458vs7	3.121	2.947	3.027	1.624	3.233	3.100	2.157
glass4	1.968	0.836	1.118	0.200	4.223	1.852	0.426
ecoli4	2.173	2.020	2.086	1.521	2.668	2.394	0.886
page_blocks13vs4	0.836	0.863	0.847	0.925	0.057	1.198	0.103
abalone918	8.586	8.058	8.325	3.973	10.025	8.077	4.159
glass016vs5	2.084	1.440	1.575	0.264	3.498	1.846	0.551
shuttlec2vsc4	0.616	1.388	1.345	0.713	0.153	1.310	0.378
yeast1289vs7	4.846	4.611	4.721	3.421	5.170	5.087	1.064
glass5	2.307	1.884	2.128	0.282	4.623	2.151	0.615
yeast2vs8	4.905	4.920	4.808	4.789	5.282	5.169	1.212
yeast4	3.133	2.902	3.002	1.580	3.379	3.173	0.959
yeast1vs7	5.196	4.870	4.982	3.869	5.523	5.382	1.711
yeast5	3.178	2.617	2.894	2.537	3.676	3.265	1.995
ecoli01vs235	0.286	0.463	0.419	0.176	0.153	0.484	-0.080
yeast6	6.825	6.376	6.448	2.412	7.216	6.900	3.083
abalone19	13.156	12.829	13.097	8.465	14.221	12.973	8.265
Mean	2.351	2.198	2.193	1.420	2.708	2.403	**0.949**

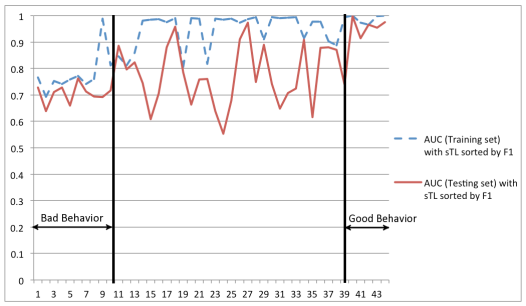
To show the behavior of different hybrid methods with data complexity measures, Figs. 5.3, 5.4, and 5.5 reveal the AUC result for each sampling method sorted by F1, L3, and N4 in ascending order. The y-axis of each figure shows the average AUC results obtained for both training and testing sets. The x-axis shows the 44 datasets sorted by F1, L3, and N4 accordingly. The solid lines in the figures represent the average AUC results for the testing set; while the dashed lines represent the average AUC results for the training set. In [81], it is suggested that different intervals of F1, L3, and N4 can present good or bad behavior under different sampling methods. In this study, good behavior means a high average AUC value for the testing set, which is experimentally set as 0.75, and the absence of over-fitting (less than a difference of 0.2 between the AUC values of training set and testing set). For bad behavior, it means the presence of over-fitting or a low average AUC value of testing set.

The performance of the SMOTE-based methods are similar to each other since they all use SMOTE as the over-sampling method. Therefore, the improvements are not significant, even they have used different under-sampling methods to eliminate the drawbacks of SMOTE.

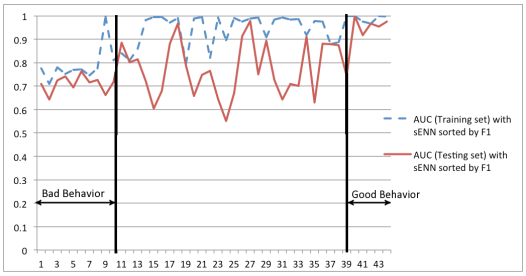
The good and bad behavior intervals of F1, L3, and N4 under different sampling methods can be extracted from the AUC graphs (Figs. 5.3, 5.4, and 5.5). Based on these intervals, Tables 5.7, 5.8, and 5.9, which sort the datasets by F1, L3, and N4 measures respectively in ascending order, are used to compare the region of good and bad behavior under different sampling methods. In Table 5.8, the region of bad behavior cannot be separated by using L3 values since there are 16 datasets with the same L3 values. Therefore, the rules of bad behavior in Table 5.6 cannot be summarized with the L3 values. Table 5.6 summarizes the intervals to form different rules to represent the good and bad



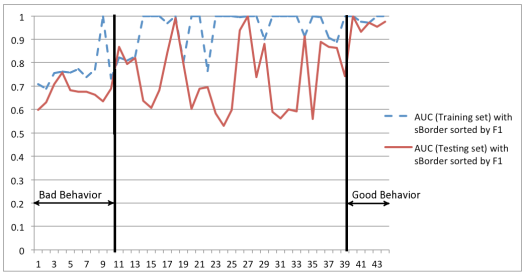
(a) AUC results with SMOTE



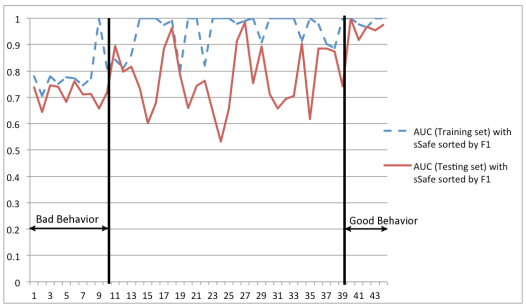
(b) AUC results with sTL



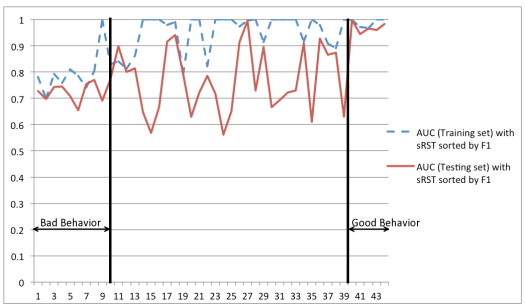
(c) AUC results with sENN



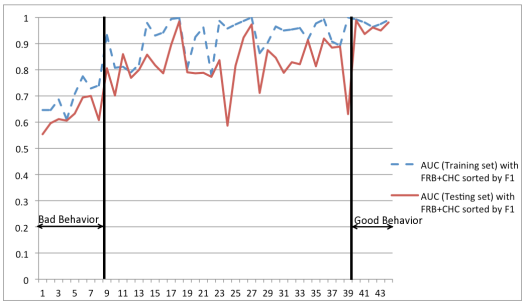
(d) AUC results with sBorder



(e) AUC results with sSafe

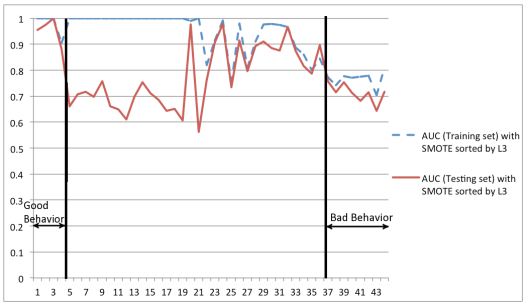


(f) AUC results with sRST

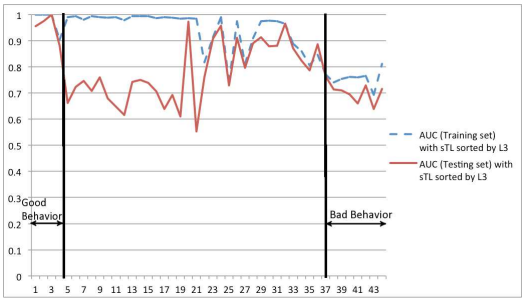


(g) AUC results with FRB+CHC

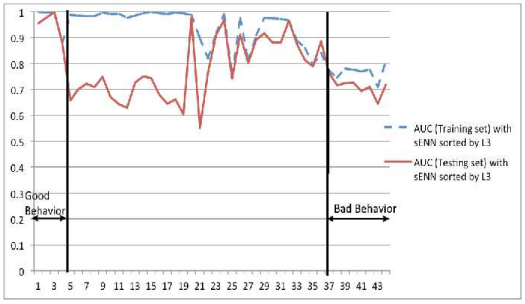
Figure 5.3: Average AUC results obtained from training and testing sets sorted by F1.



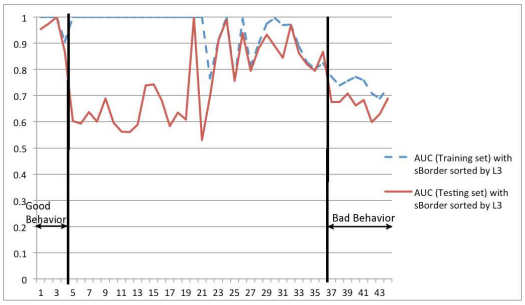
(a) AUC results with SMOTE



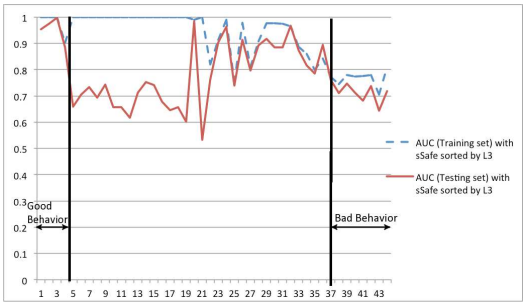
(b) AUC results with sTL



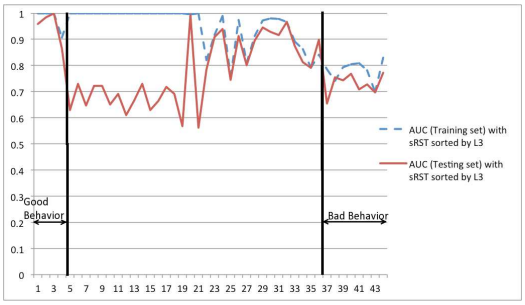
(c) AUC results with sENN



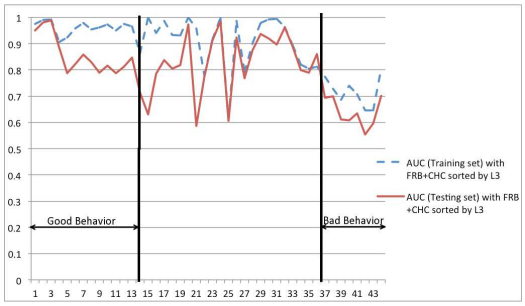
(d) AUC results with sBorder



(e) AUC results with sSafe

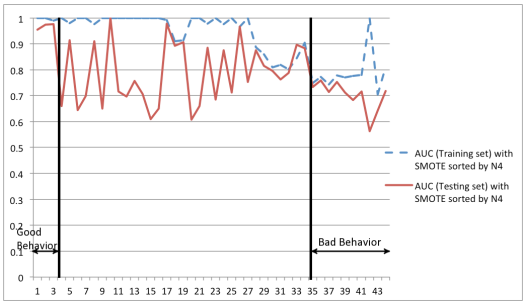


(f) AUC results with sRST

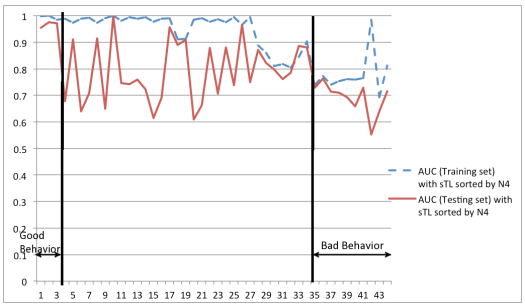


(g) AUC results with FRB+CHC

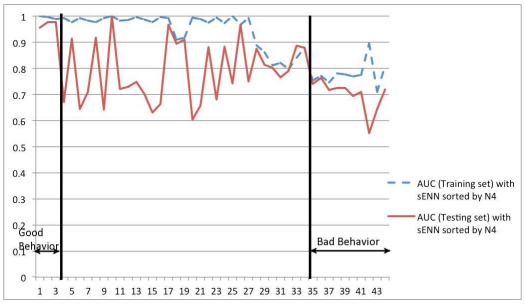
Figure 5.4: Average AUC results obtained from training and testing sets sorted by L3.



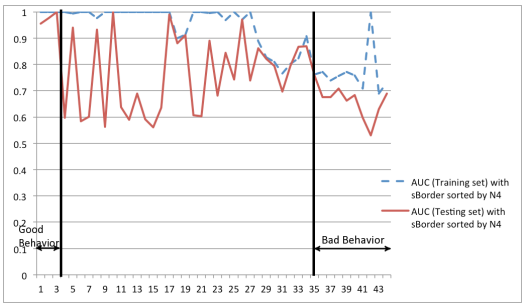
(a) AUC results with SMOTE



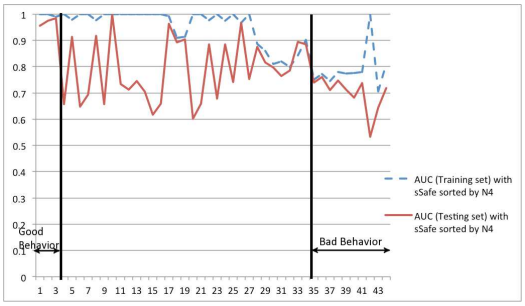
(b) AUC results with sTL



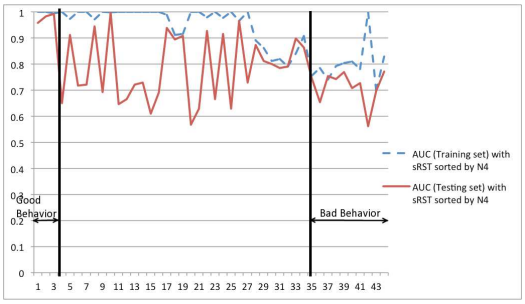
(c) AUC results with sENN



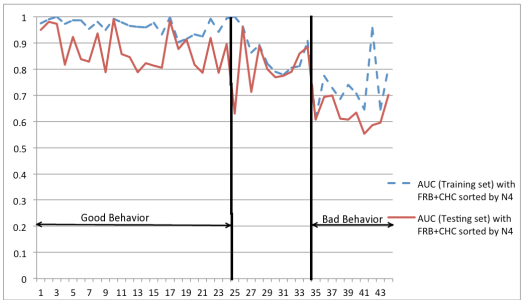
(d) AUC results with sBorder



(e) AUC results with sSafe



(f) AUC results with sRST



(g) AUC results with FRB+CHC

Figure 5.5: Average AUC results obtained from training and testing sets sorted by N4.

Table 5.6: Summary of the Intervals of F1, L3, and N4.

SMOTE	**sTL**	**sENN**	**sBorder**	**sSafe**	**sRST**	**FRB+CHC**
Good Behavior						
F1 $\geq$ 2.3018						
L3 $\leq$ 0.1080						L3 $\leq$ 0.1462
N4 $\leq$ 0.0808						N4 $\leq$ 0.2121
Bad Behavior						
F1 $\leq$ 0.6320						F1 $\leq$ 0.5275
N4 $\geq$ 0.3684						

behavior. Comparing the rules and the covered datasets, FRB+CHC covers the widest regions of good behavior of L3 and N4 data complexity measures. This indicates that FRB+CHC is more robust with the data complexity. The regions of the SMOTE-based methods are the same because of the same over-sampling method used.

5.4 Conclusion

A hybrid re-sampling method developed based on both over-sampling and under-sampling has been proposed. The synthetic samples of the minority class are generated based on fuzzy logic. To minimize the size of datasets and prevent over-generalization, CHC has been employed over the synthetic samples and the majority samples as a cleaning method to the over-sampled training set.

The proposed preprocessing method (FRB+CHC) is compared to SMOTE, sTL, sENN, sBorder, sSafe, and sRST on 44 datasets. To evaluate the performance of these seven sampling methods, the same SVM classifier has been used to obtain the experimental results. It is shown that FRB+CHC outperforms the other sampling methods on both

Table 5.7: Datasets Sorted by F1.

Dataset	F1	SMOTE	sTL	sENN	sBorder	sSafe	sRST	FRB+CHC
glass015vs2	0.1375	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior
yeast1458vs7	0.1757							
glass016vs2	0.2692							
yeast0359vs78	0.3113							
glass0146vs2	0.3487							
yeast1vs7	0.3534							
yeast1289vs7	0.3660							
glass2	0.3952							
ecoli0147vs2356	0.5275							Undefined
abalone19	0.5295							
abalone918	0.6320	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	
yeast0256vs3789	0.6939							
yeast4	0.7411							
ecoli0147vs56	0.9124							
ecoli0267vs35	0.9129							
ecoli067vs35	0.9205							
glass5	1.0185							
glass06vs5	1.0487							
yeast05679vs4	1.0507							
ecoli01vs235	1.1028							
ecoli0347vs56	1.1296							
yeast2vs8	1.1424							
ecoli0146vs5	1.3399							
cleveland0vs4	1.3442							
ecoli01vs5	1.3898							
glass4	1.4693							
glass04vs5	1.5422							
page_blocks13vs4	1.5470							
yeast2vs4	1.5793							
ecoli0346vs5	1.5952							
ecoli046vs5	1.6030							
ecoli0234vs5	1.6180							
ecoli034vs5	1.6323							
yeast02579vs368	1.6349							
ecoli067vs5	1.6922							
glass016vs5	1.8505							
led7digit02456789vs1	1.9568							
yeast6	1.9675							
ecoli0137vs26	2.3018	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior
vowel0	2.4579							
ecoli4	3.2474							
yeast5	4.1976							
shuttlec2vsc4	12.1322							
shuttlec0vsc4	12.9723							

Table 5.8: Datasets sorted by L3.

Dataset	L3	SMOTE	sTL	sENN	sBorder	sSafe	sRST	FRB+CHC
shuttlec2vsc4	0	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior
shuttlec0vsc4	0							
vowel0	0.0855							
led7digit02456789vs1	0.1010							
ecoli01vs235	0.1080	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	
ecoli034vs5	0.1083							
ecoli0147vs56	0.1088							
ecoli0234vs5	0.1094							
ecoli0347vs56	0.1137							
ecoli01vs5	0.1309							
ecoli046vs5	0.1341							
ecoli067vs5	0.1355							
ecoli0346vs5	0.1393							
page_blocks13vs4	0.1462							Undefined
ecoli0137vs26	0.1738							
ecoli067vs35	0.1760							
ecoli0146vs5	0.1777							
ecoli0147vs2356	0.1780							
ecoli0267vs35	0.1798							
glass04vs5	0.1833							
cleveland0vs4	0.1835							
yeast2vs8	0.2284							
yeast02579vs368	0.3510							
glass06vs5	0.3833							
yeast0359vs78	0.4046							
glass4	0.4325							
yeast0256vs3789	0.4506							
yeast2vs4	0.4843							
ecoli4	0.5							
glass016vs5	0.5							
glass5	0.5							
yeast5	0.5							
yeast6	0.5							
yeast4	0.5							
yeast05679vs4	0.5							
abalone918	0.5							
yeast1vs7	0.5							
yeast1289vs7	0.5	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior
glass016vs2	0.5							
glass2	0.5							
glass0146vs2	0.5							
glass015vs2	0.5							
yeast1458vs7	0.5							
abalone19	0.5							

Table 5.9: Datasets Sorted by N4.

Dataset	N4	SMOTE	sTL	sENN	sBorder	sSafe	sRST	FRB+CHC
shuttlec2vsc4	0	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior	Good Behavior
shuttlec0vsc4	0.0102							
glass04vs5	0.0400							
ecoli01vs5	0.0670							
glass4	0.0808	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	
ecoli0146vs5	0.0896							
ecoli0234vs5	0.0947							
ecoli4	0.0969							
ecoli046vs5	0.0988							
vowel0	0.1078							
ecoli0147vs56	0.1149							
ecoli0346vs5	0.1221							
ecoli0347vs56	0.1261							
ecoli034vs5	0.1300							
ecoli067vs5	0.1573							
ecoli0147vs2356	0.1585							
glass06vs5	0.1621							
yeast2vs4	0.1733							
yeast02579vs368	0.1793							
ecoli0267vs35	0.1870							
ecoli01vs235	0.1897							
glass016vs5	0.1926							
ecoli067vs35	0.1956							
glass5	0.1980							
ecoli0137vs26	0.2031							
yeast5	0.2121							Undefined
page_blocks13vs4	0.2245							
yeast6	0.2373							
yeast4	0.2787							
yeast0256vs3789	0.3047							
yeast2vs8	0.3194							
yeast05679vs4	0.3248							
abalone918	0.3630							
led7digit02456789vs1	0.3684							
yeast0359vs78	0.3758	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior	Bad Behavior
yeast1vs7	0.4228							
yeast1289vs7	0.4306							
glass016vs2	0.4311							
glass2	0.4352							
glass0146vs2	0.4399							
glass015vs2	0.4403							
cleveland0vs4	0.4511							
yeast1458vs7	0.4598							
abalone19	0.4909							

$F - measure$ and AUC. Our method has improved the $F - measure$ from 47% to 52% and the AUC from 78% to 80% when compared with sRST. To show the advantages of the proposed method, the over-sampling rate and the number of support vectors formed from SVM for different methods are also compared. FRB+CHC achieves good results under these criteria, which means that FRB+CHC obtains a good balance between accuracy and over-sampling rate. It also decreases the complexity of the learning model. The major reason is that CHC only selects the samples to increase the performance of the datasets, but not considering the locations of the samples. Therefore, the more representative samples are selected to form the training sets.

In the analysis with data complexity measures, the SMOTE-based hybrid methods cannot show a significant difference between them. In contrast, FRB+CHC uses fuzzy logic to over-sample the minority class samples. It shows a significant improvement over the previous methods and is more robust than them against the data complexity of the original datasets.

Chapter 6

An Under-sampling Method Based on Fuzzy Set Theory for Large Imbalanced Dataset (uFRB+CHC)

6.1 Introduction

Large imbalanced datasets will introduce difficulties to classification problems. The large data size and imbalanced nature may cause a high error rate of classifying the minority class samples and a long training time of the classification model. Therefore, re-sampling and data size reduction have become an important step to pre-process the data. In this chapter, an under-sampling strategy over a large imbalanced dataset is proposed, in which the samples of the majority class are selected based on fuzzy logic. To further reduce the data size, the evolutionary computational method of CHC [42] (Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation) is suggested.

In this chapter, experiments are carried out to show the performance of our proposed method, which is compared to different under-sampling methods. They are random under-sampling (RUS), condensed nearest neighbor rule (CNN) [60], Tomek Links (TL) [61], one-sided selection (OSS) [62], and neighborhood cleaning rule (NCL) [63]. A large imbalanced dataset from UCI Repository [77] is used as the dataset for evaluation. The Support Vector Machine (SVM) [80] is used as the tool for reaching a classification model from each re-sampled dataset, so as to evaluate the corresponding preprocessing method. The evaluation measures are based on the functions of precision and recall.

This chapter is organized as follows: Section 6.2 introduces the details of the proposed under-sampling strategy and the evaluation measures of this study. To show the effectiveness of our proposed method, the results and comparisons are discussed in Section 6.3. A conclusion is drawn in Section 6.4.

6.2 Methodology

In this section, the details of the proposed under-sampling method are discussed. The proposed method involves two stages. The majority class samples of the training sets are firstly under-sampled based on fuzzy logic. To further reduce the data size, CHC is then implemented to both minority and majority class samples.

6.2.1 Fuzzy Logic

In this chapter, fuzzy logic is used to cluster the majority class samples and select the samples according to their importance. Let the class $negative$ be the majority class and only λ training samples (X_α) of the class negative are considered, where $X_\alpha = (x_{\alpha 1}, \ldots, x_{\alpha\gamma})$ is a γ-dimensional vector, $\alpha = 1, 2, \ldots, \lambda$ and $x_{\alpha\beta}$ is the βth attribute value $(\beta = 1, 2, \ldots, \gamma)$ of the αth training sample. The θth fuzzy if-then rule can be written as follows:

$$\begin{aligned} &\text{Rule } \theta : \text{IF } z_1 \text{ is } A_1^\theta \text{ AND } \ldots \text{ AND } z_\gamma \text{ is } A_\gamma^\theta \\ &\qquad \text{THEN class = negative with } w_\theta \end{aligned} \tag{6.1}$$

where A_β^θ is a fuzzy term of the θth rule corresponding to the attribute z_β, $\beta = (1, 2, \ldots, \gamma)$ and $z = (z_1, z_2, \ldots, z_\gamma)$ is a γ-dimensional attribute vector, and w_θ is the θth rule weight. The number of rules is governed by the distribution of the samples. The Gaussian membership functions are used as the antecedent fuzzy sets, which are formed based on the distribution of the attributes. The choice of the shape and the arrangement of each label of the membership functions, and the use of the rule weight are introduced in the following subsections.

6.2.1.1 Membership Functions

When deciding the membership function of each label, the distribution of the attribute is considered. First, the mean value ($mean_\beta$) and standard deviation ($stdev_\beta$) of each attribute are calculated, where $\beta = 1, 2, \ldots, \gamma$. The samples closer to the mean value are

treated as more informative. Therefore, the membership function near the mean value is assigned with a narrower "bell" to cluster the samples with more fuzzy labels. An odd number should be assigned as the number of labels. Considering, L labels per attribute are employed and the Gaussian membership function of label k ($k = 1, 2, \ldots, L$) is defined as follows:

$$f_k(x_{\alpha\beta}) = e^{-\frac{(x_{\alpha\beta}-m_{\beta k})^2}{2\sigma_{\beta k}}}, \tag{6.2}$$

where $m_{\beta k}$ and $\sigma_{\beta k}$ are the mean and standard deviation of the kth label corresponding to the βth attribute respectively. Both $m_{\beta k}$ and $\sigma_{\beta k}$ are assigned based on $mean_\beta$ and $stdev_\beta$. Table 6.1 shows the method of setting the parameters of each membership function, and in Fig. 6.1, we use 5 labels as an example to explain the method. This setting of membership functions can cluster the samples near the mean value with more fuzzy labels.

Table 6.1: The Label Setting of Each Membership Function of the βth Attribute.

Label	$\boldsymbol{m_{\beta k}}$	$\boldsymbol{\sigma_{\beta k}}$
1	Area($\frac{1}{L+1}$)	$\frac{stdev_\beta * (L+1)/2}{L}$
2	Area($\frac{2}{L+1}$)	$\frac{stdev_\beta * (L-1)/2}{L}$
$\vdots$	$\vdots$	$\vdots$
$\frac{L+1}{2}$	$mean_\beta$	$\frac{stdev_\beta}{L}$
$\vdots$	$\vdots$	$\vdots$
L-1	Area($\frac{L-1}{L+1}$)	$\frac{stdev_\beta * (L-1)/2}{L}$
L	Area($\frac{L}{L+1}$)	$\frac{stdev_\beta * (L+1)/2}{L}$
Note: Area($\frac{1}{L+1}$) means the samples smaller than the value have occupied $\frac{1}{L+1}$ number of samples.		

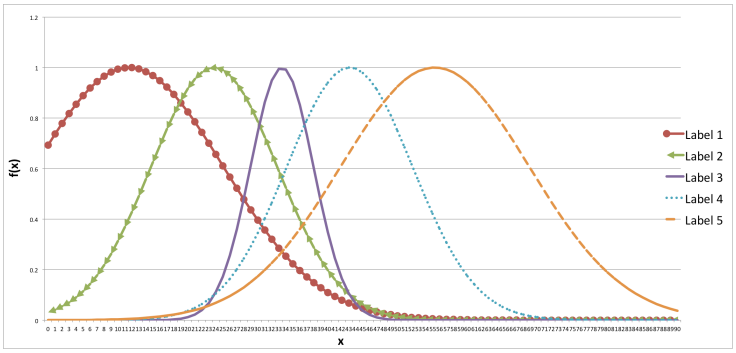


Figure 6.1: Arrangement of the membership function of each label. 5 labels are employed as an example.

6.2.1.2 Rule Weight

Rule weight is used to assign the degree of matching of each fuzzy rule over all the negative samples, so that the importance of each rule can be reflected. First, the fuzzy value of each sample is calculated. The fuzzy value of X_α for the θth fuzzy rule is defined as follows:

$$\mu_{A^\theta}(X_\alpha) = T(\mu_{A_1^\theta}(x_{\alpha 1}), \ldots, \mu_{A_\gamma^\theta}(x_{\alpha\gamma})), \tag{6.3}$$

where $\mu_{A_\beta^\theta}(x_{\alpha\beta})$ is the fuzzy value of the βth attribute for the θth fuzzy rule and the product T-norms are used. The rule weight (w_θ) is calculated by adding all the fuzzy values of each sample.

$$w_\theta = \sum_{\alpha=1}^{\lambda} (\mu_{A^\theta}(X_\alpha)). \tag{6.4}$$

6.2.1.3 Selection of the Majority Samples

After the rule base of the class negative is generated, the rules are randomly drawn based on the rule weight. The rule with a higher rule weight has a higher probability to be chosen. Then, a sample matching this rule is selected randomly to form the new dataset. These processes are repeated until the number of negative samples is twice that of positive samples. The above ratio of negative samples to positive samples of 2 to 1 is obtained through experiments.

6.2.2 Setting of CHC

After the under-sampling, the number of majority class samples is twice that of minority class and CHC is then applied. There are two important issues that need to be addressed clearly before the algorithm is employed: the representation of each chromosome and the definition of fitness function.

6.2.2.1 Chromosome Representation

CHC is used to further reduce the data size. The chromosomes are to represent subsets of the dataset. The details are the same as those for the previously proposed method (SMOTE+CHC), which can be found in Section 4.2.1 of Chapter 4.

6.2.2.2 Fitness function

In this study, the SVM is used as the evaluation method of CHC to obtain the subset with the highest classification rate. The fitness function is formed from $F-measure$ and AUC, which is introduced in Section 3.3.1 of Chapter 3.

Since both $F-measure$ and AUC are important measures on imbalanced datasets, a multi-objective fitness function is used here. If a chromosome X has a higher value of $F-measure$ and a lower value of AUC than that of chromosome Y, the difference between the chromosomes' $F-measure$ and the difference between the chromosomes' AUC will be compared. If the difference between the chromosomes' $F-measure$ is larger than that of AUC, chromosome X will be regarded as a better one; otherwise chromosome Y will be regarded as a better one.

6.3 Experimental Study

In this section, we present the experiments that compare our proposed method with other under-sampling methods. The dataset used can be found in UCI Repository [77].

The experiments involve RUS, CNN, TL, OSS, NCL, and our proposed method, which is named as uFRB+CHC. To measure the performance of the preprocessing methods, the same learning tool should be used among them. This tool is a Support Vector Machine (SVM) that attempts to obtain the classification model from the re-sampled training set. The program of all testing methods and the learning tool are based on KEEL, which is an open source software available at the Web [79]. $F-measure$ and AUC are used as

Table 6.2: Descriptions of the Selected Imbalanced Dataset.

Dataset	$N_{samp.}$	$N_{attr.}$	Min., Maj.(%)	IR
Census (Training/Testing)	57,008	41	(5.73, 94.27)	16.45
Census (Validation)	28504	41	(5.73, 94.27)	16.45

measures to analyze the results of the experimental methods, five trials are carried out and the average value of these measures of each method will be calculated.

As mentioned before, the large re-sampled training datasets will increase the complexity of the classification model. Therefore, the under-sampling rate and the number of support vectors formed from the SVM will also be compared.

6.3.1 Dataset

To evaluate the methods, a large dataset called Census from UCI is chosen. It has been divided into five parts evenly. The training set and testing set take two parts of them separately. The remaining part forms the validation set used to calculate the fitness function of CHC. Table 6.2 shows the details of the selected dataset, where the number of samples ($N_{samp.}$), the number of attributes ($N_{attr.}$), the percentage distributions of the minority and majority classes, and the imbalanced ratio (IR) are given. IR is the ratio of the number of majority class samples to the number of minority class samples.

6.3.2 Setup of Experiment

For CHC, the basic settings of the parameters are:

- Population size: 30.
- Divergence rate: 0.35.
- Threshold decreasing rate: 0.001.
- Kernel of SVM: Radial Basis Function.
- Number of Evaluations: 2,000.

SVM is used to compare the influence of each preprocessing methods. A radial basis function (RBF) is used as the SVM kernel since a non-linear classification model is needed and RBF is a common kernel to handle this problem. The RBF is defined as follows:

$$RBF = exp(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}\|^2) \tag{6.5}$$

where $\sigma > 0$ is the parameter to determine the width of the radial basis function. It controls the flexibility of the classifier. When σ decreases, the flexibility of the resulting classifier in fitting the training data increases, and this might lead to over-fitting easily. The value of σ is set as 0.01 for the experiment.

6.3.3 Results

Table 6.3 shows the $F - measure$ and AUC of each sampling method. Our proposed method of uFRB+CHC can obtain the best performance among all methods. The performance of TL and that of NCL are similar since the ideas of them are similar to remove the noisy and borderline samples. The under-sampling rate and the number of support vectors of the classification model for different methods are also shown in the table. The

Table 6.3: The Testing Results of Census.

Results	RUS	CNN	TL	OSS	NCL	uFRB+CHC
F-measure	0.1579	0.05753	0.02333	0.07913	0.02578	0.1702
AUC	0.6703	0.5095	0.5043	0.5163	0.5046	0.6869
Under-sampling Rate	0.8855	0.8455	0.05334	0.8592	0.1228	0.9083
Number of Support Vectors	6,396	8,799	40,083	8,024	36,690	4,381

under-sampling rate is defined as follows:

$$Rate_{under} = \frac{(N_{original} - N_{sampled})}{N_{original}} * 100\% \tag{6.6}$$

where $N_{sampled}$ is the number of samples in the re-sampled training set and $N_{original}$ is the number of samples in the original training set. uFRB+CHC can obtain the highest under-sampling rate. This shows that our method can use less training samples to achieve the high performance.

6.4 Conclusion

An under-sampling method over large imbalanced dataset has been proposed. The samples of the majority class are selected based on fuzzy logic. CHC is then applied to further reduce the data size. The proposed method (uFRB+CHC) is compared to RUS, CNN, TL, OSS, and NCL. To evaluate the performance of these six sampling methods, the same SVM classifier has been used to obtain the experimental results. It shows that our method outperforms the other sampling methods on both $F - measure$ and AUC. The under-sampling rate and the support vectors' number of the classification model are also compared. Our method achieves good results on all these measures, which means

the proposed method can select the most representative samples to form the training sets.

Chapter 7

Comparison

7.1 Introduction

In this thesis, three different preprocessing methods for imbalanced datasets have been presented in Chapters 4, 5, and 6 respectively. In this chapter, a general comparison among these three preprocessing methods using 44 imbalanced datasets from UCI Repository will be given out.

7.2 Summary

In this section, a summary of each preprocessing method is discussed. The first method is SMOTE+CHC, which is a hybrid data preprocessing method with two parts: Synthetic Minority Over-sampling Technique (SMOTE) and CHC algorithm (Cross-generational elitist selection, Heterogenous recombination and Cataclysmic mutation). SMOTE [22] is a common over-sampling method for imbalanced datasets. It creates new instances by interpolating several minority samples that join together. In [59], the problem of over-

generalization was mentioned. Hence, CHC is implemented to both synthetic samples and majority samples to under-sample the dataset and select the more representative samples to form the dataset.

The second method is Fuzzy Rule Base+CHC (FRB+CHC). It makes use of fuzzy logic to generate new samples of the minority class, and then uses CHC to reduce both synthetic samples and majority samples (the same latter part of SMOTE+CHC). In the over-sampling part, the fuzzy rules are first generated based on the samples of the minority class. The rules are randomly drawn based on the rule weight. The rule with a higher rule weight has a higher probability to be chosen. Then, the samples are produced using the selected rules as the criteria.

The third method is under-sampling Fuzzy Rule Base+CHC (uFRB+CHC), which is an under-sampling data preprocessing method. It focus on the large imbalanced datasets and uses fuzzy logic and CHC to reduce the datasets. This method is similar to FRB+CHC. The main difference is that the fuzzy rule base here is formed with the samples of the majority class. The Gaussian membership functions are used for the fuzzy terms. After the fuzzy rule base is generated, the rules are randomly drawn based on the rule weight. The samples matching the rules are selected to form the new dataset. Then, CHC is applied to the samples of the minority class and the under-sampled majority class to further reduce the size of dataset.

7.3 Comparison with Experimental Results

In this section, experimental results of the 44 imbalanced dataset from UCI Repository are presented. FRB+CHC using two different membership functions will be compared. One uses regular triangular membership functions, named as FRB+CHC, the other one uses Gaussian membership functions, named as FRB+CHCgau. SVM will be used as the learning tool to train the re-sampled dataset. F-measure and AUC are used as measures to analyze the results. The sampling rate and the number of support vectors formed from SVM will also be compared. At last, the behavior of each method will be analyzed.

7.3.1 Datasets

To study the methods, 44 datasets with different imbalanced ratio (IR) are chosen. IR is the ratio of the number of majority class samples to the number of minority class samples. Table 7.1 shows the details of the selected datasets, where the number of samples ($N_{samp.}$), the number of attributes ($N_{attr.}$), the percentage distribution of the minority and majority classes, IR, and the data complexity N4 for each dataset can be found. N4 is the nonlinearity based on a linear classifier by linear programming. A measure of the nonlinearity of a classifier with respect to a given dataset is suggested in [82]. This measure has been modified to study the nonlinearity of the class boundary in [83]. First, a test set is created by linear interpolation between random sample pairs from the same class. Then, the error rate on this test set is measured. In this study, the error rate, N4, is defined as $(1 - AUC)$. A 1 Nearest-Neighbor (1NN) is used as the linear classifier. Therefore, a larger value of N4 represents a higher complexity of the dataset.

Table 7.1: Descriptions of the Selected Imbalanced Datasets.

Dataset	$N_{samp.}$	$N_{attr.}$	Min., Maj.(%)	IR	N4
shuttlec2vsc4	129	9	(4.65, 95.35)	20.5	0.0000
shuttlec0vsc4	1829	9	(6.72, 93.28)	13.87	0.0102
glass04vs5	92	9	(9.78, 90.22)	9.22	0.0400
ecoli01vs5	240	6	(8.33, 91.67)	11	0.0670
glass4	214	9	(6.07, 93.93)	15.47	0.0808
ecoli0146vs5	280	6	(7.14, 92.86)	13	0.0896
ecoli0234vs5	202	7	(9.9, 90.1)	9.1	0.0947
ecoli4	336	7	(5.95, 94.05)	15.8	0.0969
ecoli046vs5	203	6	(9.85, 90.15)	9.15	0.0988
vowel0	988	13	(9.01, 90.99)	9.98	0.1078
ecoli0147vs56	332	6	(7.53, 92.47)	12.28	0.1149
ecoli0346vs5	205	7	(9.76, 90.24)	9.25	0.1221
ecoli0347vs56	257	7	(9.73, 90.27)	9.28	0.1261
ecoli034vs5	200	7	(10, 90)	9	0.1300
ecoli067vs5	220	6	(9.09, 90.91)	10	0.1573
ecoli0147vs2356	336	7	(8.63, 91.37)	10.59	0.1585
glass06vs5	108	9	(8.33, 91.67)	11	0.1621
yeast2vs4	514	8	(9.92, 90.08)	9.08	0.1733
yeast02579vs368	1004	8	(9.86, 90.14)	9.14	0.1793
ecoli0267vs35	224	7	(9.82, 90.18)	9.18	0.1870
ecoli01vs235	244	7	(9.83, 90.17)	9.17	0.1897
glass016vs5	184	9	(4.89, 95.11)	19.44	0.1926
ecoli067vs35	222	7	(9.91, 90.09)	9.09	0.1956
glass5	214	9	(4.2, 95.8)	22.78	0.1980
ecoli0137vs26	281	7	(2.49, 97.51)	39.14	0.2031
yeast5	1484	8	(2.96, 97.04)	32.73	0.2121
page_blocks13vs4	472	10	(5.93, 94.07)	15.86	0.2245
yeast6	1484	8	(2.36, 97.64)	41.4	0.2373
yeast4	1484	8	(3.43, 96.57)	28.1	0.2787
yeast0256vs3789	1004	8	(9.86, 90.14)	9.14	0.3047
yeast2vs8	482	8	(4.15, 95.85)	23.1	0.3194
yeast05679vs4	528	8	(9.66, 90.34)	9.35	0.3248
abalone918	731	8	(5.65, 94.25)	16.4	0.3630
led7digit02456789vs1	443	7	(8.35, 91.65)	10.97	0.3684
yeast0359vs78	506	8	(9.88, 90.12)	9.12	0.3758
yeast1vs7	459	7	(6.53, 93.47)	14.3	0.4228
yeast1289vs7	947	8	(3.16, 96.84)	30.57	0.4306
glass016vs2	192	9	(8.85, 91.15)	10.29	0.4311
glass2	214	9	(7.94, 92.06)	11.59	0.4352
glass0146vs2	205	9	(8.29, 91.71)	11.06	0.4399
glass015vs2	172	9	(9.88, 90.12)	9.12	0.4403
cleveland0vs4	177	13	(7.34, 92.66)	12.62	0.4511
yeast1458vs7	693	8	(4.33, 95.67)	22.1	0.4598
abalone19	4174	8	(0.77, 99.23)	129.44	0.4909

7.3.2 Setup of Experiment

For CHC, the basic settings of the parameters are:

- Population size: 50.
- Divergence rate: 0.35.
- Threshold decreasing rate: 0.001.
- k of k-NN classifier used as evaluation: 1.
- Number of Evaluations: 5,000.

The 1-Nearest Neighbor (1-NN) classifier is used as the evaluation method of CHC to obtain the subset with the highest classification rate. The fitness function is formed from $F - measure$ and AUC, which is introduced in Section 3.3.1 of Chapter 3.

Since both $F - measure$ and AUC are important measures on imbalanced datasets, a multi-objective fitness function is used here. If a chromosome X has a higher value of $F - measure$ ($F_X > F_Y$) and a lower value of AUC ($A_X < A_Y$) than that of chromosome Y, the difference between the chromosomes' $F - measure$ ($|F_X - F_Y|$) and the difference between the chromosomes' AUC ($|A_X - A_Y|$) will be compared. If $|F_X - F_Y| > |A_X - A_Y|$, chromosome X will be regarded as a better one; otherwise chromosome Y will be regarded as a better one.

SVM is used to weigh the influence of each preprocessing methods. A radial basis function (RBF) is used as the SVM kernel since a non-linear classification model is needed

and RBF is a common kernel to handle this problem. The RBF is defined as follows:

$$RBF = exp(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}\|^2) \tag{7.1}$$

where $\sigma > 0$ is the parameter to determine the width of the radial basis function. It controls the flexibility of the classifier. When σ decreases, the flexibility of the resulting classifier in fitting the training data increases, and this might lead to over-fitting easily. The value of σ is set as 0.01 and the tradeoff between training error and margin of SVM is set as 100. The above values are chosen through experiments.

We consider a 5-fold cross validation model to develop the experiments, i.e., dividing the whole dataset into five parts randomly, and combining four of them for training and the remaining part for testing. All the methods involve some random parameters, so ten experiments are carried out for each 5-fold cross validation model and the average value are calculated as the results. Therefore, totally 50 experiments were done.

7.3.3 Experimental Results

Tables 7.2 and 7.3 show the values of $F - measure$ and AUC respectively for each sampling method on the 44 datasets, and the results are sorted by N4 in ascending order. The last row shows the average value of each method on all datasets. FRB+CHC outperforms the other methods in terms of $F - measure$ and AUC. The $F - measure$ values have only a small difference among the methods. The performance of SMOTE+CHC and FRB+CHCgau are similar to each other.

Table 7.4 shows the under-sampling rate of each sampling method. The negative val-

ues mean the size of the re-sampled dataset is larger than the original one. The under-sampling rate is defined as follows:

$$Rate_{under} = \frac{(N_{original} - N_{sampled})}{N_{original}} * 100\% \tag{7.2}$$

where $N_{sampled}$ is the number of samples in the re-sampled training set and $N_{original}$ is the number of samples in the original training set. Table 7.5 shows the number of support vectors used to form the classification model. Obviously, the under-sampling rate of uFRB+CHC has the highest value, and the number of support vectors is related to the number of training samples. FRB+CHC uses the greatest number of support vectors in most of the datasets.

To show the behavior of different methods, Fig. 7.1 and 7.2 reveal the AUC and $F-measure$ results respectively sorted by N4 values in ascending order. The y-axis of each figure shows the average AUC or $F-measure$ results obtained from either the training or testing set. The x-axis shows the 44 datasets sorted by N4 values. The solid lines in the figures represent the average AUC or $F-measure$ results for the testing set; the dashed lines represent the average AUC or $F-measure$ results for the training set.

Both uFRB+CHC and FRB+CHC show the advantage on relaxing the over-fitting problem since the performances on training set and testing set are similar in both AUC and $F-measure$. However, uFRB+CHC shows its disadvantage in the value of $F-measure$ in Fig. 7.2(a). That means the precision is low and the difference between precision and recall is large. This is a common problem of under-sampling since some informative samples of the majority class may be eliminated, which makes the samples of the majority class classified wrongly. The performance of SMOTE+CHC and

Table 7.2: F-measure of Testing Datasets among Different Sampling Methods.

Dataset	uFRB+CHC	SMOTE+CHC	FRB+CHC	FRB+CHCgau
shuttlec2vsc4	0.6363	0.6103	0.6126	0.3493
shuttlec0vsc4	0.4533	0.9724	0.7964	0.8048
glass04vs5	0.3121	0.9933	0.9631	1
ecoli01vs5	0.5570	0.4392	0.6843	0.4560
glass4	0.2055	0.8190	0.7273	0.8074
ecoli0146vs5	0.6881	0.3762	0.7456	0.4005
ecoli0234vs5	0.6050	0.5577	0.6142	0.4988
ecoli4	0.5562	0.7931	0.7356	0.7594
ecoli046vs5	0.4629	0.3827	0.5225	0.4063
vowel0	0.2194	0.9833	0.9060	0.9559
ecoli0147vs56	0.6160	0.5164	0.6762	0.5499
ecoli0346vs5	0.7217	0.5985	0.6766	0.5478
ecoli0347vs56	0.6142	0.5913	0.5176	0.5503
ecoli034vs5	0.5969	0.5054	0.5829	0.4337
ecoli067vs5	0.5210	0.3787	0.6173	0.3393
ecoli0147vs2356	0.5552	0.5021	0.4043	0.4986
glass06vs5	0.5484	0.9866	0.9783	0.9553
yeast2vs4	0.1366	0.6996	0.7015	0.7127
yeast02579vs368	0.6705	0.7437	0.7747	0.7826
ecoli0267vs35	0.5128	0.3856	0.4592	0.4420
ecoli01vs235	0.4820	0.4844	0.4224	0.5030
glass016vs5	0.1377	0.7548	0.7694	0.7259
ecoli067vs35	0.4700	0.5108	0.4308	0.6064
glass5	0.3734	0.6583	0.7533	0.7477
ecoli0137vs26	0.2489	0.4306	0.2929	0.2554
yeast5	0.4819	0.5146	0.4476	0.4674
page_blocks13vs4	0.5377	0.3563	0.1803	0.1763
yeast6	0.5398	0.3577	0.3288	0.3185
yeast4	0.7599	0.3076	0.3533	0.3207
yeast0256vs3789	0.7349	0.5624	0.5899	0.5891
yeast2vs8	0.6763	0.7068	0.6967	0.6967
yeast05679vs4	0.4817	0.5066	0.4786	0.4885
abalone918	0.3691	0.5221	0.5732	0.5661
led7digit02456789vs1	0.3268	0.7308	0.6746	0.7333
yeast0359vs78	0.7071	0.4117	0.3470	0.3695
yeast1vs7	0.5616	0.3120	0.3161	0.2991
yeast1289vs7	0.3253	0.1851	0.1967	0.1741
glass016vs2	0.1755	0.2102	0.2001	0.2161
glass2	0.6020	0.2484	0.2019	0.2910
glass0146vs2	0.7536	0.2823	0.2597	0.2984
glass015vs2	0.3269	0.2137	0.2049	0.2148
cleveland0vs4	0.1357	0.0923	0.1687	0.0542
yeast1458vs7	0.1812	0.1585	0.1557	0.1662
abalone19	0.0287	0.0437	0.0482	0.0495
Mean	0.4683	0.5090	0.5179	0.4904

Table 7.3: AUC of Testing Datasets among Different Sampling Methods.

Dataset	uFRB+CHC	SMOTE+CHC	FRB+CHC	FRB+CHCgau
shuttlec2vsc4	0.8796	0.9440	0.9493	0.7691
shuttlec0vsc4	0.6978	0.9731	0.9812	0.9823
glass04vs5	0.8140	0.9988	0.9732	1
ecoli01vs5	0.7245	0.6659	0.8159	0.6761
glass4	0.5949	0.9333	0.9230	0.9323
ecoli0146vs5	0.7523	0.6260	0.8371	0.6381
ecoli0234vs5	0.7277	0.7181	0.8289	0.6914
ecoli4	0.7720	0.9244	0.9368	0.9278
ecoli046vs5	0.6858	0.6395	0.7880	0.6548
vowel0	0.6538	0.9982	0.9892	0.9952
ecoli0147vs56	0.7627	0.6905	0.8578	0.7048
ecoli0346vs5	0.8097	0.7170	0.8459	0.6951
ecoli0347vs56	0.7241	0.7511	0.7888	0.7258
ecoli034vs5	0.7678	0.6747	0.8217	0.6422
ecoli067vs5	0.7336	0.6245	0.8125	0.6083
ecoli0147vs2356	0.7250	0.6891	0.8054	0.6837
glass06vs5	0.9056	0.9895	0.9840	0.9595
yeast2vs4	0.5737	0.8656	0.8757	0.8772
yeast02579vs368	0.8505	0.9041	0.9125	0.9091
ecoli0267vs35	0.7102	0.6405	0.8176	0.6638
ecoli01vs235	0.7042	0.6758	0.7866	0.6950
glass016vs5	0.5347	0.8979	0.9186	0.8781
ecoli067vs35	0.7402	0.6943	0.7860	0.7523
glass5	0.7559	0.8515	0.8967	0.8789
ecoli0137vs26	0.6035	0.7294	0.6306	0.5989
yeast5	0.7875	0.9683	0.9621	0.9649
page_blocks13vs4	0.8622	0.6847	0.7120	0.7043
yeast6	0.9174	0.8735	0.8879	0.8798
yeast4	0.8258	0.8177	0.7991	0.8076
yeast0256vs3789	0.9305	0.8038	0.7691	0.7940
yeast2vs8	0.8011	0.7852	0.7739	0.7739
yeast05679vs4	0.7442	0.7934	0.7899	0.7971
abalone918	0.6766	0.8745	0.8597	0.8863
led7digit02456789vs1	0.8869	0.8946	0.8844	0.8975
yeast0359vs78	0.8484	0.7289	0.6062	0.6163
yeast1vs7	0.7860	0.6777	0.6932	0.6801
yeast1289vs7	0.6694	0.7201	0.6990	0.7073
glass016vs2	0.5516	0.6239	0.6114	0.6453
glass2	0.8824	0.6648	0.6078	0.7202
glass0146vs2	0.9068	0.6717	0.6336	0.7026
glass015vs2	0.5840	0.5905	0.5530	0.5727
cleveland0vs4	0.5256	0.5210	0.5857	0.4926
yeast1458vs7	0.5970	0.6638	0.5958	0.6329
abalone19	0.6675	0.7166	0.7016	0.7288
Mean	0.7421	0.7703	0.8020	0.7624

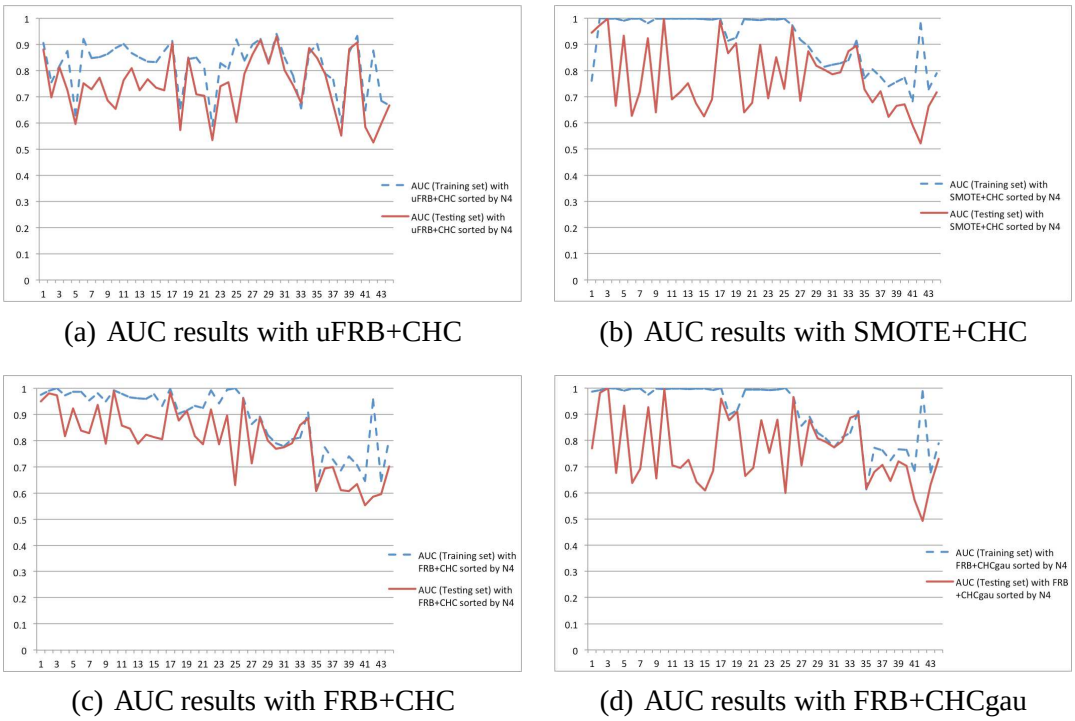
Table 7.4: Under-sampling Rate of Training Sets among Different Sampling Methods.

Dataset	uFRB+CHC(%)	SMOTE+CHC (%)	FRB+CHC (%)	FRB+CHCgau (%)
shuttlec2vsc4	91.86	3.86	3.43	3.80
shuttlec0vsc4	87.49	3.32	3.17	3.32
glass04vs5	90.78	4.39	2.90	2.50
ecoli01vs5	84.38	3.58	1.22	1.14
glass4	94.39	2.68	0.95	0.94
ecoli0146vs5	86.43	3.34	2.73	2.71
ecoli0234vs5	81.56	4.46	5.50	2.59
ecoli4	94.42	1.59	1.40	1.46
ecoli046vs5	81.90	3.99	2.72	1.89
vowel0	88.46	4.82	4.29	4.45
ecoli0147vs56	86.07	3.39	1.73	0.61
ecoli0346vs5	81.95	3.54	3.12	3.55
ecoli0347vs56	81.71	5.60	3.06	1.28
ecoli034vs5	81.50	4.43	2.73	4.09
ecoli067vs5	83.30	4.78	1.68	2.63
ecoli0147vs2356	84.23	4.05	1.50	1.83
glass06vs5	92.13	2.71	1.67	4.33
yeast2vs4	89.54	3.13	3.89	2.89
yeast02579vs368	90.64	3.46	2.52	2.71
ecoli0267vs35	81.36	4.03	2.63	1.17
ecoli01vs235	81.76	4.04	0.59	1.29
glass016vs5	95.65	2.84	1.03	1.22
ecoli067vs35	81.65	5.19	4.15	1.74
glass5	95.09	1.57	-0.11	1.46
ecoli0137vs26	95.46	1.32	-0.28	0.29
yeast5	97.57	1.66	2.09	1.07
page_blocks13vs4	89.04	2.88	0.86	2.05
yeast6	98.03	2.54	1.88	0.69
yeast4	96.16	1.69	1.16	-0.10
yeast0256vs3789	90.39	5.70	0.92	1.25
yeast2vs8	96.01	1.86	0.46	-0.27
yeast05679vs4	90.58	5.71	2.48	1.66
abalone918	93.33	1.72	1.44	1.93
led7digit02456789vs1	90.91	3.93	7.89	4.64
yeast0359vs78	88.44	4.90	2.14	-1.17
yeast1vs7	92.05	3.39	3.31	1.04
yeast1289vs7	95.99	2.50	2.35	0.34
glass016vs2	91.01	0.78	2.11	2.24
glass2	90.54	2.07	0.42	1.93
glass0146vs2	90.85	0.94	1.40	1.45
glass015vs2	89.68	2.06	2.28	3.01
cleveland0vs4	85.84	3.26	1.14	-0.71
yeast1458vs7	95.27	1.13	2.51	0.64
abalone19	99.01	-1.04	0.31	0.53
Mean	89.65	3.13	2.17	1.77

Table 7.5: Number of Support Vectors of the Classification Model formed by SVM.

Dataset	uFRB+CHC	SMOTE+CHC	FRB+CHC	FRB+CHCgau
shuttlec2vsc4	8.2	76.92	85.34	78.3
shuttlec0vsc4	110.2	361.88	944.42	939.12
glass04vs5	3	9	12.28	9.2
ecoli01vs5	29.8	156.88	185.74	169.08
glass4	3.4	18.42	22.84	18.74
ecoli0146vs5	30.2	178.38	213.86	191.42
ecoli0234vs5	29.4	133.64	149.7	143.46
ecoli4	10.8	58.46	66.4	60.56
ecoli046vs5	29	135.02	154.64	144.48
vowel0	16.4	34.04	92.08	54.38
ecoli0147vs56	36.2	213.86	257.72	241.76
ecoli0346vs5	29	136.88	155.64	143.94
ecoli0347vs56	37	168.64	195.96	188.58
ecoli034vs5	29	132.06	152.24	140.12
ecoli067vs5	29	143.38	169.8	154.12
ecoli0147vs2356	41.4	223.36	262.1	252.52
glass06vs5	3.4	14.3	17.5	13.4
yeast2vs4	32.4	165.68	152	151.26
yeast02579vs368	39.2	240.74	205.3	156.98
ecoli0267vs35	32.8	149.86	171.62	166.08
ecoli01vs235	34.8	166.3	191.52	185.02
glass016vs5	3	20.84	25.7	19.26
ecoli067vs35	32	147.16	167.42	163.76
glass5	3.6	21.24	26.76	19.46
ecoli0137vs26	10.2	168.28	177.68	155.08
yeast5	15.6	188.18	219.5	199.08
page_blocks13vs4	40.8	322.26	352.02	347.7
yeast6	16.4	331.1	366.8	339.36
yeast4	34	459.56	468.16	491.16
yeast0256vs3789	48.8	376.64	310.5	309.82
yeast2vs8	11.8	186.04	131.02	135.4
yeast05679vs4	32.8	206.48	212.36	215.66
abalone918	35.8	332.66	371.6	348.46
led7digit02456789vs1	10.8	58.48	67.2	40.8
yeast0359vs78	40.6	257.8	225.08	215.56
yeast1vs7	27.2	235.1	218.98	236.42
yeast1289vs7	28.2	507.2	450.9	508.8
glass016vs2	10.6	113.58	101.18	108.3
glass2	13.2	121	113.9	118.02
glass0146vs2	13.2	119.52	108.38	111.84
glass015vs2	10	108.2	91.16	98.6
cleveland0vs4	19.4	124.64	136.14	132.5
yeast1458vs7	24.4	417.02	397.14	408.6
abalone19	28.2	1659.98	2343.46	1890.46

FRB+CHCgau are similar.



(a) AUC results with uFRB+CHC

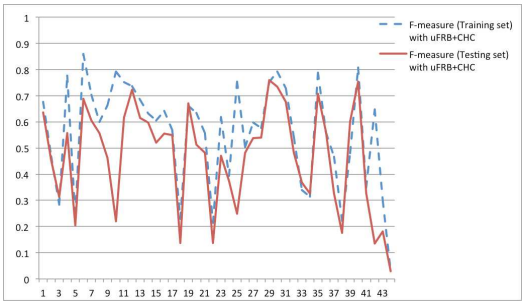
(b) AUC results with SMOTE+CHC

(c) AUC results with FRB+CHC

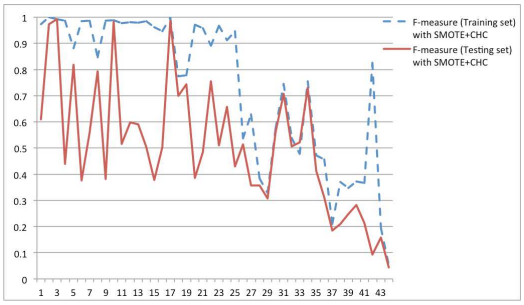
(d) AUC results with FRB+CHCgau

Figure 7.1: Average AUC results obtained from training and testing set sorted by N4.

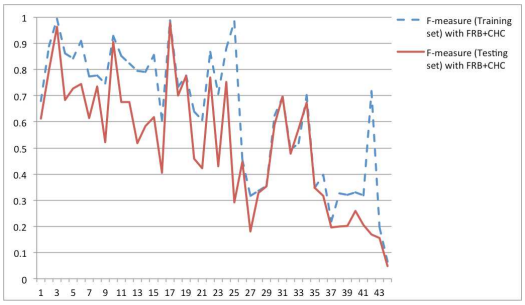
Although the implementation of FRB+CHC and FRB+CHCgau are similar and they only differ in terms of the membership function used at the over-sampling stage, their performance has a great difference. Fig. 7.3 and 7.4 show an example of the distribution of the positive samples and negative samples after the re-sampling of FRB+CHC and FRB+CHCgau respectively. The circle dots correspond to the samples of the majority class. The square dots correspond to the samples of the original minority class. The triangle dots correspond to the synthetic samples. Fig. 7.4 show that the synthetic samples are generated densely around some of the original minority samples. On the contrary, the synthetic samples in Fig. 7.3 are distributed more evenly in the area of the original minority samples. Therefore, FRB+CHCgau runs into the over-fitting problem more easily. Fig. 7.5 shows an example after the implementation of SMOTE+CHC. The distribution of the synthetic samples is similar to that of FRB+CHCgau. Therefore, their



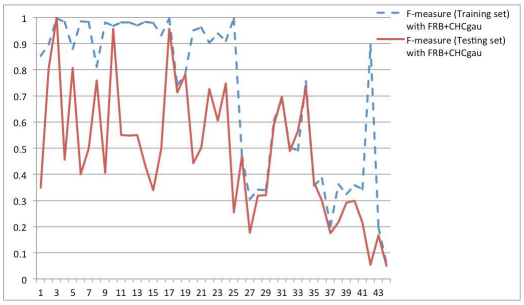
(a) F-measure results with uFRB+CHC



(b) F-measure results with SMOTE+CHC



(c) F-measure results with FRB+CHC



(d) F-measure results with FRB+CHCgau

Figure 7.2: Average F-measure results obtained from training and testing set sorted by N4.

experimental results are nearly the same.

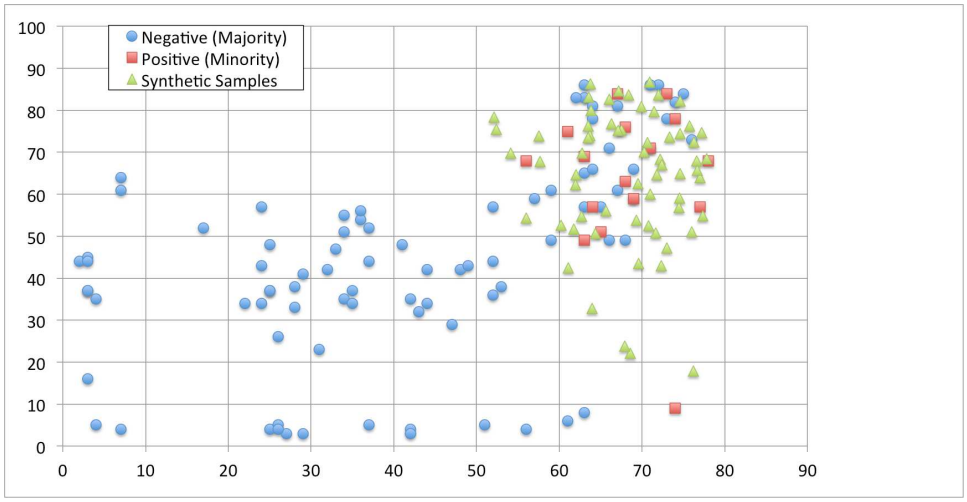


Figure 7.3: Distribution of the samples after the implementation of FRB+CHC.

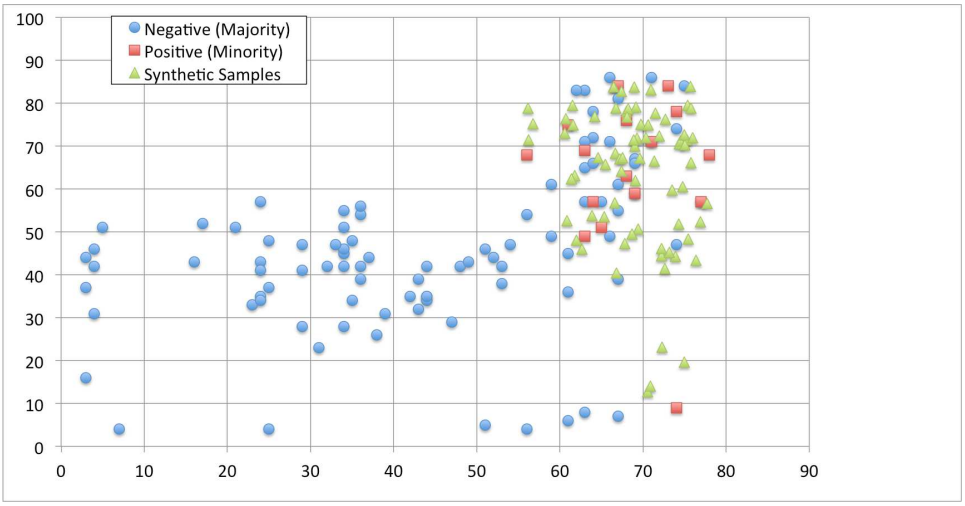


Figure 7.4: Distribution of the samples after the implementation of FRB+CHCgau.

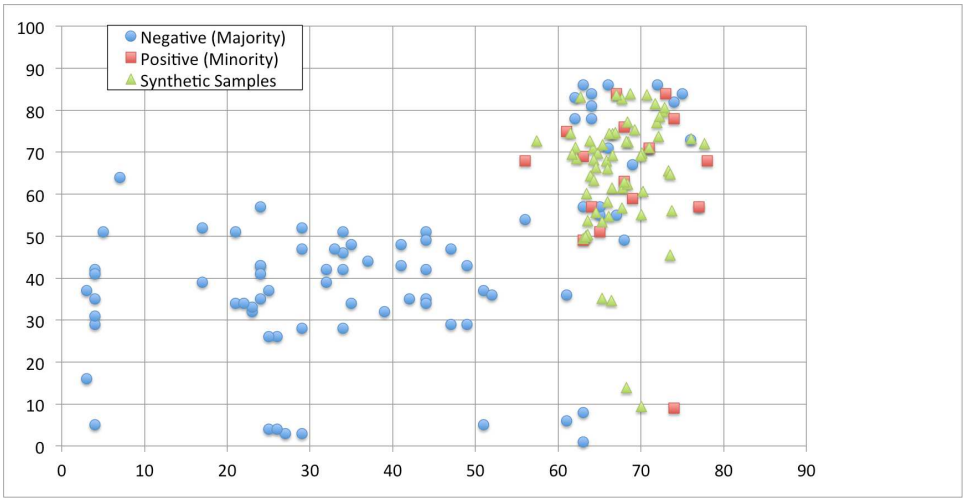


Figure 7.5: Distribution of the samples after the implementation of SMOTE+CHC.

7.4 Conclusion

In this chapter, four preprocessing methods presented in Chapters 4, 5, and 6 are compared. The FRB+CHC method proposed in Chapter 5 with triangular membership functions outperforms the other preprocessing methods in both AUC and $F-measure$ values. As it is a hybrid preprocessing method, the number of samples has not decreased

in a large rate. There is only around 1.7% of under-sampling rate. Therefore, it uses more support vectors to form the classification model and increases the complexity of the classifier. On the contrary, the uFRB+CHC method can offer a high under-sampling rate, which is around 89.7%. However, its AUC and $F - measure$ values are inferior to the other methods, which is a common phenomenon of under-sampling methods. Example class distribution graphs of the sampled dataset are shown to explain the different performance when the shape of the membership functions is changed in FRB+CHC, and the similarity of the performance of SMOTE+CHC and FRB+CHCgau.

To conclude, FRB+CHC with triangular membership function should first be considered to deal with the imbalanced dataset. If the size of the re-sampled dataset is too large, uFRB+CHC could be an alternative.

Chapter 8

Predicting Protein-Ligand Binding Site using Support Vector Machine and Hybrid Preprocessing Method

8.1 Introduction

In this chapter, the hybrid preprocessing method of FRB+CHC presented in Chapter 5 is applied on the datasets of binding sites before classification in order to improve the results in Chapter 3. The comparison among the preprocessing methods presented in this thesis has been shown in Chapter 7 and FRB+CHC is found to be superior to the other methods in terms of both $F - measure$ and AUC values.

SVM is employed to classify the protein-ligand binding sites by using 29 proteins' attributes as discussed in Chapter 3. To solve the imbalanced problem of the training dataset, random under-sampling method is used previously. In this chapter, FRB+CHC, which is a hybrid preprocessing method proposed in Chapter 5, is applied before the

training of SVM. Our new method is named as SVMBs2; while the old method in Chapter 3 is named as SVMBs1.

Two benchmark datasets are used to evaluate our methods. The first one involves 210 bound structures and the other one involves 198 drug-target complexes. Both of them are developed in MetaPocket. SVMBs2 is first compared to SVMBs1, which used random under-sampling method as the preprocessing. Then, our approach is compared with five other methods. They are LIGSITE^{CSC}, PASS, SURFNET, Q-SiteFinder, and MetaPocket.

8.2 Methodology

8.2.1 Overall Process

This section describes the overall process of the proposed method for predicting the protein-ligand binding sites. Each site is represented by a center grid point. First, a 3D grid is generated surrounding the protein based on its structure information of each protein from Protein Data Bank (PDB) [6]. Then, 29 properties (attributes) of each grid point are obtained. The details of each property are introduced in Section 3.2.2 of Chapter 3. The grid resolution of the training dataset and testing dataset are assigned differently, which are 2.5Å and 1Å respectively. A decrease of grid resolution can reduce the data size greatly. However, if the spacing value is too large, the calculated values of each grid point will become unreliable. Therefore, the spacing value of training dataset is just a bit larger than that of testing dataset to decrease the data size. The radii for common

atoms are from 1.1Å to 1.948Å, so the grid resolution should not be set greater than 3.9Å.

After the attributes of each grid point are calculated, the FRB+CHC method is applied on the training dataset to solve the imbalanced problem. The change of data size is shown in Table 8.1, where the grid points of the binding sites are represented by the positive samples. SVM is employed to the re-sampled training dataset to form the classification model, which is used to classify the grid points of the testing dataset. The grid points, which are predicted as the binding sites, are clustered into different groups by K-means clustering [84]. The initial number of clusters and centroids are set based on groups of neighboring predicted grid points. The initial centroids of the groups of neighboring grid points are found. Then, the K-means clustering [84] is performed for the grid points. After the K-means clustering, each cluster is represented by a final centroid. Fig. 8.1 shows the overall process of the proposed predicting method.

Table 8.1: Change of data size before and after applying FRB+CHC.

	Before	After
Number of negative samples	263,289	131,816
Number of positive samples	5,206	134,644

8.2.2 Datasets

In this study, the same training dataset, which has been introduced in Chapter 3, is used and shown in Table 8.2. In this chapter, two different testing datasets are used to evaluate our method against the other six methods. They are 210 bound structures and 198 drug-target complexes (that was used in Chapter 3, which are developed in MetaPocket.

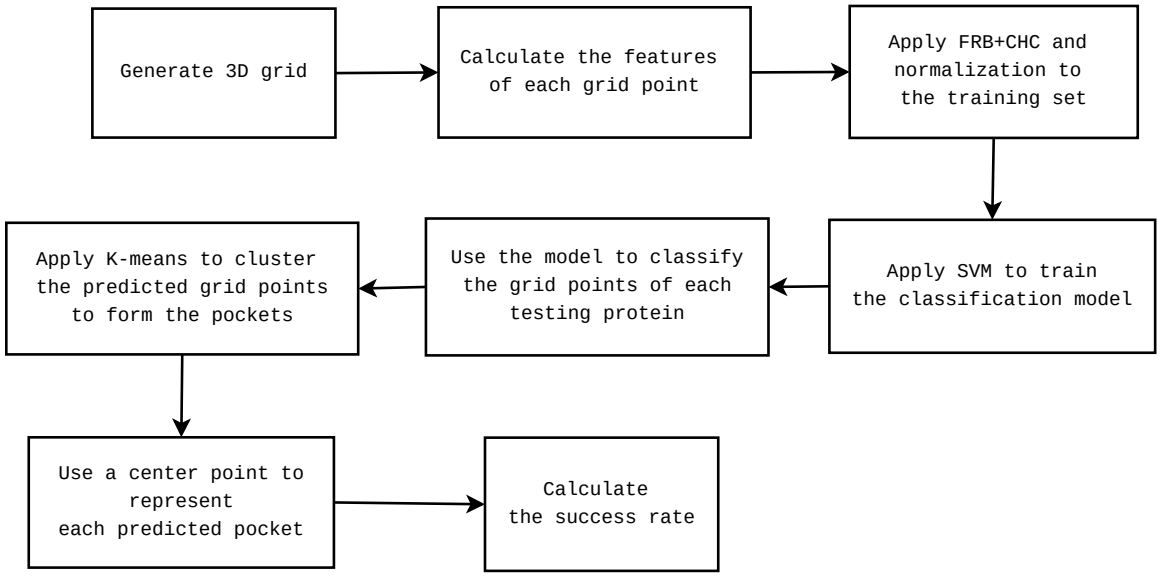


Figure 8.1: Flowchart for the proposed predicting method.

Table 8.2: Training Data Set.

1pkj	3gd9	1lf3	3lem	1llo
1ybu	4tpi	3h72	2j4e	1rn8
2v8l	1x2b	1g97	2zhz	3a0t
1o26	1rzu	1znz	1ojz	1sqf
2gga	3gh6	3d1g	2jgv	1dy3
1jyl	2e1t	2ywm	1kwc	2g28
3d4p	2wyw	2dtt	1tjw	2za1
2art	1u7z	3gid	1i1h	2w1a

8.3 Evaluation

To evaluate the performances of SVMBs2, SVMBs1 and the other five methods, the same measure is used. First, several clusters are formed from the predicted grid points by K-means clustering and each cluster is represented by a center grid point. Only the three largest clusters are selected to do the identification of binding sites since most of the ligands bind to large pockets [38]. Then, if the center grid points are located at the real pocket sites (i.e. the distance between the center grid points and any atoms of the

ligand is within 4Å), the prediction will be counted as a hit, which means the predicted binding site is identified correctly. There are sometimes more than one binding site within a protein and the prediction may identify more than one binding site correctly at the same time. In this case, only one hit in the larger cluster will be counted. The success rate is calculated by the following equation to compare the performance of different methods:

$$success\_rate = \frac{N_{HIT}}{N_P} * 100\% \tag{8.1}$$

where N_{HIT} is the number of proteins that at least one binding sites can be located correctly and N_P is the total number of proteins in the dataset.

8.4 Results

This section shows the comparison of our method and the other methods. In the following tables, top 1 represents the success rate of the largest cluster; top 1-2 represents the success rate of the two largest clusters; top 1-3 represents that of the three largest clusters.

8.4.1 Improvement of SVMBs2 by using FRB+CHC as preprocessing method

The main difference between SVMBs2 and SVMBs1 is the preprocessing method used before the training of the SVM classification model. SVMBs2 takes advantage of FRB+CHC while SVMBs1 uses random under-sampling method. They are evaluated on the 198 drug-target complexes and 210 bounded structures datasets. Table 8.3 shows the com-

parison of the success rate (8.1) of these datasets. In the 210 bounded dataset, SVMBs2 has improved the success rate by 6% for all the top 3 predictions. In the 198 drug-target complexes, a significant improvement of SVMBs2 over SVMBs1 is illustrated. The success rate is increased by 7% at the top 1 prediction and 4% for all the top three predictions. Fig. 8.2 shows an example for the different prediction of binding sites between SVMBs2 and SVMBs1. The real ligands are represented by red sticks. The predicted pockets of SVMBs2 and SVMBs1 are represented by blue spheres and magenta spheres, respectively.

There are totally 408 proteins in the two testing datasets. Table 8.4 shows the distribution of the number of chains of these proteins and the success rate of SVMBs2 and SVMBs1 under different numbers of the chains. Most of the proteins are less than three chains and the number of chains is more likely to be an even number. Although the success rate is decreased by the increased number of chains, SVMBs2 has improved the success rate from 3% to 8% for all the top 3 predictions. Overall, SVMBs2 has improved the performance of prediction for different datasets. This shows that the preprocessing method is an important on doing the prediction of binding sites.

Table 8.3: Comparison of SVMBs2 and SVMBs1 on Success Rate (%) for Different Datasets.

Dataset	Method	Top 1	Top 1-2	Top 1-3
210 bounded structures	SVMBs2	71.4	85.2	90.0
	SVMBs1	66.7	78.6	83.8
198 drug-target complexes	SVMBs2	68.7	81.3	85.9
	SVMBs1	61.6	76.8	81.8

Table 8.4: Comparison of SVMBs2 and SVMBs1 on Success Rate for Different Number of Chains.

Chain No.	Number of Proteins	Top 1-3 of SVMBs2 (%)	Top 1-3 of SMVBs1 (%)
1	152	92.8	90.1
2	176	92.0	85.2
3	18	83.3	77.8
4	49	71.4	67.3
>=5	13	46.2	38.5

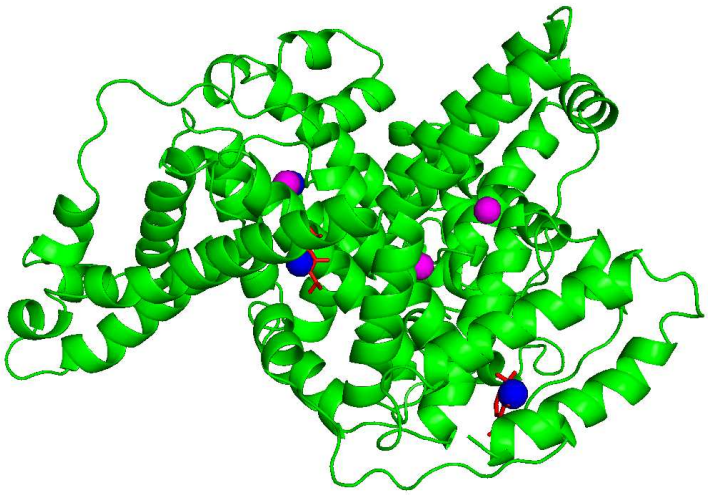


Figure 8.2: The real ligand (red) binding site and the predicted pockets for protein 1e7a. The predicted pockets of SVMBs1 (magenta) and SVMBs2 (blue) are shown in spheres.

8.4.2 Improvement of SVMBs2 over the other prediction methods

Table 8.5 shows the success rate of SVMBs2 and the other five prediction methods. The success rate is calculated by adding the results of two testing datasets (198 drug-target complexes and 210 bounded structure). Overall, SVMBs2 performs better than the other predicting methods. Although the success rate at top 1 prediction of SVMBs2 is a bit lower than that of MetaPocket, the success rate at top 3 prediction of SVMBs2 has improved by 3%. Table 8.6 shows the number of hit proteins among the six predicting

methods. SVMBs2 can locate the binding sites of 286 proteins correctly at top 1 prediction. There are 54 and 19 proteins that their binding sites can be located correctly at top 2 and top 3 prediction, respectively. There are still 49 proteins that their binding sites cannot be located correctly. Overall, SVMBs2 can identify the binding sites correctly with the highest number of proteins over the other five methods.

Table 8.5: Success Rate (%) of Top 3 Binding Sites Prediction with SVMBs2 and the Other 5 Predicting Methods.

Method	Top 1	Top 1-2	Top 1-3
SVMBs2	70.1	83.3	88.0
MetaPocket	71.3	80.6	84.8
LIGSITE^{CSC}	59.1	68.9	73.5
PASS	43.4	61.0	67.9
Q-SiteFinder	56.6	69.9	76.2
SURFNET	32.8	40.7	45.3

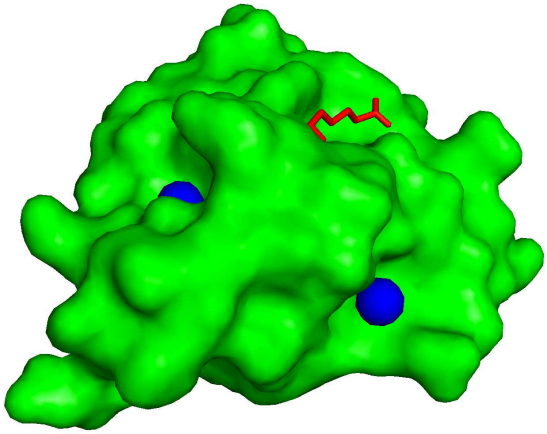
Table 8.6: Number of Hit Proteins of Top 3 Binding Sites Prediction with SVMBs2 and the Other 5 Predicting Methods.

Method	Top 1	Top 2	Top 3	None
SVMBs2	286	54	19	49
MetaPocket	291	38	17	62
LIGSITE^{CSC}	241	40	19	98
PASS	177	72	28	131
Q-SiteFinder	231	54	26	97
SURFNET	134	32	19	223

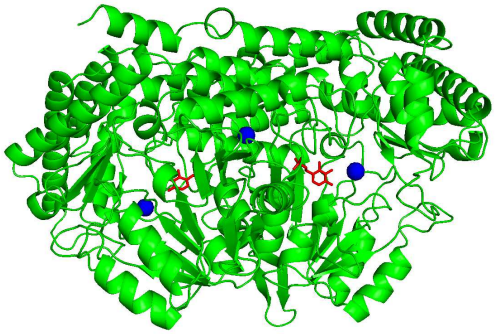
8.4.3 Discussion

SVMBs2 still has some limitations. There are 49 proteins that the correct binding sites cannot be identified in the three largest predicted pockets. From these cases, two limitations of SVMBs2 can be concluded. The first limitation is that ligands are bound to the atoms at a flat region. Since SVMBs2 tends to predict the binding sites in a cleft, the flat region is likely to be discarded. Totally, 32 cases belong to this category. The other limitation is that ligands are bound to some small binding sites. Since SVMBs2 only selects the three largest predicted pockets for classification, the smaller binding sites are

not easy to be discovered. There are 17 cases belonging to this category. Fig. 8.3 shows two examples of the difficult structures mentioned above. The real ligands are shown in red sticks. The predicted binding sites are shown in blue spheres.



(a) 2pk4.



(b) 1bj4.

Figure 8.3: Examples of the two limitations of SVMBs2. (a) The ligand binds to the atoms at a flat region. (b) The ligands bind to small binding sites.

8.5 Conclusion

The prediction of the protein-ligand binding site has been investigated in this chapter. A preprocessing method (FRB+CHC) has been added to solve the imbalanced problem

of protein dataset. FRB+CHC outperforms the other preprocessing methods in terms of $F - measure$ and AUC in our previous study. Then, SVM is employed to train a classification model and locate the grid points, which are most likely to form the binding sites. K-means is applied to cluster the grid points to form the predicted pockets, and we select the three largest pockets to evaluate the performance.

The method proposed in this chapter is named as SVMBs2, which is first compared with our previous method that uses random under-sampling as the preprocessing method. SVMBs2 has a significant improvement in both the 198 drug-target complexes and 210 bounded structures datasets. SVMBs2 has improved the success rate from 3% to 8 % for proteins of different number of chains. Moreover, SVMBs2 is compared to MetaPocket, LIGSITE^{CSC}, PASS, Q-SiteFinder, and SURFNET. Although the success rate at top 1 prediction of SVMBs2 is a bit lower than that of MetaPocket, the success rate at top 3 prediction of SVMBs2 has improved by 3%. Overall, SVMBs2 can locate the binding sites correctly for the largest number of the proteins among all prediction methods mentioned in this chapter.

Chapter 9

Conclusion

9.1 Achievements

In this thesis, a prediction method of protein-ligand binding site using Support Vector Machine (SVM) with 29 protein properties and three pre-processing methods of imbalanced datasets are developed. Two of the pre-processing methods use hybrid re-sampling methods and the other one uses an under-sampling method. These studies improve the ability of a classifier to deal with imbalanced datasets, and the success rate of the binding sites prediction.

The details of the proposed prediction method for protein-ligand binding site are presented in Chapter 3. SVM is employed to classify the grid points near the protein surface for the binding sites. It make uses of 29 different protein properties, including geometric characteristics, interaction potential, distance from protein, conservation score, and the properties of the grid points nearby to do the classification. Two datasets (LigASite and 198 drug-target complexes) are used to test and evaluate the success rate of the proposed method. Our method is compared to LIGSITE, LIGSITE^{CSC}, SURFNET,

Fpocket, PocketFinder, Q-SiteFinder, ConCavity, and MetaPocket. For the LigASite dataset, the proposed method is shown to offer more comprehensive results than the other methods as less proteins have the binding sites located wrongly. For the 198 drug-target complexes, the proposed method outperforms the other methods, and shows the highest success rate to identify the binding sites.

The three proposed pre-processing methods are discussed in Chapter 4 to Chapter 6. The first one is called SMOTE+CHC, which is presented in Chapter 4. This proposed sampling method consists of two stages. SMOTE is first employed to generate new samples of the minority class. Then, CHC is applied on the synthetic samples and the samples of the majority class to do under-sampling. The proposed method is compared to RUS, TL, ROS, SMOTE+TL on 22 datasets. All the over-sampling and hybrid methods get the similar results. SMOTE+CHC shows its ability of obtaining the lowest over-sampling rate while keeping the advantages of hybrid methods.

The second pre-processing method is called FRB+CHC, which is presented in Chapter 5. This proposed hybrid method generates new samples of the minority class based on a fuzzy rule base, and CHC is then applied on the synthetic samples and the samples of majority class. FRB+CHC is compared to different over-sampling and hybrid methods, including SMOTE, sTL, sENN, sBorder, sSafe, and sRST, on 44 datasets. It outperforms the other pre-processing methods in terms of $F - measure$ and AUC values, and gives the lowest over-sampling rate. Data complexity measures are also investigated to show that FRB+CHC is more robust to data complexity than the other methods.

The third pre-processing method is called uFRB+CHC, which is presented in Chapter 6. This proposed method is different from our previous methods. It is an under-sampling

method over large imbalanced datasets. The samples of the majority are selected based on a fuzzy rule base and CHC is then applied to further decrease the data size. The proposed method is compared to different under-sampling methods, including RUS, CNN, TL, OSS, and NCL on a large dataset called Census. It outperforms the other methods in terms of $F - measure$ and AUC values.

A general comparison among these three proposed methods is given in Chapter 7. Triangular and Gaussian membership functions are used in FRB+CHC to show its impact on the results. Overall, FRB+CHC with triangular membership function shows the best performance in terms of $F - measure$ and AUC values. The performance becomes worse when the Gaussian membership functions are used instead. The reason is that the synthetic samples are generated densely around some of the original minority samples and cause the over-fitting problem when the Gaussian membership functions are used. Although uFRB+CHC is inferior to the other proposed methods in terms of $F - measure$ and AUC values, its results does not have the over-fitting problem and has the highest under-sampling rate, which is beneficial to large datasets.

Finally, FRB+CHC with triangular membership functions is applied on the datasets of protein-ligand binding sites. The details are presented in Chapter 8. Two testing datasets (198 drug-target complexes and 210 bound structure) are used to illustrate the improvement brought by different pre-processing methods. Our proposed method shows a significant improvement in both datasets. Then, the success rates of the two datasets are added together, which is used to compare the proposed method with MetaPocket, LIGSITE^{CSC}, PASS, Q-SiteFinder, and SURFNET. Overall, our method can locate the binding sites successfully for a larger number of proteins than the other prediction methods.

9.2 Future Works

In this thesis, the protein-ligand binding sites prediction is done by a SVM classifier. The binding sites are represented by geometric grids. Contacting protein residues to the ligands is also a common representations of the binding sites. We might further improve the accuracy and decrease the false positive rate by using the "protein-family approach" to train the SVM classification model. It groups the similar residues of the proteins, based on the proteins' sequence, and trains the classification model of each group. This approach can also be used in the docking phase to determine the scoring function. Docking phase is the next step of structure-based drug design after the protein-ligand binding site is located, and the scoring function is used to rank the best poses of ligands.

References

[1] I. Kola and J. Landis, “Can the pharmaceutical industry reduce attrition rates?” *Nature Reviews Drug Discovery*, vol. 3, pp. 711–716, 2004.

[2] M. Dickson and J. Gagnon, “Key factors in the rising cost of new drug discovery and development,” *Nature Reviews Drug Discovery*, vol. 3, pp. 417–429, 2004.

[3] K. Qu and N. Brooijmans, “Structure-based drug design,” in *Computational Methods for Protein Structure Prediction and Modeling*, Y. Xu, D. Xu, and J. Liang, Eds. Springer New York, 2007, pp. 135–176.

[4] A. Laurie and R. Jackson, “Methods for the prediction of protein-ligand binding sites for structure-based drug design and virtual ligand screening,” *Current Protein and Peptide Science*, vol. 7, no. 5, pp. 395–406, Oct. 2006.

[5] I. Kuntz, “Structure-based strategies for drug design and discovery,” *Science*, vol. 257, pp. 1078–1082, 1992.

[6] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne, “The protein data bank,” *Nucleic Acids Research*, vol. 28, no. 1, pp. 235–242, 2000.

[7] K. Henrick and J. Thornton, “PQS: a protein quaternary structure file server,” *Trends in Biochemical Sciences*, vol. 23, no. 9, pp. 358–361, Sept. 1998.

[8] S. Kalyaanamoorthy and Y. Chen, “Structure-based drug design to augment hit discovery,” *Drug Discovery Today*, vol. 16, no. 17–18, pp. 831–839, 2011.

[9] A. Wlodawer and J. Vondrasek, “Inhibitors of HIV-1 protease: A major success

of structure-assisted drug design," *Annual Review of Biophysics and Biomolecular Structure*, vol. 27, pp. 249–284, 1998.

[10] M. Kastenholz, M. Pastor, G. Cruciani, E. Haaksma, and T. Fox, "GRID/CPCA: a new computational tool to design selective ligands," *Journal of Medicinal Chemistry*, vol. 43, no. 16, pp. 3033–3044, 2000.

[11] S. Schmitt, M. Kuhn, and G. Klebe, "A new method to detect related function among proteins independent of sequence and fold homology," *Journal of Molecular Biology*, vol. 323, no. 2, pp. 387–406, 2002.

[12] K. Kinoshita and H. Nakamura, "Identification of the ligand binding sites on the molecular surface of proteins," *Protein Science*, vol. 14, no. 3, pp. 711–718, 2005.

[13] M. Hendlich, F. Rippmann, and G. Barnickel, "LIGSITE: automatic and efficient detection of potential small molecule-binding sites in proteins," *Journal of Molecular Graphics and Modelling*, vol. 15, no. 6, pp. 359–363, 1997.

[14] J. An, M. Totrov, and R. Abagyan, "Pocketome via comprehensive identification and classification of ligand binding envelopes," *Molecular and Cellular Proteomics*, vol. 4, pp. 752–761, 2005.

[15] J. Capra, R. Laskowski, J. Thornton, M. Singh, and T. Funkhouser, "Predicting protein ligand binding sites by combining evolutionary sequence conservation and 3D structure," *PLoS Computational Biology*, vol. 5, no. 12, 2009. [Online]. Available: http://compbio.cs.princeton.edu/concavity

[16] A. Ben-Hur, C. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, "Support vector machines and kernels for computational biology," *PLoS Computational Biology*, vol. 4, no. 10, 2008.

[17] H. He and E. García, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[18] M. Kubat, R. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning*, vol. 30, no. 2–3, pp. 195–215, 1998.

[19] K. Ezawa, M. Singh, and S. Norton, "Learning goal oriented bayesian networks for telecommunications risk management," in *Processdings of the International Conference on Machine Learning*. Bari, Italy: Morgan Kauffmann, 1996, pp. 139–147.

[20] L. Xu, M. Chow, and L. Taylor, "Power distribution fault cause identification with imbalanced data using the data mining-based fuzzy classification e-algorithm," *IEEE Transactions on Power Systems*, vol. 22, no. 1, pp. 164–171, 2007.

[21] G. Batista, R. Prati, and M. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explorations*, vol. 6, no. 1, pp. 20–29, 2004.

[22] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal Of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[23] H. Guo and H. Viktor, "Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach," *SIGKDD Explorations*, vol. 6, no. 1, pp. 30–39, 2004.

[24] G. Weiss and F. Provost, "Learning when training data are costly: the effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.

[25] F. Provost and T. Fawcett, “Robust classification for imprecise environments,” *Machine Learning*, vol. 42, no. 3, pp. 203–231, 2001.

[26] Y. Lin, Y. Lee, and G. Wahba, “Support vector machines for classification in non-standard situations,” *Machine Learning*, vol. 46, no. 1–3, pp. 191–202, 2002.

[27] B. Raskutti and A. Kowalczyk, “Extreme rebalanceing for SVMs: a case study,” *SIGKDD Explorations*, vol. 6, no. 1, pp. 60–69, 2004.

[28] A. Fernández, S. García, M. Jesus, and F. Herrera, “A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets,” *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2378–2398, 2008.

[29] F. Provost, “Machine learning from imbalanced data sets 101,” in *Proceedings of the AAAI’2000 Workshop on Imbalanced Data Sets*, 2000, pp. 1–3.

[30] B. Boser, I. Guyon, and V. Vapnik, “An training algorithm for optimal margin classifiers,” in *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[31] V. Vapnik, *The nature of statistical learning theory*, 2nd ed. Springer, 1999.

[32] B. Schölkopf and A. Smola, *Learning with kernels*. Cambridge (Massachusetts): MIT Press, 2002.

[33] K. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, “An introduction to kernel-based learning algorithms,” *IEEE Transactions on Neural Networks*, vol. 12, pp. 181–201, 2001.

[34] B. Schölkopf, K. Tsuda, and J. Vert, *Kernel methods in computational biology*. Cambridge (Massachusetts): MIT Press, 2004.

[35] J. Vert, "Kernel methods in genomics and computational biology," in *Kernel Methods in Bioengineering, Signal and Image Processing*, G. Camps-Valls, J. Rojo-Álvarez, and M. M.-R. M, Eds. Idea Group, 2007, ch. 2, pp. 42–63.

[36] B. Dessailly, M. Lensink, C. Orengo, and S. Wodak, "LigASite: a database of biologically relevant binding sites in proteins with known apo-structures," *Nucleic Acids Research*, vol. 36, pp. 667–673, 2008.

[37] R. Laskowski, "SURFNET: a program for visualizing molecular surfaces, cavities and intermolecular interactions," *Journal of Molecular Graphics*, vol. 13, pp. 323–330, 1995.

[38] Z. Zhang, Y. Li, B. Lin, M. Schroeder, and B. Huang, "Identification of cavities on protein surface using multiple computational approaches for drug binding site prediction," *Bioinformatics*, vol. 27, no. 15, pp. 2083–2088, 2011.

[39] B. Huang and M. Schroeder, "LIGSITEcsc: predicting ligand binding sites using the connolly surface and degree of conservation," *BMC Structural Biology*, vol. 6, no. 1, p. 19, 2006.

[40] V. Guilloux, P. Schmidtke, and P. Tuffery, "Fpocket: An open source platform for ligand pocket detection," *BMC Bioinformatics*, vol. 10, no. 1, p. 168, 2009.

[41] A. Laurie and R. Jackson, "Q-SiteFinder: an energy-based method for the prediction of protein-ligand binding sites," *Bioinformatics*, vol. 21, pp. 1908–1916, 2005.

[42] L. Eshelman, "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination," in *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 265–283.

[43] J. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561–575, 2003.

[44] G. Brady and P. Stouten, "Fast prediction and visualization of protein binding pockets with PASS," *Journal of Computer-Aided Molecular Design*, vol. 14, pp. 383–401, 2000.

[45] D. Levitt and L. Banaszak, "POCKET: a computer graphics method for identifying and displaying protein cavities and their surrounding amino acids," *Journal of Molecular Graphics*, vol. 10, pp. 229–234, 1992.

[46] T. Binkowski, S. Naghibzadeh, and J. Liang, "CASTp: computed atlas of surface topography of proteins," *Nucleic Acids Research*, vol. 31, pp. 3352–3355, 2000.

[47] H. Edelsbrunner and E. Mucke, "Three-dimensional alpha shapes," *ACM Transactions on Graphics*, vol. 13, pp. 43–72, 1994.

[48] H. Edelsbrunner, M. Facello, P. Fu, and J. Liang, "Measuring proteins and voids in proteins," in *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, vol. 5. IEEE Computer Society Press, 1995, pp. 256–264.

[49] H. Edelsbrunner, M. Facello, and J. Liang, "On the definition and the construction of pockets in macromolecules," *Discrete Applied Mathematics*, vol. 88, pp. 83–102, 1998.

[50] W. Valdar, "Scoring residue conservation," *Proteins: Structure, Function, and Genetics*, vol. 48, no. 227–241, 2002.

[51] K. Wang and R. Samudrala, "Incorporating background frequency improves

entropy-based residue conservation measures," *BMC Bioinformatics*, vol. 7, no. 1, p. 385, 2006.

[52] J. Capra and M. Singh, "Predicting functionally important residues from sequence conservation," *Bioinformatics*, vol. 23, pp. 1875–1882, 2007. [Online]. Available: http://compbio.cs.princeton.edu/conservation

[53] S. Liang, C. Zhang, S. Liu, and Y. Zhou, "Protein binding site prediction using and empirical scoring function," *Nucleic Acids Research*, vol. 34, pp. 3698–3707, 2006.

[54] T. Magliery and L. Regan, "Sequence variation in ligand binding sites in proteins," *BMC Bioinformatics*, vol. 6, no. 1, p. 240, 2005.

[55] M. Connolly, "Analytical molecular surface calculation," *Journal of Applied Crystallography*, vol. 16, pp. 548–558, 1983.

[56] B. Huang, "MetaPocket: a meta approach to improve protein ligand binding site prediction," *Journal of Integrative Biology*, vol. 13, no. 4, pp. 325–330, 2009.

[57] T. Kawabata, "Detection of multi-scale pockets on protein surfaces using mathematical morphology," *Proteins*, vol. 78, pp. 1195–1121, 2010.

[58] J. Yu, Y. Zhou, I. Tanaka, and M. Yao, "Roll: a new algorithm for the detection of protein pockets and cavities with a rolling probe sphere," *Bioinformatics*, vol. 26, pp. 46–52, 2010.

[59] Q. Gu, Z. Cai, L. Zhu, and B. Huang, "Data mining on imbalanced data sets," in *International Conference on Advanced Computer Theory and Engineering*, 2008, pp. 1020–1024.

[60] P. Hart, “The condensed nearest neighbor rule,” *IEEE Transactions on Information Theory*, vol. 14, pp. 515–516, 1968.

[61] I. Tomek, “Two modifications of CNN,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, pp. 769–772, 1976.

[62] M. Kubat and S. Matwin, “Addressing the curse of imbalanced training sets: one-sided selection,” in *Proceedings of the 14th International Conference on Machine Learning*, 1997, pp. 179–186.

[63] J. Laurikkala, “Improving identification of difficult small classes by balancing class distribution,” in *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, ser. AIME ’01. London, UK, UK: Springer-Verlag, 2001, pp. 63–66.

[64] D. Wilson, “Asymptotic properties of nearest neighbor rules using edited data,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 2, no. 3, pp. 408–421, July 1972.

[65] S. García and F. Herrera, “Evolutionary undersmpling for classification with imbalanced datasets: proposals and taxonomy,” *Evolutionary Computation*, vol. 17, no. 3, pp. 275–306, 2009.

[66] S. García, J. Derrac, I. Triguero, C. Carmona, and F. Herrera, “Evolutionary-based selection of generalized instances for imbalanced classification,” *Knowledge-Based Systems*, vol. 25, no. 1, pp. 3–12, 2012.

[67] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, “SMOTE-RSB*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory,” *Knowledge and Information Systems*, pp. 1–21, 2011.

[68] H. Han, W. Wang, and B. Mao, “Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning,” in *International Conference on Intelligent Computing (ICIC05)*. Springer, 2005, pp. 878–887.

[69] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, “Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem,” in *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, ser. PAKDD ’09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 475–482.

[70] A. Gutteridge, G. Bartlett, and J. Thornton, “Using a neural network and spatial clustering to predict the location of active sites in enzymes,” *Journal of Molecular Biology*, vol. 330, pp. 719–734, 2003.

[71] A. Koike and T. Takagi, “Prediction of protein-protein interaction sites using support vector machines,” *Protein Engineering, Design and Selection*, vol. 17, no. 2, pp. 165–173, 2004.

[72] M. Keil, T. Exner, and J. Brickmann, “Pattern recognition strategies for molecular surfaces: binding site prediction with a neural network,” *Journal of Computational Chrmistry*, vol. 25, no. 6, pp. 779–789, 2004.

[73] J. Bradford and D. Westhead, “Improved prediction of protein-protein binding sites using a support vector machines approach,” *Bioinformatics*, vol. 21, no. 8, pp. 1487–1494, 2005.

[74] N. Petrova and C. Wu, “Prediction of catalytic residues using support vector machines with selected protein sequence and structural properties,” *BMC Bioinformatics*, vol. 7, no. 1, p. 312, 2006.

[75] T. Joachims, “Making large-scale SVM learning practical,” in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT Press, 1999, ch. 11, pp. 169–184. [Online]. Available: http://svmlight.joachims.org/

[76] T. Joachims, *Learning to classify text using Support Vector Machines: methods, theory, and algorithms*. Kluwer Academic Publishers, 2002.

[77] A. Asuncion and D. Newman. (2007) UCI machine learning repository. School of Information and Computer Sciences. University of California, Irvine. [Online]. Available: http://www.ics.uci.edu/ mlearn/MLRepository.html

[78] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[79] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, “KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework,” *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2–3, pp. 255–287, 2011. [Online]. Available: http://www.keel.es/

[80] C. Chang and C. Lin, “LIBSVM: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 27, pp. 1–27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

[81] J. Luengo, A. Fernández, S. García, and F. Herrera, “Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling,” *Soft Computing*, vol. 15, pp. 1909–1936, 2011.

[82] A. Hoekstra and R. Duin, “On the nonlinearity of pattern classifiers,” in *Proceedings of the International Conference on Pattern Recognition (ICPR ’96)*, 1996, pp. 271–275.

[83] T. Ho and M. Basu, “Complexity measures of supervised classification problems,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 298–300, 2002.

[84] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.