Machine Learning for Mining Imbalanced Data

Sabera Hoque

Department of Computer Science and Engineering

United International University

A thesis submitted for the degree of

MSc in Computer Science & Engineering

February 2018

Declaration

I, Sabera Hoque, declare that this thesis titled, "Machine Learning for Mining Imbalanced Data" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a MSc degree at United International University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Sabera Hoque

Certificate

I do hereby declare that the research works embodied in this thesis entitled "Machine Learning for Mining Imbalanced Data" is the outcome of an original work carried out by Sabera Hoque under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of MSc in Computer Science and Engineering.

Signed: _____

Date: _____

Dr. Dewan Md. Farid,

Associate Professor,

Department of Computer Science and Engineering,

United International University,

Dhaka-1209, Bangladesh.

Abstract

Mining imbalanced data, which is also known as class imbalanced problem is one of the most enormous challenging tasks in machine learning for data mining applications. To achieve overall accurate performance in imbalanced classification employing machine learning techniques is difficult as the majority class instances always overpower the minority class instances by a huge difference. An unequal distribution is very common in real world high dimensional datasets, where binary classification is more frequent than the multi-class classification task. Most existing machine learning algorithms are more focused on classifying majority class instances while ignoring or misclassifying minority class instances. Several techniques have been introduced in the last decades for imbalanced data classification, where each of this technique has their own advantages and disadvantages. In this paper, we have study and compared 12 extensively imbalanced data classification methods: SMOTE, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade, OverBagging, UnderBagging, SMOTEBagging to extract their characteristics and performance on 22 imbalanced datasets.

My most humble and sincere thanks to my extraordinary husband for his love and patience with the thesis completion process and also for his super senses and sincerity as a co-worker. To my sweetest sun whose unconditional love inspired me to improve myself.

Published Papers

Work relating to the research presented in this thesis has been published by the author in the following conferences:

1. Md Yasir Arafat, Sabera Hoque, and Dewan Md. Farid, "Cluster-based Under-sampling with Random Forest for Multi-Class Imbalanced Classification," 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), December 6-8, 2017, Colombo, Sri Lanka, pp. 1-6.
2. Sabera Hoque, Md. Yasir Arafat, and Dewan Md. Farid, "Machine Learning for Mining Imbalanced Data," International Conference on Emerging Technology in Data Mining and Information Security (IEMIS 2018), February 23-25, 2018, Kolkata, India (Accepted).

Acknowledgements

I would like to express profound gratitude to my honorable supervisor Dr. Dewan Md. Farid Sir, whose assiduous supervision support me to accomplish my thesis work successfully and timely. His continual directions, kind encouraging nature and motivational speech guided me for achieving this research outcome and hold me on the right track, as a consequence I have successfully accomplished my thesis work.

I would like to thank to Dr. Chowdhury Mofizur Rahman(Professor and Vice Chancellor (In-Charge), Interim Dean) whose Google scholars profile inspired me on Data mining,

I remarkably grateful to Professor Dr. Nurul Huda Sir, our honorable course coordinator, for his continuous valuable guidance throughout my entire program.

I am thankful to Dr. Hasan Sarwar (Professor), Dr Salekul Islam (Associate Professor and Head of the Dept.), Dr. Khondoker Abdullah Al Mamun(Associate Professor and Director - AIMS Lab), Dr. Swakkhar Shatabda(Associate Professor and Undergraduate Program Coordinator) and some other teachers and faculty members who are effortlessly working to enlightening our Computer Science and Engineering department.

I must thank our external reviewer Professor Dr. Sayed Akhter Hussain for his valuable time on my thesis work.

I am thankful to United International University (UIU) for giving me such a pleasant academic atmosphere.

Last but not least, I want to appreciate to my family, particularly my husband and my son for their extraordinary enthusiastic support.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Imbalanced Data Classification

Imbalanced data classification is an important problem in supervised learning, where the classes are not represented fairly i.e one class is underwhelmed called as minority class while the other class is overwhelmed called as majority class. In many real-world applications imbalanced datasets are very common such as medical diagnosis, credit card fraud detection, anomaly detection, managing risk and predicting failures of technical equipment, defected product on manufacturing. For example, consider a dataset that consists of 100,000 patient records of a hospital, where a list of candidates who tested for diagnosis cancer. It was found in the dataset that 98% patients did not have cancer and only 2% got unlucky. Also, in credit card fraud detection, fraudulent transactions will occur very rare than regular transactions. Imbalanced datasets often considers imbalanced to mean a minority class of 10% to 20%. In reality, datasets can get far more imbalanced than this like about 2% of credit card accounts are defrauded per year. Medical screening for a situation is generally performed on an enormous community without the condition, to detect a finite minority with it (e.g., HIV prevalence in the USA is 0.4%).

The conventional machine learning and data mining algorithms classify the majority class instances more accurately, rather failed to classify the minority class instances [1]. To resolve this problem various methods have been proposed in the last decade such as sampling techniques, cost-sensitive methods, and ensemble learning methods [2]. The most popular sampling techniques are under-sampling and over-sampling. The under-

sampling method randomly deducts the majority class instances from the dataset, while the over-sampling method artificially generated and added minority class instances to the dataset in order to balance the imbalanced dataset like SMOTE (Synthetic Minority class Over-Sampling Technique) [3]. In under-sampling method some significant instances may eliminate and an overfitting of data may cause in over-sampling method [4]. On the other hand, classification results are most often unstable in cost-sensitive learning methods as it is very difficult to get the accurate misclassification cost. Also, ensemble classifier increases the classification accuracy of a single classifier, but neither the ensemble learning solves the class imbalance problem. The ensemble methods use sampling techniques to obtain balanced data in each iteration, as a result they might suffer from overfitting or drop some potential information [5]. To deal with imbalanced datasets ensemble methods need to be designed specifically.

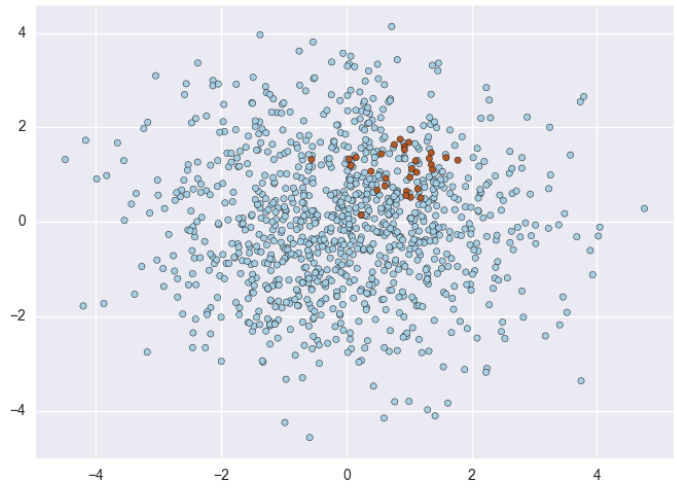

Figure 1.1: An example of the imbalanced data.

1.2 Mining Complex Data

Real-world data are traditionally imperfect, noisy, carry errors, missing feature values and often inappropriate. But if the data quality is not strong enough then it will not produce first-class mining results. To make these data error-free and produce the superior quality result we need to used mining process to mine these complex data.

Data mining is a well-known method of discovering hidden knowledge in data. The two most extensively used data mining methods are - i) classification: This is a supervised learning technique where the classes are predefined and ii) clustering: an unsupervised learning technique whose job is to cluster the data into groups of similar objects [6].

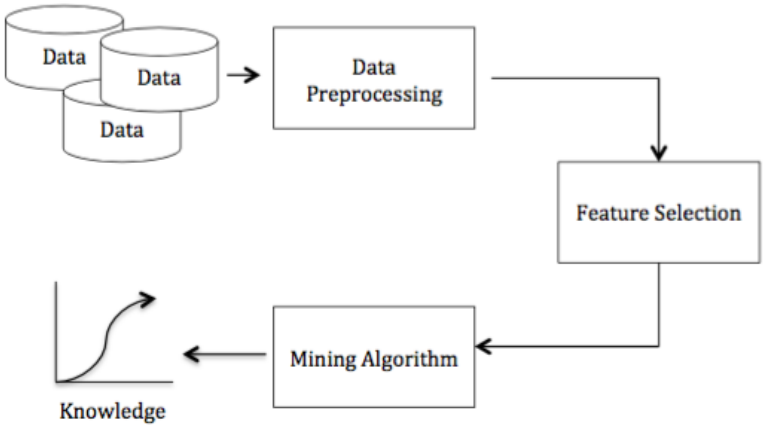

Figure 1.2: Process of Mining Complex Data.

1.3 Motivation

The algorithms of machine learning can be performed best only when the total instances present in each class is almost in the same quantity. The problems arise when the instances of each class become imbalanced i.e one class overpower other. Binary class faced this complication more frequently rather than the multi-class dataset. In plain words, a data set that consists a different ordination among its classes is considered to be imbalanced [5]. For example, let's consider a dataset consist of 100,000 observations of the cancer patient in a hospital. Here the dependent variable case represents if a patient has been invaded or not. After analyzing the data, it was found 98% did not get invaded and only 2% got unlucky.which is a perfect imbalanced condition. Also in credit card fraud detection, fraudulent transactions will occur very rare than regular transactions. In the following, we tried to show some examples of imbalanced data whom we can depict as a "needle in a haystack problems" as well because machine learning classifiers are used to find a tiny amount of negative classes from a vast number of positive classes.

- Every year only 2% of credit card accounts are cheated.

- Medical screening for a situation is generally performed on an enormous dataset with limited minority.(e.g., In the USA the total AIDS outbreak is only 0.4%).
- On an average, around 1% disk drive fails to function properly per year.
- Factory production defect rates about 0.1 and many more real world examples.

The above examples prove that mining imbalanced data is such a challenging job and there are still various difficulties remain in the enormous field of imbalanced learning which caught our attention for research and intensive development. Yet various unexamined direction to be taken in this branch of machine learning, which makes it up-to-date and thrilling. Since a number of classifiers have already been revealed to solve the mining issue we tried to make a review to make a decision or to chose one of the best classifiers to defeat the data mining challenge.

1.4 Objectives of the Thesis

One of the common real-world examples is, in the medical sectors incorrect prediction of uncommon but crucial disease which may cause a life-threatening situation. Resembling cases are noticed in several sectors such as fraudulent in credit card or other banking transaction, detecting network trespassing, collapsing of technical apparatus, defected product on manufacturing. There are many more real-life situations which produce imbalanced dataset as an output. Distinct approaches are applied by the researchers for decades for classification of the imbalanced dataset. This work will perform an empirical comparison amongst some extensively accepted algorithm of class imbalance problem using various existing data set to extract the characteristics and summarizes the most significant one for the novel dataset. Apart from that, though ensemble classifiers are well established for their high accuracy, no single learning technique capable of solving the class imbalance problem solely. That's why ensemble learning algorithms are sometimes designed specifically for an individual problem. In our thesis work the core objective is to review the state of the art techniques for imbalanced data classification using 22 imbalanced data-sets to reveal the best one for this data set.

1.5 Thesis Contributions

In this thesis we, We study various ensemble learning method for handling class imbalance problem and review 12 state of the art techniques: SMOTE, OverBagging, UnderBagging, SmoteBagging, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade for mining imbalanced data and compare their performance on 22 imbalanced datasets.

1.6 Organization of the Thesis

The entire thesis is constructed as :

Chapter 2 Briefly describes the data balancing methods with examples.

Chapter 3 We presents the data balancing methods : SMOTE, OverBagging, UnderBagging, SmoteBagging, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade.

Chapter 4 Presents datasets and experimental results.

Chapter 5 Presents the conclusion and future works.

Chapter 2

Data Balancing Methods

2.1 Sampling Methods

Sampling data methods solve the imbalance problems either by re-arranging or creating synthetic instances in the class. It can be attained by two procedures such as under-sampling and over-sampling or sometimes by uniting over and under-sampling methods as a hybrid approach [5]. Sampling methods are divided into different forms such as - (1) Under-sampling and (2) Over-sampling. Diverse intellectual algorithms have been proposed [7], like SMOTE, borderline SMOTE, and Wilson's editing [8]. Hulse [9] observe the performance of diverse data sampling methods (around seven) using learning algorithms including investigational data sets, by discovering that the two most successful sampling algorithm is RUS and SMOTE.

2.1.1 Over-sampling Methods

The Over-sampling method works with minority class instances that require no information loss. It is divided into two ways: (1) Random Over-sampling, and (2) Informative Over-sampling. Random over-sampling methods stable the datasets with randomly over-sampling of the inferior class instances. Informative over-sampling synthetically generates and adds the minority class instances into the dataset. The disadvantage of this method is overfitting as it adds replicated data from the original dataset.

SMOTE : Synthetic Minority Over-sampling Technique

This sampling method combined under-sampling of the majority class with a notable form of over-sampling of the minority class [3]. The concept of the method is

developed by various investigational dataset, C4.5 as the base classifier, and Naive Bayes classifier including the AUROC curve [10]. This is an over-sampling approach where not only the instances are over-sampled, the minority class builds some synthetic samples also. One of the well-established examples of this approach is the identification of optical character [11]. They produced additional training data by performing particular operations on real data. To perturb the training data they have used operations like rotation and skew. Assume that, the total adjacent instances of minority class is K. According to the quantities of total instances needed, adjacent instances among the K (where k=5) instances are selected at random. For example, if the required over-sampling amount is 400%, then only 4 are selected from the 5 adjacent instances and 1 instances are produced in each direction. Synthetic instances are created by taking the difference between the adjacent instances and featured instances. Now assume 2 random number between 0 and 1 and multiply this difference by the random number. Then add the result value to the feature vector. The minority class to become more general when this method successfully forces the decision region. SMOTE algorithm the latest approach to over-sampling technique enhance the appropriateness of a classifier for a minority class. The combination of SMOTE and under-sampling method performs finer than ordinary under-sampling technique. The novel approaches SMOTE was examined using different datasets, with varying levels of imbalance. Merging SMOTE and under-sampling technique also functions better, based on possession in the ROC space. As SMOTE makes the setting territory larger and unspecified, which makes the classifier very much unique than that of other classifiers.

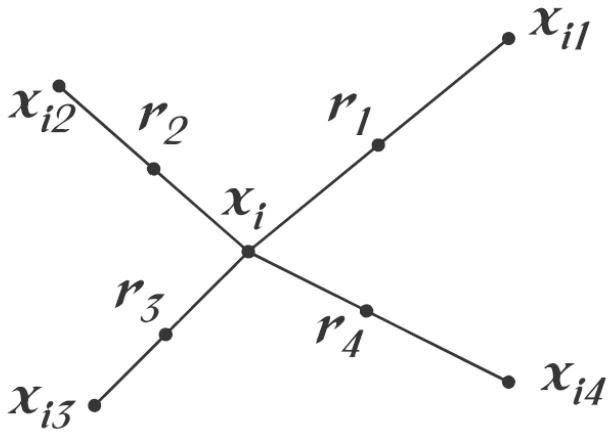

Figure 2.1: SMOTE : Synthetic Minority Over-sampling Technique

ADASYN : Adaptive Synthetic Sampling Approach for Imbalanced Learning

Inspired by the favorable outcome of some popular concepts of using artificial instances like SMOTE [3], SMOTEBoost [3], and DataBoost [12], this theory proposes an adaptive method called ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning). The goal of this topic is mainly divided into 2 types one of them is to alleviate the warp and adaptively learning. ADASYN can adaptively produce synthetic data instances similar to original data distribution for the minority class, which make the imbalanced data less biased. To be more focused on improving learning performance ADASYN individually shift the classifier decision boundary. This will ease to learn the difficult examples. To perform this function the classifier will separate the objectives into two ways i.e a gradual coordination of weights and an adaptive training method following to data patterns. The method used a weighted ordination of minority class samples following to their stiffness in learning. For adding up this weight more synthetic data is propagated from minority class instances, that are difficult to acquired as compared to those minority instances that are simpler to adopt. As a consequence, the ADASYN method balanced learning in two ways: (1) reduced the biases to the minority class of imbalanced data, and (2) attempered to changed the classification decision edge by approaching the complex instances.

Borderline-SMOTE-I : To attained excellent prediction, the borderline instances of the minority class are over-sampled only. This sampling concept is separate from the existing over-sampling ideas where the negligible instances or the arbitrary subset of the minority class are over-sampled [13] [14] [3]. The foundation of this method is the popular algorithm SMOTE [3] and it functions almost same as SMOTE except for the instances taken from borderline of the cluster to generates artificial negligible instances to oversample the minority class. As we know, inside SMOTE algorithm, k numbered closest instances of the similar class are calculated, where k=5. After that, some samples are arbitrarily chosen at the over-sampling rate. The novel synthetic instances are propagated than towards the line segment and chosen nearest neighbor among the minority sample. This is the idea that fortifies the borderline minority instances by over-sample which differ from the remaining over-sampling methods. Here firstly, the borderline instances are selected. After that, the synthetic instances are

generated from the selected ones and eventually affixed to the main training dataset [8].

2.1.2 Under-sampling Methods

The under-sampling method works with majority class instances, which removes instances randomly from majority class to make the dataset balanced. It is best for use when the dataset is big. Under-sampling methods are of two types: (1) Random under-sampling, and (2) Informative under-sampling. Random under-sampling method randomly chooses instances from the majority class, which later eliminated until the dataset gets balanced. Removing instances randomly may cause the training data to lose important information. Informative under-sampling uses a pre-specified selection criterion to remove the majority class instances.

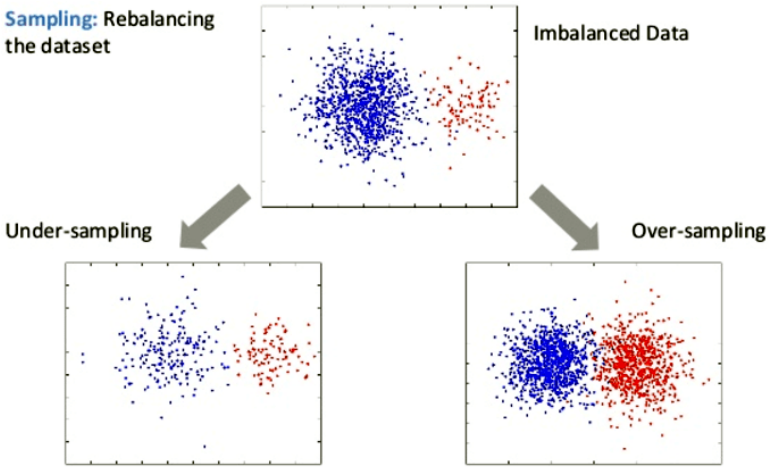

Figure 2.2: The example of Data Balancing method

RUS-I : Random Under-sampling

Usually, over-sampling methods give further exact output than under-sampling methods following to the AUROC curve. The imbalance problem is not only the result of the class imbalance, also happen due to the data overlapping in the classes. The main inspiration behind the scene is balanced the training data and eliminate noisy instances as well. Elimination of noisy instances could help in discovering well-defined clusters which will help to create easier and error-free potential models. The proposed

method [15] used the Laplace correction method for standardizing the node accuracy in order to draw AUROC for evaluating the experiments potentiality. The core function of random over-sampling is to stabling class ordination by replicating the minority class instances arbitrarily. Random over-sampling may cause the possibility of the situation of overfitting since it replicates exactly the same of the original minority class instances. In this way, an emblem classifier, might create new rules that are seemingly perfect, though cover one repeated example indeed [16]. As opposed to, the leading handicap of random under-sampling is that this method can abandon the possibly necessary data that could be significant for the procedure.

NCL-I : Neighborhood Cleaning Rule

Neighborhood Cleaning Rule (NCL) is better in simple random and one-sided selection methods. Though in each diminishing method, the negligible classes i.e around (20-30%) enhance their identification, the distinctions were really very unemphatic. Diversity, accuracies, TP(true-positive) rates and TN (true-negative) rates obtained with the K (where k=3) nearest neighbor method and C4.5 used as a base classifier from the minimized data. The final output proves that NCL is a potential approach that helps improve the model quality when working with the tough negligible classes and constructing classifiers to recognize these tiny classes form the real-life imbalanced data [17].

2.1.3 Cost-sensitive Learning Methods

The cost-sensitive learning (CSL) takes the misclassification costs into account in the learning process to minimize the total cost. It is also a common approach to settle the class imbalanced problem [18]. The objective of the cost-sensitive learning is to achieve high propriety of classifying minority class instances. However, it is difficult to find the misclassification cost in different iterations as it depends on the different types of errors.

The cost-sensitive learning method evaluates the misclassification costs of data in any class to different class and after that minimise the total costs instead of total errors. Diverse costs produce several types of error in various practical classification problems. Breiman et al., [19] disclosed a couple of approaches related to including variable misclassification costs into the tree fetching process. These two methods accommodate the test data election standard in the tree progress procedure and the classic covetous

divide-and-conquer procedure for inducing trees. A pruning process is applied to fudge overfitting which abates the volume of the tree such that the calculated error is a minimum. In brief, the typical tree fetching procedure looks for producing the least error tree [18]. Several costs related to multiple types of error in various real-life classification problems. If an errors occurred in analyzing someone as healthy when they have a life-threatening disease is generally deliberated to be far more serious and thus, higher cost) than the opposite type of error diagnosing someone as diseased when they are in fact normal and healthy condition. For instance, a dataset of cancer invaded people is given. We have to find out if a person is invaded or not. There is no cost associated with identifying a person with cancer as positive and a person without negative. But, the cost associated with identifying a person with cancer cell as negative (False Negative) is much more dangerous than identifying a person without cancer cell as positive (False Positive).

Cost-sensitive Tree

According to the researchers, the cost-sensitive trees not always consist of minor misclassification costs, sometimes conferred with unseen test data than those trees induced without cost discretion. The greedy divide-and-conquer algorithm is applied on cost-sensitive tree induction. Negative empirical results produced when using one of Breiman constructions to persuaded with the cost-sensitive trees [20]. By employing a post-processing technique, Webb [21] conveyed that using a cost-sensitive exceptional technique to a tree with least error can lessen its misclassification costs by about 3% on average. On the other hand, when using the greedy divide-and-conquer algorithm on the cost-sensitive tree it is possible to learn effectively straight from the training data. This concept is motivated by Quinlan's popular concept of instance weight modification in boosting decision trees [10].

Cost-sensitive Neural Network

We can see the consequences of sampling into training cost-sensitive neural networks. When dealing with novel instances the classifiers normally tried to alleviate the total errors [22]. This condition is tenable if the costs of multiple errors are identical level, whereas in most of the real-world applications the costs of diverse errors are different. For example, in the medical sector, if a patient who is in a good health condition diagnosed as an invaded person then this will be not a high cost. On the other hand, if an invaded patient mistakenly diagnosing as a healthy this will be much larger cost

as because this may occur in the destruction of a life. Cost-sensitive neural networks is not a very common topic and applying cost-sensitive decision tree learning methods directly to neural networks is not possible. In the example-hefting method, the research algorithm required weighted samples which is very tough for regular neural network method but not a great issue for C4.5 decision trees [18]. Maloof [23] proposed that when costs are dissimilar and uncertain can be managed in the same way. According to Weiss [14], when solving the class imbalance problem cost-sensitive learning proved as a good solution.

Cost-sensitive Support Vector Machine

The novel algorithm Support Vector Machine (SVM) have been effectively used to lots of real-life complications like digit and handwritten character recognition, speaker identification and face detection, fingerprint and iris detection, Text and hypertext categorization, Classification of images, Bioinformatics and so on. The main advantages of this method are, it has displayed an extraordinary resistance to overfitting a feature explained by the fact the direct implementation of the principle of structural risk minimization. The fundamental concept is to inaugurate different functions for positive and negative points, which transform a larger multiplier for the class where the misclassification cost is weighty [24]. Here, the authors proposed a unique idea where SVM organizes the structural-risk-minimization concepts to lessen a higher bound on the expected risk [25], [26]. On the whole, the structural risk is a logical agreement between the training difficulty and the modeling obstacle. The SVM modeling algorithm which has a superior generalization potentiality functions by forming a different plan with the maximal margin. SVM is rigorously more correct classifier as compared to other conventional classifiers. It can be responsive to high-dimensional imbalance dataset as well.

2.1.4 Ensemble Classifiers

Ensemble learning is the procedure of combining multiple classifiers to form a strong single classifier to classify new instances with high prediction accuracy, such as RandomForest, Bagging, and Boosting. While constructing ensemble methods the most challenging task to be faced by the classifiers are: (1) the coalescence of the classifiers to be used, (2) the base classifiers used for ensemble must be naive to overcome overfitting, and (3) the base learners used should be accurate and distinct enough.

Nowadays, though different classification algorithms are attainable, researchers are facing the typical question of choosing the best model for a particular data set as most of the conventional algorithms undergoes with some common issues, such as computational complexity, running through to native minima or overfitting to the data set used for training. One of the most eminently used approaches to overcome these problems is the ensemble learning which is primarily used to improve the performance of a classifier. An ensemble is a combination of multiple classifiers so as to improve the generalization ability and increase the prediction accuracy, mainly focused on two significant techniques: bagging and boosting [27]. Bagging was initially developed by Breiman [19]. In a bagging ensemble, each individual classifier is trained using a different bootstrap of the data set.The most popular bagging ensemble is the random forest [19].The random forest uses decision trees as the individual classifier. Before training the individual decision trees on their bootstraps, a random feature selection algorithm removes all but a small number of the features to increase the speed of training and the disparity between models. Because the random forest tries to maximize the accuracy of its predictions. Boosting was introduced by Schapire [28], as a way to train a series of classifiers on the most difficult samples to predict. In boosting, each classifier is subject to previous one and focuses on the previous one's errors. Examples that are misclassified in earlier classifiers are chosen more often or weighted more heavily. Boosting Classifier iteratively boost the classifier's performance for imbalanced datasets by either revising misclassification cost or amending the distribution ratio of each class. The main advantage of boosting is it reduces bias while bagging end up with the output that reduced variance.

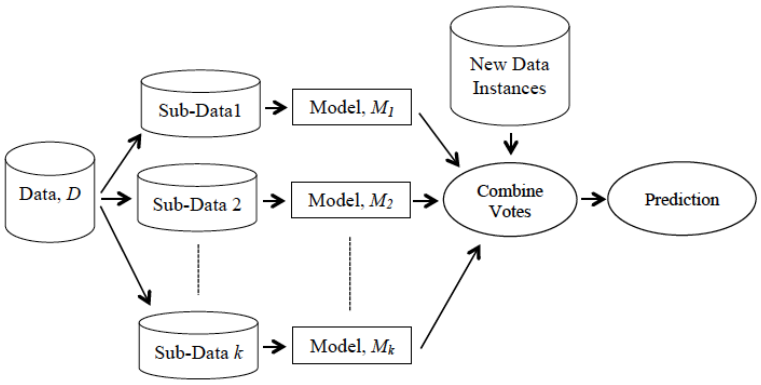

Figure 2.3: The ensemble classifier

Random Forest Random forest (RF) is an ensemble classifier that uses several decision trees with majority voting techniques for classifying test instances. In this method attribute bagging technique is applied for creating sub-data sets from original data set. Attribute bagging also an ensemble learning approach that select attributes randomly from the full space. A random forest is a collection of decision trees where identically disbursed random vectors and each tree cast a vote for the most popular class of input instance. Trees are one of the most favoured and accurate learning methods commonly used for data exploration. For many data sets, it fabricates exact classifier and executed smoothly on extensive databases. Without variable deletion this classifier able to handle enormous input variables. Apart from that, notable enhancement in classification accuracy has resulted from developing an ensemble of trees and allows them to vote for the most popular class. However, with noisy classification/regression tasks, random forests have been detected as overfitted.

Bagging

Bagging also called Bootstrap Aggregating, which merges multiple classifiers into a single prediction model. In order to classify a novel instance, it mostly uses a weighted voting technique. Firstly, the training data is separated into diverse sub-datasets, then classifiers are assigned to an equal weight. From these classifiers, a particular classifier is built using the sub-dataset and then the selection prepared by the classifiers for categorizing the novel samples. This bagged classifier then calculates the votes and after completion, the most voted instances then assign to the novel classifier. Now

averaging of continuous prediction values of each prediction applied by the bagging classifier for a given test instance, and taking the average values.

Boosting Boosting is such an efficient technique that can enhance the performance of any weak classifier. In fact, boosting has been observed as a kind of most significant data sampling technique [29]. Boosting can be performed either by reweighting or resampling [28] process. When boosting accomplished by reweighting, the altered example weights make a straight start along to the base learner through each iteration.

At the time of beginning, all instances are imposed with identical impact. Boosting is the most intelligent ways to process imbalance data sets. Through focusing on these difficult instances at the time of training, boosting engages the establishment of an order of classifiers which destined to classify the tough instances accurately. As Boosting provides the misclassified instances a high weighted value in every iteration, so it alters the ordination of training data successfully. The boosting classifier is proved useful for the minority class classification, due to the innovation of the format of imbalanced data sets.

AdaBoost is the most popular boosting classifier [30]. Chawla et al. [3] proposed SMOTEBoost algorithm which enhances the performance of classifiers. In the SMOTE-Boost algorithm, the SMOTE (Synthetic Minority over-sampling Technique) technique combines with the standard boosting technique. Here the SMOTE is used for enhancing prediction accuracy and boosting is applied for overall accuracy. SMOTE generates data and increase the sampling weights for the minority class during each iteration of boosting.

2.2 Algorithmic Modifications For Class Imbalance

A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets. The coordination between fuzzy logic and genetic algorithms leading to Genetic Fuzzy Systems (GFSs) is one of the most popular methods applied when various computational intelligence techniques are mixed together. Originally GFS is a system based on evolutionary computation, which augmented a learning process using the fuzzy logic. Genetic Programming (GP) [31] is a progress of classical genetic algorithms that enlarge tree-shaped solutions using variable length chromosomes. Though Genetic Programming has been used to learn

fuzzy rule bases [32] gain from its high eloquent power and flexibility. However, the disadvantage is the inflexibility of the concept of a linguistic variable because it loads difficult restrictions on the fuzzy rule structure [33] which may suppose a loss in accuracy when dealing with some complex systems, such as high-dimensional problems, the presence of noise or overlapping classes [34].

2.3 Summary

In this chapter, we tried to depict the various data balancing methods with some appropriate example. We can see that imbalanced dataset can be handled by applying three level of approaches like data level, algorithmic level, and ensemble or hybrid approaches. Data level can be divided into two types sampling method like over-sampling and under-sampling. In the over-sampling method, the minority class instances are duplicated, while majority instances are randomly discarded in order to re-arrange the class distribution of imbalanced data in the under-sampling method. It has been shown that under-sampling may lose some potential useful majority instances while oversampling may cause overfitting of minority instances by making an exact copy. In the algorithmic level approach, machine learning algorithms are modified to accommodate imbalanced data.

Cost-sensitive methods incorporate different misclassification costs for each class in the learning phase.The two most popular ensemble-learning algorithms are boosting and bagging. Bagging stands for bootstrap aggregating and uses the majority voting technique. On the other hand, in boosting bootstrap samples are taken randomly and improving the accuracy of a classifier by a reweighting of misclassified instances.

Chapter 3

Imbalanced Class Classification Methods

In this section, we discuss about the following machine learning algorithms: SMOTE, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, Data-Boost, Easy Ensemble, BalanceCascade, OverBagging, UnderBagging, Smote-Bagging that widely using in many real-world imbalanced data classification problems.

3.1 SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) is an over-sampling technique of the minority class instances to form a balanced dataset. It created synthetic instances minority class instances instead of randomly over-sampling the minority class instances [3]. In general, it performs better than ordinary under-sampling techniques by modifying loss ratio and class priors approach.

The algorithm for smote is shown in algorithm 1:

Algorithm 1 SMOTE

Input: Total minority class instance T; Amount of SMOTE N %; Number of nearest neighbors k

Output: (N/100)* T synthetic minority class instance

Method:

(*If N is less than 100%, randomize the minority class instance as only a random percent of them will be SMOTEd.*);
if $N < 100$ **then**
  Randomize the T minority class instance;
  T = (N/100) * T;
  N = 100;
end if
N = (int)(N/100)(* The amount of SMOTE is assumed to be in integral multiples of 100. *);
k = Total nearest neighbors;
totalAttr = Total attributes;
Sample[ ][ ]: array for main minority class instance;
newindex: record synthetic instance created, started from zero;
Synthetic[ ][ ]: array for synthetic instance; // (* Calculate k nearest neighbors for each minority class sample only. *)
for i = 1 to T **do**
  Calculate k nearest neighbors for i, and save the indices in the nn array;
  Populate(N, i, nnarray);
end for//
Populate(N, i, nnarray) (* Function to generate the synthetic instance. *);
while $N \neq 0$ **do**
  Choose a random number between 1 and k, call it nn.
  This step chooses one of the k nearest neighbors of i.
  for attr = 1 to totalAttr **do**
    $Calculate : dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
    $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
  end for
  $newindex++$
  $N = N - 1$
end while
return
End

3.2 Adaboost : Adaptive Boosting

AdaBoost is the most popular ensemble classifier, which basically improves the performance of weak learning [30, 35]. During its, each iteration, instances weights are changed according to how they are classified and focus on misclassified instances in the next iteration. AdaBoost uses the weighted vote strategy to classify the unlabeled instances. In class imbalanced situation AdaBoost algorithm performs efficiently as each of its iteration, minority class instances are misclassified and given higher weights to consider in subsequent iterations. AdaBoost is a most desired and used boosting algorithm, which assumes a sequence of classifiers and incorporates the votes of each isolate classifier for classifying an unknown or known instance [36]. In class imbalanced situation AdaBoost algorithm performs efficiently as each of its iteration the misclassified minority class instances are given higher weights. During every iteration of this algorithm, an infirm presumption is made by the foundation learner of the classifier.

Total k classifiers are iteratively acquired. First of all, every single instance is imposed to a uniform weight, $\frac{1}{d}$, where d represent all instances present in training data. Here a classifier M_i is learned after the weights are updated which will allow the following classifier M_{i+1}. Moreover, M^* is the final boosted classifier which will join the all the selected instances of every separate classifier. Now if the instances are accurately classified its weight will be reduced or if not correctly classified then it will increase weight. The AdaBoost algorithm is explained in algorithm 2.

Algorithm 2 AdaBoost Algorithm

Input: Training data, D, total, k, and a learning plan.
Output: Ensemble model, M^*
Method:

initialise weight, $x_i \in D$ to $\frac{1}{d}$;
for i = 1 to k **do**
  sample D with replacement according to instance weight to obtain D_i;
  use D_i, and learning plan to derive a method, M_i;
  calculate $error(M_i)$;
  if $error(M_i) \geq 0.5$ **then**
    Do step 3 and rerun;
  end if
  for each classify correctly $x_i \in D_i$ **do**
    multiply weight of x_i by $(\frac{error(M_i)}{1-error(M_i)})$;
  end for
  normalise weight of sample;
end for

To use M^* to classify a new sample, x_{New}:

initialise weight of each class to 0;
for i = 1 to n **do**
  $w_i = log\frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
  $c = M_i(x_{New})$; // class prediction by M_i
  add w_i to weight for class c;
end for
return largest weight class;

3.3 RUSBoost

RUSBoost is an ensemble classifier which is an extended version of SMOTEBoost that produces a quicker and easier another approach with an achievement which is normally perfect and sometimes more than that of SMOTEBoost. Seiffert et al. [37] say that RUSBoost is an unprecedented mongrel data sampling technique, which has a strong outline to enhance the performance of models. The accuracy of RUSBoost is measured with SMOTEBoost [38]. The common thing among the classifiers is that these two classifiers RUSBoost and SMOTEBoost introduce data sampling with the popular AdaBoost classifier. RUS, a sampling technique that arbitrarily dispels instances from the leading class [16] used by RUSBoost. On the other hand, SMOTEBoost [3] which constructs new minority class instances applied the SMOTE technique. The used of RUS technique into the boosting process are its intelligibility, fastness, and performance. On the other side SMOTE is a puzzled and extended data sampling technique which makes the SMOTEBoost more complex and sufferer by the disadvantages of prolonged model training time as compared to RUSBoost. As opposed to, the main advantage of RUS classifier is that reduces the period necessary to construct a model, especially when constructing an ensemble of models. Though there is no universally conventional optimal class distribution, a balanced (50: 50) distribution is often thought to be near optimal [14]. Moreover, when instances of the minority class are extremely uncommon, a ratio closer to 35: 65 (minority: majority) may result in better classification performance.

Algorithm 3 RUSBoost Algorithm

Input: Set S of examples $(x_1, y_1), ...(x_m, y_m)$ with minority class $y^r \in Y, |Y| = 2$
Weak Learner, WeakLearn
Number of iterations, T
Desired percentage of total instances to be represented by the minority class, N
Output: The final hypothesis:
H(x) $= argmax_{(y \epsilon Y)} \sum_{t=1}^{T} h_t(x, y) \frac{log1}{\alpha_t}$; **Method:**

1: initialise $\mathrm{D}_1(i) = \frac{1}{m} for all i$;
2: for t = 1,2, to ... T ;
3: Create temporary training dataset S'_t with Distribution D'_t using random undersampling;
4: Call $WeekLearn$, providing it with examples S'_t and their weights D'_t;
5: Get back a hypothesis h_t: X $\times Y \leftarrow [0, 1]$;
6: Calculate the pseudo-loss for (S and D_t):
$$\epsilon = \sum_{(i,y):y_t \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y));$$
7: Calculate the weight update parameter; $\alpha_t = \frac{\epsilon_t}{1-\epsilon_t}$;
8: Update Dt: $\mathrm{D}_{t+1}(i) = D_t(i) \alpha_t^{\frac{1}{2}} (1 + h_t(x_i, y_i) - h_t(x_i, y : y \neq y_i))$;
9: Normalize $\mathrm{D}_{t+1} : Let Z_t = \sum_i D_{t+1}(i)$;
$\mathrm{D}_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}$;
10: end for;

3.4 EUSBoost

Enhancing ensembles for extremely imbalanced datasets by evolutionary under-sampling is called as EUSBoost [39]. It is based on the RUSBoost method as it amalgamates random under-sampling with boosting technique [5, 40]. EUSBoost starts with the under-sampling technique that resulted until the currently finest under-sampled dataset cannot be further uplifted. This method is computationally more expensive compared to the RUSBoost algorithm as it executed EUS in all iterations of boosting. In real-world application the usefulness and aptness of EUS has been proved successfully [41]. But the main drawback of EUSBoost is that it seeks for perfect base learners that causes diversity loss in the final output. EUSBoost training phase is computationally more expensive than that of the other methods. This is due to EUS, which is executed in all iterations of Boosting.

Algorithm 4 EUSBoost, EUS embedded in AdaBoost.M2 Algorithm

Input: Training set $S = x_i, y_i, i = 1 \ldots N; and \quad \mathrm{y}_i \ \epsilon(c_1, c_2)$;
T: Number of iterations;
I; Weak Learner
Output: Boosted Classifier : H(x) : $arg\frac{max}{y\epsilon C} \sum_{t=1}^{T} ln\frac{1}{\beta_t}\mathrm{h}_t(x, y)$
where h_t, $\beta_t(with\mathrm{h}_t$ (x,y) $\epsilon[0, 1])$
are the classifiers and their assigned weights, respectively

Method:

1: D_1(i) $\leftarrow \frac{1}{N}$ $fori = 1, \ldots, N$;
2: W_i^1,$_y \leftarrow$D$_i$ $fori = 1, \ldots, N, y \neq$y$_i$
3: for t = 1,2, to ... T ;
4: W$_i^t \leftarrow \sum_y y_i W_i^t{}_y$;
5: q$_t(i, y) \leftarrow \frac{W_i^t{}_y}{W_i^t} fory \neq$y$_i$;
6: D$_t(i) \leftarrow \frac{W_i^t}{\sum_{x_i}} NW_i^t$;
7: S' = Evolutionary Under-sampling (S);
8: D' $_t(k) \leftarrow$
$\{ \ W_i^t \sum_{x_i \epsilon S'} W_i^t \quad if \ x_i \epsilon S'$
0 otherwise
9: $h_t \leftarrow$(S, D'$_t$);

$$\epsilon_t \leftarrow \frac{1}{2} \sum_{i=1} ND_t(i) \Bigg(1 - h_t(x_i, y_i) + \sum i, y \neq y_i q_t(i, y) h_t(x_i, y));$$

10: $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$;
11: W$^{t+1}_{i}$,$_y = W_i^t{}_y.\beta_t^(12)(1 + h_t(x_i, y_i) - h_t(x_i, y))fori = 1, \ldots, N, y \neq y_i$;

3.5 SMOTEBoost

SMOTEBoost combines the oversampling technique SMOTE with the AdaBoost algorithm, which outperforms on both the SMOTE and AdaBoost algorithms for learning from imbalanced data [42]. It acquires a balanced dataset applying SMOTE in each round of iteration of boosting process [3].

This is an extremely potential sampling/boosting method to learn from imbalanced data, that outperforms on both the SMOTE and AdaBoost algorithms. Before building the weak hypothesis during each round of iteration, to acquire a more balanced training data set SMOTE is applied to the training data. Hence, the algorithm for SMOTEBoost is analogous to that of RUSBoost. At this point, the application of SMOTE has two defects that RUSBoost is designed to overcome. First, it enhances the complexity of the algorithm. SMOTE must find the k nearest neighbors of the minority class examples and have to make an educated guess between them to make new instances. RUS, contrariwise, simply eliminate the majority class instance indiscriminately. Second, SMOTE requires elongated time to train the model, as it is an oversampling technique. When using larger training dataset many models required to build for the accurate output which produces elongated training time, whereas, shorter model training times will be required for smaller training data sets[3]. The possibility of creating redundancy. If only the borderline and noisy majority instances are removed then the selection procedures tried to find a representative subset of the majority samples. As a result, the complexity of the algorithm increased and training time of the model have elongated as well, which makes the algorithm more complex and prolonged than RUS.

Algorithm 5 SMOTEBoost Algorithm

Input : Set S $(x_1, y_1), \ldots, (x_m, y_m) x_i \epsilon X$,with labels $y_i \ \epsilon Y = 1, \ldots, C$,
where $C_m, C_m < C$,corresponds to a minority class.
$LetB = (i, y) : i = 1, \ldots, m, y \neq yi$;
Initialize the distribution D1 over the examples, such that$D1(i) = 1/m$.
$Fort = 1, 2, 3, 4, \ldots T$
Output : The final hypothesis :
$h_{fn} = arg\, max_{y \epsilon Y} \sum_{t=1}^{T} (log \frac{1}{\beta_t}) \times h_t(x, y)$
Method:

1: Modify distribution Dt by creating N synthetic examples from minority class Cm using SMOTE algorithm;
2: Train a weak learner using distribution Dt ;
3: Compute weak hypothesis ht: X $\times Y \leftarrow [0, 1]$;
4: . Compute the pseudo-loss of hypothesis h_t :
$\varepsilon_t = \sum_{(i,y)_{\epsilon B}} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$;
5: Set $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$ *and*
$W_t = (1/2) \times (1 - h_t(x_i, y) + h_t(x_i, y_i))$;
6: Update D_t : $D_{t+1}(i, y) = (D_t(i, y)/Z_t) \times \beta_t^w$,
where Z_t is a normalization constant chosen such that $D_{t+1} isadistribution.$

3.6 MSMOTEBoost

MSMOTEBoost combines the MSMOTE (Modified Synthetic Minority Over-sampling Technique) and AdaBoost algorithm [30]. MSMOTE is the modified and improved form of SMOTE method. MSMOTE categorized the minority class instances into three types according to the distances between the instances. Such as border instances, security instances and noisy instances [43]. By using the nearest neighbors method MSMOTE algorithm generates synthetic instances. The classification of MSMOTEBoost is more accurate than SMOTEBoost for classifying minority class instances [44]. MSMOTE generates synthetic samples by calculating the space between each minority samples using the nearest neighbor technique. MSMOTE Technique has maintained the following strategy are as follows: first of all the algorithm arbitrarily selects a data point from the nearest neighbor instances for the security instances. After that, the nearest neighbor for the border instances is selected and doesn't perform anything for latent noise instances. It is mandatory for this method to compute the space among the instances of a negligible class and all the instances in order to judge the samples type. Security instances can improve the function of the classifier. Whereas, Noisy instances reduces the performance of a model and border instances is very hard to classify for the model. The experimental analysis proved that the prediction accuracy of MSMOTE model is outperformed than SMOTEBoost in case of negligible class [44].

Algorithm 6 MSMOTE(D,x, N, k)

Input: total instances D, minority class instances x; Amount of SMOTE N%; Total nearest neighbors k
Output: synthetic minority class instance (N%*x)
Method:

1: k = total nearest instances;
2: N=N%*x //Total generating instances;
3: totalAttr = Total attributes ;
4: Instance[ ][ ]: array of minority class intances;
5: newPosition: records total synthetic instances created, started from 0;
6: Synthetic[ ][ ]: array for synthetic instances;
7: FOR i * 1 to x//(Total minority class) ;
8: Calculate k nearest neighbors for i, and store the indices in the nnarray and evaluate type of instance
9: initialise each class weight to 0;
10: type $\neq$ 0; Here, 0 is latent noises
11: Populate(N, i, nnarray=type)
12: ENDFOR
13: Populate (N, i, nnarray=type). (synthetic instances generating function.)
14: while N = 0
15: if (type==1) (1: secutity instances 2 border instances)
16: Randomly select one of the k nearest neighbors of i. call it nn. ;
17: else
18: Select nearest neighbors of i., call it nn. ;
19: for attr * 1 to totalAttr
20: Calculate: dif = Instance[nnarray[nn]][attr] * Instance[i][attr]
21: Calculate: gap = select random number between 0 and 1
22: Synthetic[newPosition][attr] = Instance[i][attr] + gap* dif
23: endfor
24: newPosition++
25: N = N *q_1;
26: endwhile;
27: return (Populate function end)

3.7 DataBoost

The DataBoost-IM is an ensemble classifier that amalgamates data creation and boosting processes to achieve high classification accuracy to classify both the majority and minority class instances without omitting the majority and minority classes[12]. It built with the following three phases. Step1, an equal weight is assigned to each training instances. Here, the original training set is applied to build the initial classifier. Step2, the difficult instances are identified and for each of these core instances, a set of artificial instances is created. Finally, at Step 3, the artificial majority and minority class instances are merged with original training set to create a balanced dataset.

Algorithm 7 DataBoost-IM Algorithm

Input: Series of m instance $\{(x_1, y_1), ..., (x_m, y_m)\}$
with labels $y_i \epsilon Y = 1, \ldots, k$
Weak learning model **WeakLearn**
Integer T identify total iterations
Initialize: $D_1(i) = 1/m$ for all i .

Output: Final presumption:
$h_{fin}(x) = argmax_{y \epsilon Y} \sum_{t:h_t\,(x)=y} log\frac{1}{\beta}$
Method:

1: **Do:** for t = 1, 2, . . . , T .
2: Specify hard instance from main data set for different classes ;
3: Produce synthetic data to balance training knowledge of different classes
4: Add synthetic data form a new training data set to main training set;
5: Update and balance total weights of different classes in new training data set;
6: Call WeakLearn, providing it with new training set with synthetic data and rebalanced weights;
7: Get back a presumption $h_t : X \leftarrow Y$;
8: Calculate error of $h_t : \epsilon_t = \sum_{i:h\ x_i \neq y_i} D_t(i)\ if\ \varepsilon_t > 1/2$,
then set T = t - 1 and exit loop.
9: Set $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$;
10: Update ordering
$D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \frac{\beta_t \leftarrow if h_t\,(x_i)=y_i}{1 \leftarrow otherwise} \end{cases}$, $where Z_t$ is a normalization constant. (chosen so that D_{t+1} will be a ordering).

3.8 EasyEnsemble

EasyEnsemble builds a classifier to lead the sampling technique for following classifier and to further utilize the majority class instances disregarded by under-sampling [45]. It has the advantage of combining boosting and bagging methods with data balancing method. The core concept EasyEnsemble is straighforward and in some direction it is analogous to the balanced Random Forest algorithm [46]. Here the technique starts by producing T regular subdataset. The i'th subdataset will outputed an adaboost classifier named as H_i, and an ensemble with s_i and a weak classifiers h_i, j. An alternative view of $h_{i,j}$ is it extracted by the ensemble learning method and only accepts the binary datas [47]. N_i comprise details about the original majority training set N. Finally, rather not taking votes from H_i where i=1,..., T, the features will accept the valus of $h_{i,j}$ where (i=1,2,..., T, j=1,2,..., s_i) and form an ensemble classifier. The final output of this method is a individual ensemble, though it act like an ensemble of ensembles. Diverse researches [48], [49], [50], [22] amalgamates various ensemble techniques to attain stronger generalization. MultiBoosting [49], [50] amalgamates boosting and bagging [36] through applying boosted ensembles as foundation learners. Different ensemble strategies are combined to release a better solution like Stochastic Gradient Boosting [51] and Cocktail Ensemble [22].

Algorithm 8 EasyEnsemble Algorithm

Input: A set of minority class examples P , a set of majority class examples N, $|P|$ $<|N|$, the number of subsets T to sample from N, and s_i, the number of iterations to train an AdaBoost ensemble H_i

Output: An ensemble;

$$H_i(x) = sgn\left(\sum_{i=1}^{T}\sum_{i=1}^{S_i}\alpha_{i,j}\,h_{i,j}\,(x)\sum_{i=1}^{T}\theta_i\right);$$

Method:

1: $i \Leftarrow 0$;
2: repeat;
3: $i \Leftarrow i + 1$;
4: Randomly sample a subset N_i from N , $|N_i| = |P|$;
5: Learn H_i using P and N_i.H_i is an AdaBoost ensemble with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$ .The ensemble's threshold is $\theta_i, i.e.$,;
6: $\mathrm{H}_i(x) = sgn\left(\sum_{i=1}^{S_i}\alpha_{i,j}\,h_{i,j}\,(x) - \theta_i\right)$;
7: until i = T ;

3.9 BalanceCascade

BalanceCascade uses an under-sampling process to combine weak classifiers [52]. In this method, the sequential reliance among classifiers is predominantly utilized for lessening the duplicate information in the majority class. It conducts instance space for the under-sampling method to pursue proper knowledge [18]. Assume that P is the number of minority training set and N is the majority training set. H_i is a classifier instructed by N_i and the total number of AdaBoost ensemble is P. Given the minority training set P and the majority training set N, a classifier H_i is trained using N_i and all of P, an AdaBoost ensemble. Taking subset from N and applied them to teach the classifiers. BalanceCascade investigates N in a unique way that is similar to the supervised method. Firstly trained H_1 classifier, after that if an instance x_1 is accurately classified H_1, it is logical to guess that x_1 is a bit unnecessary, in the meanwhile H_1 is collected. Thus, some correctly classified majority class instances can be removed from N. To some extent, this process is similar to the cascade classifier in [53] that is why this procedure is called as BalanceCascade. N is getting smaller consecutively in each step of H_i training, and at every phase, H_i have to calculate with a subproblem ($|N_i| = |P|$). Finally a cascade classifier H_i will be created where i=1,...,T, i.e..). To be successful in the rapid testing speed test the usually the balanced cascade classifier is an appropriate choice [45]. BalanceCascade omitting the duplicate instances in the majority class and confined instance difference to pursue convenient knowledge. BalanceCascade and EasyEnsemble both are analogous to their structures. The most popular technique called stacking [52] which combine weak classifiers with EasyEnsemble and BalanceCascade.

Algorithm 9 BalanceCascade Algorithm

Input: A set of minority class examples P , a set of majority class examples N, $|P|$ $<|N|$, the number of subsets T to sample from N, and s_i, the number of iterations to train an AdaBoost ensemble H_i

Output: A single ensemble;

$$\mathrm{H}_i(x) = sgn\left(\sum_{i=1}^{T}\sum_{i=1}^{S_i}\alpha_{i,j}\,h_{i,j}\,(x)\sum_{i=1}^{T}\theta_i\right);$$

Method:

1: $i \Leftarrow 0$; f $\Leftarrow$ $\sqrt[t-1]{|P|/|N|}$ is the false positive rate (the error rate of misclassifying a majority class example to the minority class) that H_i should achieve.

2: repeat;

3: $i \Leftarrow i + 1$;

4: Randomly sample a subset N_i from N , $|N_i| = |P|$;

5: Learn H_i using P and N_i.H_i is an AdaBoost ensemble with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$ .The ensemble's threshold is $\theta_i, i.e.$;

6: $$\mathrm{H}_i(x) = sgn\left(\sum_{i=1}^{S_i}\alpha_{i,j}\,h_{i,j}\,(x) - \theta_i\right);$$

7: until i = T ;

3.10 OverBagging

OverBagging is the ensemble process of combining over-sampling with bagging methods. It randomly over-sampling minority classes in each bagging iteration. It uses majority-voting technique to classify new instances, as each classifier gives their decision and final classification made by majority of votes. There is a possibility of occurring a tie that may return the class with negligible instances. [44]. The entire over-bagging method can be divided into 3 order from its training phase to testing phase i.e. i) re-sampling, ii) constructing ensemble and iii) voting. Because of multiple minorities and majority class, it is very complicated to choose the re-sampling rate.The output of the algorithm shows that variety of instances dominates recall value notably. The larger diversity consequences excellent recall for a minority whereas poor recall for majority classes.

Algorithm 10 OverBagging Algorithm

Input: Training data,
Output: Generate outputs from each model.
Return most voted class.
Method:

1: Let D, original training set.
2: Generate subset DK containing Samples from all classes with same number by executing the following:
3: (a) Set re-sampling ratio at a%.
4: (b) For each class i, re-sample Samples with replacement at the ratio of (NC /Ni) a%.
5: Train a model from DK.
6: Do step 2 and 3 until k equals M.

3.11 UnderBagging

UnderBagging method uses random under-sampling technique of majority class instances with bagging to build an ensemble classifier [7]. It also considers majority voting of classifiers to classify a new/ known instance. We may lost some valuable informative majority class instances when randomly sampling majority class instances. The core idea of this method is to educate each of the single components of the ensemble with a balanced learning instances. By replacing an individual classification model, (Here the 1-NN rule) with an imbalanced training set, by an amalgamation of various classifiers, each using a balanced training set for its learning process. To earn this, as many training sub-samples as possible to get balanced subsets are beget. The number of sub-samples will be discovered by the difference between the number of prototypes from the majority class and that of the minority class. This workflow makes it possible to properly handle the difficulties of the imbalance, and stay away from the implicit disadvantages to both the over and undersampling techniques .

Algorithm 11 UnderBagging Algorithm

Input: Training data
Output: Generate outputs from each model.
Return most voted class.

Method:

1: Let D, main training set.
2: Generate subset DK containing samples from all classes with same number by executing the following:
3: (a) Set re-sampling ratio at a%.
4: (b) For each class i, re-sample samples with replacement at the ratio of (NC /Ni) a%.
5: Train a model from DK.
6: Do step 2 and 3 until k equals M.

3.12 UnderOverBagging

From the above discussion we can see that OverBagging is utterly similar to UnderBagging either using random oversampling or random undersampling. So according to this concept UnderOverBagging trains the base classifier with varying resampling rates using random oversampling and random undersampling according to necessity [44].

3.13 SMOTEBagging

SMOTEBagging builds a model by employing SMOTE with bagging methods to balance the group ordination. It over-sampling from the negligible class instances using SMOTE. In SMOTEBagging, we need to set the volume of over-sampling and the nearest-neighbors of negligible class instances [44]. SMOTEBagging is differed from UnderBagging and OverBagging, by involving in subset creation of synthetic instances. As claimed by the SMOTE algorithm, at first the (k) nearest neighbors and the volume of over-sampling from minority class (N) should be set. Here N=100, 200, 300, 400 and 500 i.e K=5 nearest neighbour. The correlative class must be reviewed at the time of all minority class ordination. An example is illustrated below: assume that the minority class X consist 10 instances and the majority class Y consist of 60 instances. To over-sampling, both X and Y through the same N value are applied this two class are yet inner-imbalanced. In SMOTEBagging, x% value used to control the instances from each class that is used for propagating novel instances.

Algorithm 12 SMOTEBagging Algorithm

Input: Training data
Output: Generate outputs from each model.
Return most voted class;
Method:

1: Let D, Main training set;
2: Generate subset Dk containing samples from all classes with same number by executing the following:;
3: Re-sample class C with replacement at ratio 100%.
4: For each class i (1, . . . , C - 1):
   Re-sample from Main samples
   with replacement at the rate of (NC /Ni) $\dot{b}$%.
   Set N = (NC /Ni) (1 - b%) 100.
   Generate new samples by using SMOTE (k, N);
5: Train a model from Dk.;
6: Change ratio b%
7: Do step 2 and 3 until k equals M

3.14 Summary

After the new taxonomy of machine learning has been evolved, an increasing amount of research has been done to boost up the performance imbalance classification. The review has been accomplished by comparing 12 state-of-the-art methods using 22 real-world imbalanced datasets. Here, C4.5 considered as a base classifier for all the algorithm, as it has been extensively used by the imbalanced dataset classifier. However, ensemble learning is more complicated to implement and understand and single classifier proved faster and more efficient. I have pointed some comparison among the 12 state-of-the-art technique which are illustrated in Table 3.1. The pros and cons of the 12 state of the art technique is shown in tabulated form here in below:

Table 3.1: Pros and Cons of different algorithm

SN	Algorithm	Advantages	Dis Advantages
1	SMOTE	The classifiers are very much unique and computationally simple [3].	It is a perplexed and prolonged procedure.
2	AdaBoost	Prediction accuracy improve in case of minority data set and resolve the multi class imbalanced data set problem [30]	Ignore overall performance of the classifier.
3	RUSBoost	Simpler, easier, faster and less complex algorithm. RUS has been operated skilfully and swiftly [54].	The major handicap of this model is that important data can be destroyed.
4	EUSBoost	Easier, quicker and outperformed in highly imbalanced datasets.	Computationally more expensive than that of the other methods [55].
5	SMOTEBoost	This method is a perfect selection when the training instance size is excessively huge [3].	The possibility of creating redundancy. Besides more complex and prolonged than RUS.
6	DataBoost	Produce high accuracies of both classes (minority and majority)[12].	The use of many classifiers makes them more complex and produces output that is very hard to analyze
7	MSMOTE Boost	This method has better prediction accuracy [56].	Time-consuming and perplexing task.
8	Easy Ensemble	Reduce information loss. Faster training speed of undersampling [3].	Examining only binary classification problems [45].
9	BalanceCascade	Reduces the rate of creating duplicate information. Most popular method is stacking [52] .	Performance standard reduces in multi class.
10	Over Bagging	Performed masterly both in the binary and multi-class data set	Performance standard degrades in minority class [44].
11	Under Bagging	This procedure makes it possible to properly handle the difficulties of both the over and undersampling techniques [7].	If the available data size is not plenty the performance may be deteriorated.
12	SMOTE Bagging	Increased accuracy level [7].	Biased in favor of the majority class on high-dimensional data

Chapter 4

Experimental Analysis

4.1 Imbalanced Datasets

In this thesis work, we have considered 22 binary datasets from KEEL dataset repository [57]. Table 4.1 gives an outline of the features of the selected datasets. Imbalanced datasets are a unique case for classification problem where the class distribution is not equal between the classes. Generally, they have formed of two classes: The majority class and the minority class. These type of datasets makes a distinct challenging difficulty for Data Mining as conventional classification algorithms normally influenced to the majority class.

Table 4.1: Datasets description.

Sl No.	Name	Feature	Instance	Imbalanced Ratio
1	glass1	9	214	1.82
2	ecoli0_vs_1	7	220	1.86
3	wisconsin	9	683	1.86
4	pima	8	768	1.87
5	iris0	4	150	2
6	glass0	9	214	2.06
7	yeast1	8	1484	2.46
8	haberman	3	306	2.78
9	vehicle2	18	846	2.88
10	vehicle1	18	846	2.9
11	vehicle3	18	846	2.99
12	glass0123_vs_456	9	214	3.2
13	vehicle0	18	846	1.82
14	ecoli1	7	336	1.82
15	new-thyroid1	5	215	1.82
16	new-thyroid2	5	215	1.82
17	ecoli2	7	336	1.82
18	segment0	19	2308	1.82
19	glass6	9	214	1.82
20	yeast3	8	1484	1.82
21	ecoli3	7	336	1.82
22	page-blocks0	10	5472	1.82

4.2 Performance Measures

A confusion matrix is a table that describe the performance of a classification model or a classifier on a set of test data for which the true values are known, as illustrated in Table 4.2.

Table 4.2: 2x2 Confusion Matrix

Confusion matrix		
	Positive Prediction	Negative Prediction
Positive class	True Positive	False Negative
Negative class	False Positive	True Negative

The most evaluated metrics related to imbalanced classes are: 1) Recall (sensitivity) 2) Specificity 3) Precision 4) F- measure and 5) Geometric mean (g-mean) [4]. Sensitivity (also called True Positive Rate) and specificity (also called True Negative Rate) both monitor the classification accuracy in every separate class. On the other hand, AUROC curve is the ratio between true positive rate (TPR) along the y-axis and false positive rate (FPR) along the x-axis. The classifier has an AUROC value of 1 acknowledged as a high-performance classifier, where TPR = 1 and FPR = 0.The Area Under the ROC Curve (AUC) is a good metric for evaluation of classifiers for imbalanced dataset. At various cutoff points, the curve displays the TPR and FPR ratio of the classifier at different range shows in the following table 4.3:

Table 4.3: Confusion Measure Extended

SN	**Name**	**Definition**	**Formula**	**Explanation**
1	True Positive (TP Rate)	An instance which is positive and correctly classified as positive.	TP / (T P + F N)	The closer to 1 , the better. TP Rate = 1 when FP = 0. (No False Positives)
2	True Negative (TN Rate)	An instance which is negative and correctly classified as negative.	TN / (T N + F N)	The closer to 1 , the better. TN Rate = 1 when FN = 0. (No False Negatives).
3	False Positive (FP Rate)	An instance which is negative but incorrectly classified as positive.	FP / (FP + T N)	The closer to 0 , the better. FP Rate = 0 when FP = 0. (No False Positives).
4	False Negative (FN Rate)	An instance which is positive but incorrectly classified as negative	FN / (FN + TP)	The closer to 0 , the better. FN Rate = 1 when FP = 0. (No False Negatives).

4.2.1 ROC Curves (Receiver operating characteristic)

In the case of imbalanced classification issues, insistence is established notably on the predictive accuracy of minority/positive class while having accuracy for the majority/negative class. This would correlate to high TPR and low FPR, and thus reflected

by a high AUROC. In this review, we have used AUROC technique for comparing all the state-of-the-art techniques. AUROC is a very significant process used in multiple sectors of data mining like in signal detection field, statistical sector, medical diagnosis [18], and more recently in machine learning sector and alternative method for comparing learning systems as well [58].

ROC space prevails a coordinate method applied for discerning the performance of a classifier, where the false positive rate imposed on the x-axis and the true positive rate is plotted on the y-axis. By this process, classifiers in a plane are compared by point, not by numbers. For classifiers obtained by thresholding a real-valued function or depending on a real parameter, this produces a curve, called a ROC curve, describing the trade-off between sensitivity and specificity. Two systems can, therefore, be compared with the better one being the highest and leftmost one. Here we tried to outline the methods for controlling the balance between the 0-diagonal terms in the confusion matrix.

4.2.2 Results

We have tested the performance of 12 machine learning algorithms for imbalanced data classification using the area under an ROC curve (AUC) with 5-fold cross-validation. The experimental results are tabulated in Table 4.4 and Table 4.5. And Fig. 4.1 shows the comparison of 12 classifiers using AUC on 22 imbalanced datasets.

Table 4.4: The AUC comparison of SMOTE, AdaBoost, RUSBoost, SMOTEBoost, EUSBoost, DataBoost on 22 imbalanced datasets.

Dataset	SMOTE	Ada Boost	RUS Boost	SMOTE Boost	EUS Boost	Data Boost
glass1	**0.992**	0.809	0.77	0.783	0.783	0.7
ecoli0_vs_1	**1**	0.969	0.976	0.972	0.962	0.94
wisconsin	0.955	0.961	0.956	0.965	0.944	0.960
pima	0.731	0.689	0.739	0.729	0.7	0.711
iris0	**1**	0.99	0.99	0.99	0.99	0.99
glass0	**0.875**	0.823	0.852	0.823	0.845	0.835
yeast1	**0.769**	0.652	0.704	0.712	0.688	0.699
haberman	**0.741**	0.581	0.629	0.614	0.554	0.561
vehicle2	**0.968**	0.966	0.972	0.981	0.971	0.966
vehicle1	0.753	0.7	0.743	0.752	**0.79**	0.966
vehicle3	0.755	0.681	0.764	0.744	0.722	0.691
glass012-3_vs_456	**0.953**	0.877	0.879	0.918	0.914	0.897
vehicle0	**0.959**	0.931	0.958	0.976	0.922	0.944
ecoli1	**0.961**	0.822	0.912	0.847	0.898	0.806
new-thyroid1	0.986	0.931	0.932	**0.988**	0.96	0.957
new-thyroid2	**0.972**	0.954	0.918	0.971	0.935	0.957
ecoli2	**0.964**	0.851	0.865	0.909	0.928	808
segment0	**0.997**	0.99	0.991	0.993	0.991	0.991
glass6	**0.986**	0.849	0.909	0.855	0.906	0.841
yeast3	**0.918**	0.840	0.916	0.887	0.885	0.897
ecoli3	**0.95**	0.716	0.871	0.858	0.881	0.812
page-blocks0	0.964	0.930	0.947	0.939	0.902	0.874

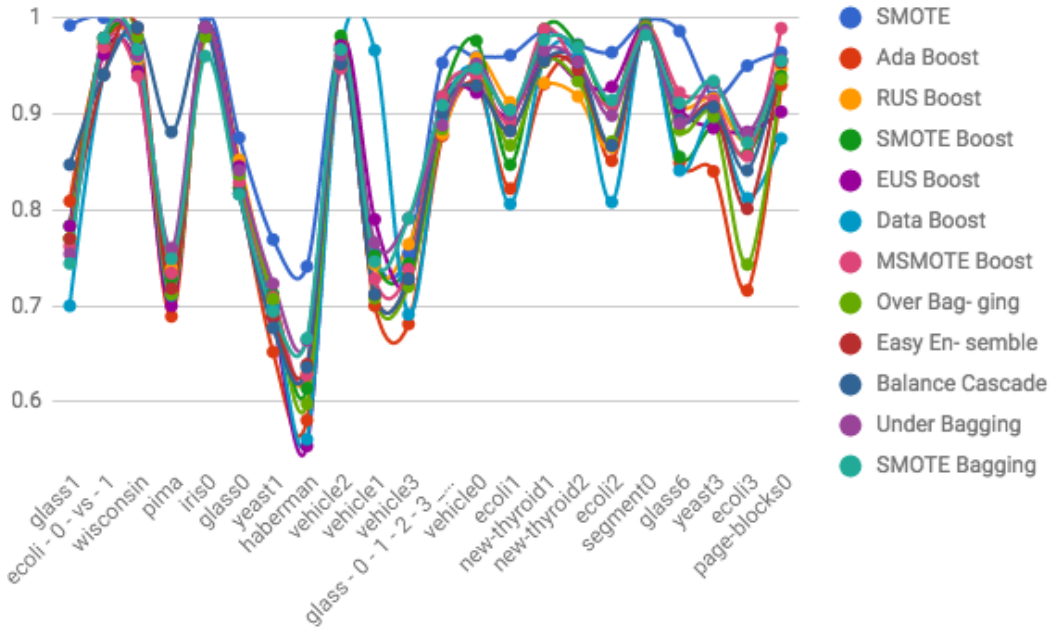

Figure 4.1: The AUC comparison of SMOTE, AdaBoost, RUSBoost, SMOTEBoost, EUSBoost, DataBoost, MSMOTEBoost, Over-Bagging, EasyEnsemble, BalanceCascade, Under-Bagging and SMOTEBagging on 22 imbalanced datasets.

Table 4.5: The AUC comparison of MSMOTEBoost, Over-Bagging, EasyEnsemble, BalanceCascade, Under-Bagging, SMOTEBagging on 22 imbalanced datasets.

Dataset	**MSMOTE Boost**	**Over Bagging**	**Easy Ensemble**	**Balance Cascade**	**Under Bagging**	**SMOTE Bagging**
glass1	0.762	0.77	0.77	0.847	0.754	0.744
ecoli0_vs_1	0.969	0.979	0.941	0.941	0.979	0.979
wisconsin	0.939	0.98	0.989	**0.99**	0.961	0.967
pima	0.734	0.712	0.718	**0.881**	0.76	0.749
iris0	0.99	0.98	0.99	0.99	0.99	0.96
glass0	0.828	0.837	0.817	0.817	0.841	0.816
yeast1	0.709	0.707	0.689	0.677	0.723	0.694
haberman	0.628	0.598	0.639	0.636	0.664	0.666
vehicle2	0.947	0.960	0.952	0.952	0.961	0.967
vehicle1	0.728	0.708	0.712	0.712	0.766	0.746
vehicle3	0.738	0.720	0.728	0.728	**0.791**	**0.791**
glass012-3_vs_456	0.918	0.884	0.901	0.901	0.888	0.909
vehicle0	0.939	0.949	0.929	0.929	0.952	0.947
ecoli1	0.892	0.867	0.882	0.882	0.899	0.904
new-thyroid1	**0.988**	0.954	0.955	0.955	0.966	0.977
new-thyroid2	0.948	0.934	0.946	0.954	0.954	0.969
ecoli2	0.909	0.871	0.867	0.867	0.898	0.914
segment0	0.991	0.990	0.985	0.987	0.985	0.982
glass6	0.922	0.883	0.893	0.893	0.890	0.911
yeast3	0.914	0.897	0.907	0.907	0.932	0.934
ecoli3	0.856	0.743	0.801	0.841	0.880	0.870
page-blocks0	**0.989**	0.936	0.953	0.956	0.956	0.955

Table 4.6: Performance of each algorithm on 22 imbalance datasets

Dataset	1st	2nd	3rd	4th	5th	6th
glass1	**SMOTE**	Balance Cascade	Ada Boost	SMOTE Boost	EUS Boost	Over Bagging
ecoli0_vs_1	**SMOTE**	Over Bagging	Under Bagging	SMOTE Bagging	RUS Boost	SMOTE Boost
wisconsin	Balance Cascade	Easy Ensemble	Over Bagging	SMOTE Bagging	RUS Boost	SMOTE Boost
pima	Balance Cascade	Under Bagging	SMOTE Bagging	RUS Boost	MSMOTE Boost	SMOTE
iris0	**SMOTE**	Ada Boost	RUS Boost	RUS Boost	SMOTE Boost	Data Boost
glass0	**SMOTE**	RUS Boost	EUS Boost	Under Bagging	Over Bagging	Data Boost
yeast1	**SMOTE**	Under Bagging	SMOTE Boost	MSMOTE Boost	Over Bagging	RUS Boost
haberman	**SMOTE**	SMOTE Bagging	Under Bagging	Easy Ensemble	Balance Cascade	RUS Boost
vehicle2	SMOTE Boost	RUS Boost	EUS Boost	SMOTE	SMOTE Bagging	Ada Boost
vehicle1	Data Boost	EUS Boost	Under Bagging	SMOTE	SMOTE Boost	SMOTE Bagging
vehicle3	Under Bagging	SMOTE Bagging	RUS Boost	SMOTE	SMOTE Boost	MSMOTE Boost
glass012-3_vs_456	**SMOTE**	SMOTE Boost	MSMOTE Boost	SMOTE Bagging	EUS Boost	Easy Ensemble
vehicle0	SMOTE Boost	SMOTE	SMOTE	Under Bagging	Over Bagging	SMOTE Bagging
ecoli1	**SMOTE**	RUS Boost	SMOTE Bagging	Under Bagging	EUS Boost	MSMOTE Boost
new-thyroid1	**SMOTE**	MSMOTE Boost	SMOTE	SMOTE Bagging	Under Bagging	Data Boost
new-thyroid2	**SMOTE**	SMOTE Boost	SMOTE Bagging	Data Boost	Balance Cascade	Under Bagging
ecoli2	**SMOTE**	EUS Boost	SMOTE Bagging	SMOTE Boost	MSMOTE Boost	Under Bagging
segment0	**SMOTE**	SMOTE Boost	EUS Boost	Data Boost	MSMOTE Boost	Over Bagging
glass6	**SMOTE**	MSMOTE Boost	SMOTE Bagging	RUS Boost	EUS Boost	Easy Ensemble
yeast3	SMOTE Bagging	Under Bagging	SMOTE	RUS Boost	MSMOTE Boost	Easy Ensemble
ecoli3	**SMOTE**	EUS Boost	Under Bagging	RUS Boost	SMOTE Bagging	SMOTE Boost
page-blocks0	MSMOTE Boost	SMOTE	Balance Cascade	Under Bagging	SMOTE Bagging	RUS Boost

Table 4.7: Performance of each algorithm on 22 imbalance datasets (Cont.

Dataset	7th	8th	9th	10th	11th	12th
glass1	Easy Ensemble	RUS Boost	MSMOTE Boost	Under Bagging	SMOTE Baggingt	Data Boost
ecoli0_vs_1	Ada Boost	MSMOTE Boost	EUS Boost	Easy Ensemble	Balance Cascade	Data Boost
wisconsin	Ada Boost	Under Bagging	Data Boost	SMOTE	EUS Boost	MSMOTE Boost
pima	SMOTE Boost	Easy Ensemble	Over Bagging	Data Boost	EUS Boost	Ada Boost
iris0	MSMOTE Boost	Easy Ensemble	Balance Cascade	Under Bagging	Over Bagging	SMOTE Bagging
glass0	MSMOTE Boost	Ada Boost	SMOTE Boost	Easy Ensemble	Balance Cascade	SMOTE Bagging
yeast1	Data Boost	SMOTE Bagging	Easy Ensemble	EUS Boost	Balance Cascade	Ada Boost
haberman	MSMOTE Boost	SMOTE Boost	Over Bagging	Ada Boost	Data Boost	EUS Boost
vehicle2	Data Boost	Under Bagging	Over Bagging	Easy Ensemble	Balance Cascade	MSMOTE Boost
vehicle1	RUS Boost	MSMOTE Boost	Easy Ensemble	Ada Boost	Over Bagging	Balance Cascade
vehicle3	Easy Ensemble	Balance Cascade	EUS Boost	Over Bagging	SMOTE Boost	Ada Boost
glass012-3_vs_456	Balance Cascade	Data Boost	Under Bagging	Over Bagging	Data Boost	Ada Boost
vehicle0	Data Boost	MSMOTE Boost	Ada Boost	Easy Ensemble	RUS Boost	EUS Boost
ecoli1	Easy Ensemble	Balance Cascade	Over Bagging	SMOTE Boost	Balance Cascade	Data Boost
new-thyroid1	EUS Boost	Easy Ensemble	Balance Cascade	Over Bagging	Ada Boost	Ada Boost
new-thyroid2	Ada Boost	MSMOTE Boost	Easy Ensemble	Over Bagging	RUS Boost	EUS Boost
ecoli2	Over Bagging	Easy Ensemble	Balance Cascade	RUS Boost	EUS Boost	Data Boost
segment0	RUS Boost	Ada Boost	Balance Cascade	Under Bagging	Ada Boost	SMOTE Bagging
glass6	Balance Cascade	Under Bagging	Over Bagging	SMOTE Boost	Easy Ensemble	Data Boost
yeast3	Balance Cascade	Over Bagging	Data Boost	SMOTE Boost	EUS Boost	Ada Boost
ecoli3	MSMOTE Boost	Balance Cascade	Data Boost	Easy Ensemble	Over Bagging	Ada Boost
page-blocks0	Easy Ensemble	SMOTE Boost	Over Bagging	Easy Ensemble	EUS Boost	Data Boost

Table 4.8: Performance of each algorithm on 22 imbalance datasets (Cont.)

1st	2nd	3rd	4th	5th	6th
SMOTE (13)	Balance Cascade (1)	Ada Boost (1)	SMOTE Boost (2)	EUS Boost (4)	Over Bagging (2)
SMOTE Boost (3)	Over Bagging (1)	Under Bagging (4)	SMOTE Bagging (4)	RUS Boost (2)	SMOTE Boost (3)
Balance Cascade (2)	Easy Ensemble (1)	Over Bagging (1)	RUS Boost (5)	MSMOTEBoost (4)	SMOTE (1)
Data Boost (1)	Under Bagging (3)	SMOTE Bagging (5)	Under Bagging (4)	SMOTE Boost (3)	Data Boost (3)
MSMOTE Boost (1)	Ada Boost (1)	RUS Boost (2)	MSMOTE Boost (1)	Over Bagging (3)	RUS Boost (3)
SMOTE Bagging (1)	RUS Boost (3)	EUS Boost (3)	Easy Ensemble (1)	Balance Cascade (2)	MSMOTE Boost (2)
Under Bagging (1)	SMOTE Bagging (2)	SMOTE Boost (1)	Data Boost (2)	SMOTE Bagging (3)	SMOTE Bagging (2)
	EUS Boost (3)	MSMOTEBoost (1)	SMOTE (3)	Under Bagging (1)	Easy Ensemble (1)
	SMOTE Boost (3)	SMOTE (3)			Under Bagging (2)
	SMOTE (2)	Balance Cascade (1)			SMOTE Bagging
	MSMOTE Boost (2)				Over Bagging (1)

Table 4.9: Performance of each algorithm on 22 imbalance datasets Cont

7th	8th	9th	10th	11th	12th
Easy Ensemble (4)	RUS Boost (1)	MSMOTE Boost (4)	Under Bagging (3)	SMOTE Bagging (1)	Data Boost (6)
Ada Boost (3)	MSMOTE Boost (4)	EUS Boost (2)	Easy Ensemble (6)	Balance Cascade (5)	MSMOTE Boost (2)
SMOTE Boost (1)	Under Bagging (3)	Over Bagging (6)	SMOTE (1)	EUS Boost (5)	Ada Boost (7)
MSMOTE Boost (4)	Easy Ensemble (4)	Balance Cascade (4)	Data Boost (1)	Over Bagging (3)	SMOTE Bagging (1)
Data Boost (3)	Ada Boost (2)	SMOTE Boost (1)	EUS Boost (1)	Data Boost (2)	EUS Boost (3)
RUS Boost (1)	SMOTE Bagging (1)	Easy Ensemble (3)	Ada Boost (2)	RUS Boost (2)	Balance Cascade (1)
Balance Cascade (3)	SMOTE Boost (2)	Under Bagging (1)	Over Bagging (4)	Ada Boost (3)	
EUS Boost (1)	Balance Cascade (3)	Ada Boost (1)	SMOTE Boost (3)	Under Bagging (1)	
Over Bagging (1)	Data Boost (1)	Data Boost (3)	RUS Boost (1)		
RUS Boost (1)	Over Bagging (1)				

4.3 Summary

We have tried to explore which is the appropriate and best approaches to managing a huge and high dimensional data-sets. The comparison is accomplished by the development of an empirical analysis of ensemble methods, which is conducted by statistical tests. To do so, we have applied the AUROC curve as the assessment standard. The experimental results summed up that SMOTE performed better in most datasets in compare with ensemble classifiers. On the other hand, ensemble classifiers are more complicated to implement and understand. In general, the combination of both over-sampling and under-sampling techniques with ensemble classifier such as bagging and boosting achieve the highest accuracy for classifying both majority and minority class instances.

Chapter 5

Conclusions and Future Work

5.1 Summary

In this thesis work, we, mostly involved in investigating imbalance classification problem and the receptivity of ensembles of classifiers. Contemporary algorithms have been reviewed and the best re-sampling strategies are proposed based on the studies. The performance of diverse learning problem is influenced by several causes which obstructed and sometimes reduced the classifier quality [59], [60], [61], [62], [63]. Multiple data environment are available like facile dataset, overlapped dataset and disconnect dataset, high-dimensional data set [64], [65]. Also if there is a situation of more than one minority class, the condition will be more complicated. In this thesis, we here examine how those imbalance data set regime the diverse classification algorithms. If the amount of training data is insufficient to form a perfect classifier it would be wise to use an ensemble classifier. Here we have discussed and compared total 12 learning classifiers. The magnetic part of the thesis is, it reviews the algorithm features that collected from the keel data set repository and derived a decision about the best performance among them.

5.2 Conclusion

In the last decade, several machine learning methods have been proposed for dealing with class imbalanced datasets to improve the performance of classifiers to classify minority class instances. In this thesis work, we have reviewed the several machine

learning algorithms for imbalanced data classification. We have tested the performances of 12 imbalanced data classification methods: SMOTE, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade, OverBagging, UnderBagging, SMOTEBagging on 22 datasets from KEEL Repository. The experimental results summed up that SMOTE performed better in most datasets in compare with ensemble classifiers. On the other hand, ensemble classifiers are more complicated to implement and understand. In general, combination of both over-sampling and under-sampling techniques with ensemble classifier such as bagging and boosting achieve the highest accuracy for classifying both majority and minority class instances.

5.3 Future Work

In future work, we will apply these imbalanced data classification methods in real-life high-dimensional imbalanced big data and apply sampling technique with modified adaboost algorithm.

Bibliography

[1] T. N and M. D., "Class imbalance and the curse of minority hubs," *Class imbalance and the curse of minority hubs*, vol. 24, no. 3, pp. 301–312, April 2013.

[2] H. He and E. A. Garcia, "Learning from imbalanced data. ieee transactions on knowledge and data engineering," *Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 15, pp. 1263–1284, November 2009.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 10, pp. 321–357, October 2002.

[4] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data. pattern recognition," *A novel ensemble method for classifying imbalanced data. Pattern Recognition*, vol. 48, no. 4, pp. 1623–1637, March 2014.

[5] M. Galar, A. Fernandez, and E. Barrenechea, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 4, no. 42, pp. 463–484, August 2012.

[6] D. M. Farid, L. Zhang, A. Hossain, C. MofizurRahman, R. Strachan, G. Sexton, and K. Dahal, "An adaptive ensemble classifier for mining concept drifting data streams," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5895–5906, November 2013.

[7] R. Barandela, R. Valdovinos, and J. Sánchez, "New applications of ensembles of classifiers," *Pattern Analysis and Applications*, vol. 6, pp. 245–256, 2003.

[8] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," *2005 International Conference on Intelligent Computing(ICIC05)*, vol. 3644, pp. 878–887, 2005.

[9] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "Rusboost: Improving classification performance when training data is skewed," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS,*, vol. 40, no. 1, January 2010.

[10] J. R. Quinlan, "Boosting, bagging, and c4.5,," *Proc. 13th Nat'l Conf. Artificial Intelligence*, pp. 725–730, 1996.

[11] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 5, pp. 535 – 539, May 1997.

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

[13] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, vol. 6, no. 1, pp. 1–6, June 2004.

[14] G. M. Weiss, "Sigkdd explorations. volume 6, issue 1 - page 7 mining with rarity: A unifying framework," *SIGKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[15] C. Ferri, P. A. Flach, and J. Hernández-Orallo, "Learning decision trees using the area under the roc curve," *ICML '02 Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 139–146, July 08 - 12 2002.

[16] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, vol. 6, no. 1, pp. 20–29, June 2004.

[17] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," *8th Conference on AI in Medicine in Europe(AIME01)*, vol. 2001, pp. 63–66, 2001.

[18] K. M. Ting, "An instance weighting method to induce cost-sensitive trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 3, pp. 659–665, 2002.

[19] L. Breiman, "Bagging predictors," *Machine Learning*, no. 12, pp. 123–140, May 1996.

[20] M. Pazzani, C. Merz, P. M. K. Ali, T. Hume, and C. Brunk, "Reducing misclassification costs," *Proc. 11th Int'l Conf. Machine Learning*, pp. 17–225, 1994.

[21] G. I. Webb, "Cost-sensitive specialization," *Pacific Rim International Conference on Artificial Intelligence*, pp. 23–34, 1996.

[22] Y. Yu, Z. H. Zhou, and K. M. Ting, "Cocktail ensemble for regression," *inProc. 7th IEEE Int. Conf. Data Mining, Omeha, NE*, pp. 721–726, 2007.

[23] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," *ICML'03 Workshop on Learning from Imbalanced Data Sets, Washington, DC*, 2003.

[24] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "Correspondence svms modeling for highly imbalanced classification," *IEEE Transactions on Systems and Man and and Cybernetics and Part B: Cybernetics*, vol. 39, no. 1, February 2009.

[25] V. N. Vapnik, "The nature of statistical learning theory," *Book by Vladimir Vapnik*, 2000.

[26] C. J. BURGES, "A tutorial on support vector machines for pattern recognition,," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 121–167, Jun 1998.

[27] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561–2573, July 2014.

[28] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods.the annals of statistics," *Boosting the margin: A new explanation for the effectiveness of voting methods.The Annals of Statistics*, vol. 20, no. 7, pp. 1651–1686, October 1998.

[29] D. M. Farid, L. Zhang, C. M. Rahman, M. Hossain, and R. Strachan, "Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1937–1946, March 2014.

[30] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[31] J. R. Koza, "Genetic programming: On the programming of computers by means of natural selection," *The MIT Press*, 1992.

[32] F. J. Berlanga, A. J. R. Rivas, M. J. D. Jesus, and F. Herrera, "Gp-coach: Genetic programming-based learning of compact and accurate fuzzy rule-based classification systems for high-dimensional problems," *Information Sciences*, vol. 180, pp. 1183–1200, 2010.

[33] A. BASTIAN, "How to handle the flexibility of linguistic variables with applications,," *International Journal of Uncertainty, Fuzziness and Knowledge- Based Systems*, vol. 2, pp. 463–484, 1994.

[34] VictoriaLópez, AlbertoFernández, M. J. Jesus, and FranciscoHerrera, "A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets," *Knowledge-Based Systems*, vol. 38, pp. 85–104, 2013.

[35] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth, "How to use expert advice," *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pp. 382–391, May 16 - 18 1993.

[36] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1-2, pp. 105–139, July 1999.

[37] T. M. Khoshgoftaar, C. Seiffert, and J. V. Hulse, "Learning with limited minority class data," *Proc. 6th ICMLA*, pp. 348–353, 2007.

[38] N. V. Chawla, N. Japkowicz, and A. K. lcz, "Special issue on learning from imbalanced data sets," vol. 6, no. 1, pp. 1–6, January 2008.

[39] S. G. a and F. Herrera, "Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy," *Evolutionary Computation*, vol. 17, pp. 275–306, 2009.

[40] MikelGalar, AlbertoFernández, EdurneBarrenechea, and FranciscoHerrera, "Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognition*, vol. 46, no. 12, pp. 3460–3471, 2013.

[41] D. J. Drown, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary sampling and softwarequality modeling of high-assurance systems," *IEEE Transactions on Systems,Man, and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 5, pp. 1097–1107, 2009.

[42] K. S. WOODS, C. C. DOSS, K. W. BOWYER, J. L. SOLKA, C. E. PRIEBE, and J. W. PHILIP KEGELMEYER, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *Int. J. Patt. Recogn. Artif. Intell.*, vol. 07(6), pp. 1417–1436, 1993.

[43] M. Buckland and F. Gey, "The relationship between recall and precision, journal of the american society for information science," *Journal of the American Society for Information Science*, vol. 45, no. 1, pp. 12–19, 1994.

[44] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," *IEEE Symposium Series on Computational Intelligence and Data Mining, IEEE CIDM 2009 Nashville TN, USA,*, vol. 19, no. 8, pp. 324–331, August 2009.

[45] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory under sampling for class imbalance learning," *Proc. Int'l Conf. Data Mining*, vol. 0, no. 0, pp. 965–969, May 2006.

[46] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–347, August 2014.

[47] G. Wu and E. Chang, "Kba: Kernel boundary alignment considering imbalanced data distribution," *IEEE Trans. Knowl. Data Eng*, vol. 17, no. 6, pp. 786–795, Jun 2005.

[48] J. H. F. riedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, February 2002.

[49] G. I. Webb, "Multiboosting: A technique for combining boosting and wagging," *Machine Learning*, vol. 40, no. 2, pp. 159–196, August 2000.

[50] G. I. Webb and Z. Zheng, "Multistrategy ensemble learning: Reducingerror by combining ensemble learning techniques," *EEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 980–991, August 2004.

[51] J. H. Friedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal*, vol. 38, no. 4, pp. 367–378, February 2002.

[52] D. H. Wolpert, "Stacked generalization," *Complex Systems Group, Theoretical Division, and Center for Non-linear Studies*, vol. 5, no. 2, pp. 241–260, 1992.

[53] P. A. VIOLA and M. J. JONES, "Robust real-time face detection," *International Journal of Computer Vision KL2213-05/5263672 10, 2004 20:17 UNCORRECTED PROOF International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, January 2004.

[54] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Transactions On Systems,*

Man, And Cybernetics—Part A: Systems And Humans, vol. 40, no. 1, pp. 176–190, January 2010.

[55] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognition*, vol. 46, no. 12, pp. 3460–3471, 2013.

[56] S. Hu, Y. Liang, L. Ma, and Y. He, "Msmote: Improving classification performance when training data is imbalanced," *2nd International Workshop on Computer Science and Engineering(WCSE 2009)*, pp. 13–17, 2009.

[57] 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.

[58] A. J. M. A. Afza, D. M. Farid, and C. M. Rahman, "A hybrid classifier using boosting, clustering, and naïve bayesian classifier," *World of Computer Science and Information Technology Journal (WCSIT)*, vol. 1, no. 3, pp. 105–109, 2011.

[59] G. M. Weiss, "Mining with rarity : a unifying framework." *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 7–19, 2004.

[60] S. Visa and A. Ralescu, "Issues in mining imbalanced data sets a review paper," *In Proceedings of the Sixteen Midwest Artificial Intelligence*, 2005.

[61] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets." *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 1–6, 2004.

[62] R. C. Prati, G. E. Batista, and M. C. Monard, "Class imbalances versus class overlapping: An analysis of a learning system behavior," *Lecture Notes in Computer Science*, vol. 2972, no. 312–321, 2004.

[63] F. Provost, "Machine learning from imbalanced data sets 101," *Extended Abstract*, 2017.

[64] N. Japkowicz, "Class imbalances: are we focusing on the right issue." *In Workshop on Learning from Imbalanced Data Sets II*, pp. 17–23, January 2003.

[65] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429 – 449, October 2002.

Appendix A

KEEL Data Mining Software

A.1 Introduction

Knowledge Extraction based on Evolutionary Learning (KEEL) is an open software under General Public License (GPLv3) compatible with Java environment which allows the end user to evaluate the performance of the evolutionary algorithm and computational intelligence based techniques for various kind of machine learning problems such as pattern recognition, classification, regression, clustering, etc.

The principal features of KEEL are:

- This is a package of an extensive collection of evolutionary algorithms for predicting outputs.
- It incorporates approximately 100 data preprocessing algorithms offered.
- It includes an analytical library to examine the outputs of the algorithms.
- It contains a set of statistical analyses.
- It presents a user-friendly interface.
- Independently offline/online run on any Java supported Virtual Machine..

A.1.1 Organization

The latest configuration of KEEL is organized into the following steps:

Figure A.1: The Main Menu Section

- **Data Management:** Through the data mining process, datasets that related to all the operations are managed by the data management section.
- **Experiments:** The experiments segment is outlined to perform a data mining experiment employing the graphical interface. This is the most strong segment as it allows the user to implement more than 500 algorithms by using any given dataset to fulfill a data mining experiment.
- **Educational:**This section designed mainly for the teaching environment, which gives a glimpse of the development of the algorithms in order to achieve its objective.
- **Modules:** Here new modules are included which enlarging the functionalities of the software for particular tasks. This Process also divided into four segments such as:

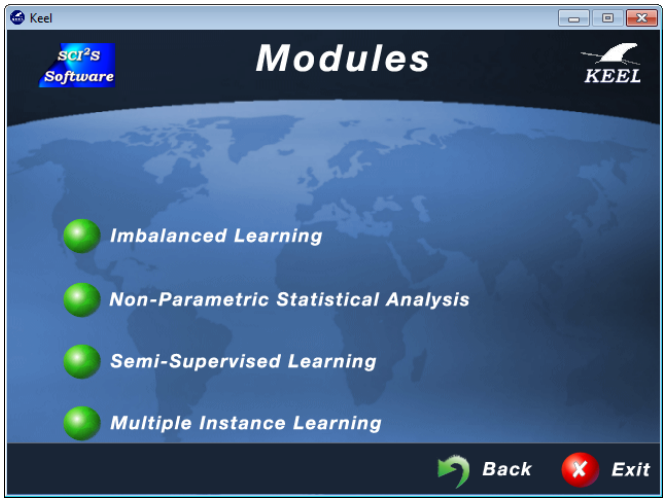

Figure A.2: The Modules Menu Section

- **Imbalanced Learning:** This part maintaining the identical composition and purposes of the experiments section including all the latest technology in Imbalanced Classification.
- **Non-Parametric Statistical Analysis:** This module presents several methods for pairwise and multiple comparisons, along with numerous procedures in raw CSV format and outputs printed in latex format.
- **Semi-Supervised Learning:** This module is dedicated to the production and design of experiments.
- **Multiple Instance Learning:** This section outline and build experiments for multi-instance Learning.

A.1.2 How to get KEEL

To download KEEL first of all go to the web page named "http:www.keel.es" and downloading the latest version of the software. Once the compressed file of KEEL has saved, just required to unzip all files into anywhere on the machine. Then, move into the "dist" folder and execute the "GraphInterKeel.jar" file.

A.1.3 Required Environment

Since "KEEL" is completely built in Java, indicates that any machine has installed and run a Java Virtual Machine (JVM) can able to run the KEEL software properly.

A.2 Basic Operation

A.2.1 Importing data

- Select the Data Management section.
- Then click the Import Data option.
- Select "Next" in the section Import Dataset.
- If the data format is CSV, definitely set the "Select Input Format" option as: "CSV to Keel".
- Select the input file and then press next.

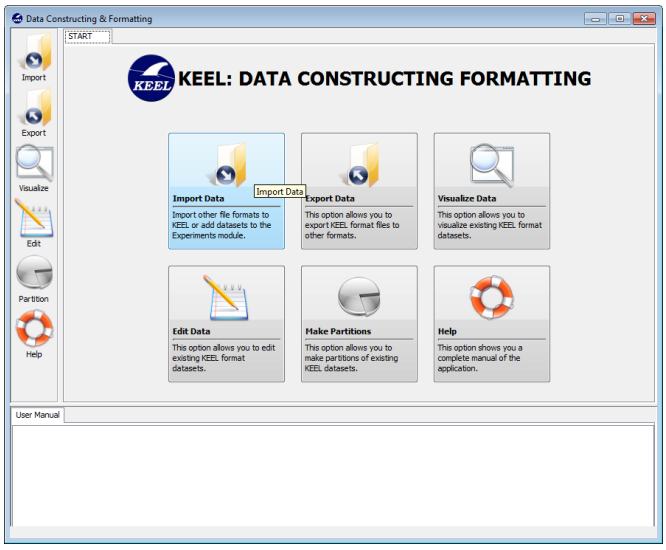

Figure A.3: Import data-set in KEEL software

For example, we have downloaded the "mammographic" data set from the KEEL dataset repository. To import the "mammographic" dataset into KEEL software the "Import Partitions" option will be selected shown in figure A.3, as we downloaded the dataset from KEEL which already partitioned in KEEL format. In the following image in figure A.4, we will choose the location for selecting the dataset and re-arrange according to their type i.e whether they are training or test files.

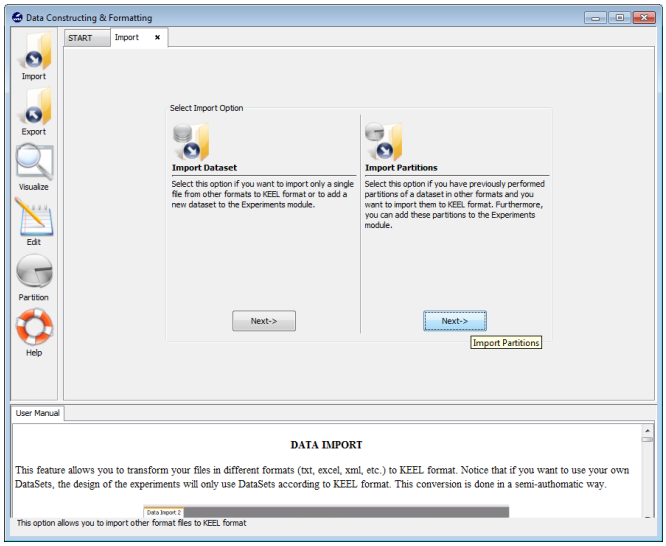

Figure A.4: Import partitions in KEEL software

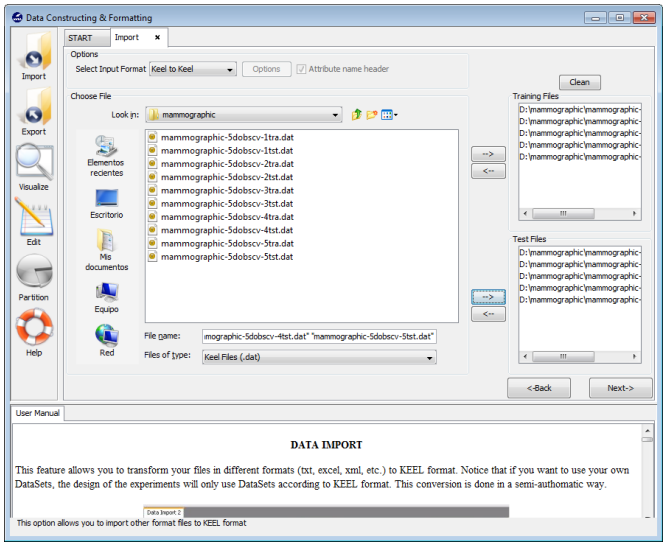

Figure A.5: Re-arranging the dataset

We are going to receive another confirmation window before adding our dataset to KEEL which has shown in figure A.6. In this dialogue box we need to mention the supplementary information regarding the dataset we are adding. After that clicking the save button will added the data set in KEEL data repository.

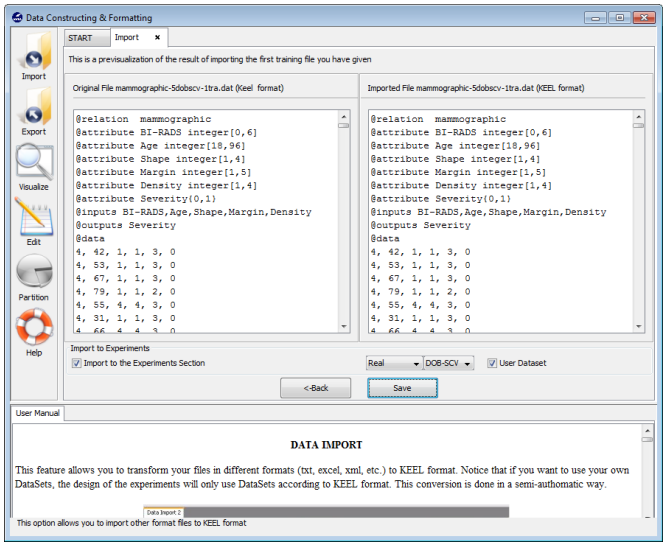

Figure A.6: Affixing the partitions to include the "mammographic" dataset

Now a new dialog box will prompt which will seek a name for the dataset figure A.7. We can any name we want and then in figure A.8 select the "Classification" option. Finally, the "mammographic" dataset is imported/saved successfully.

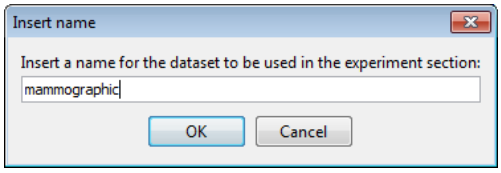

Figure A.7: Choosing a name for the "mammographic" dataset

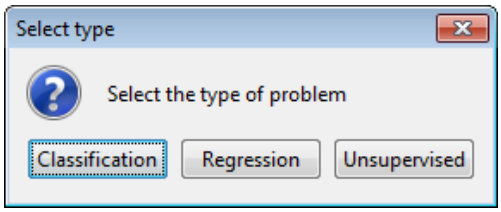

Figure A.8: Fixing the problem type for the "mammographic" dataset

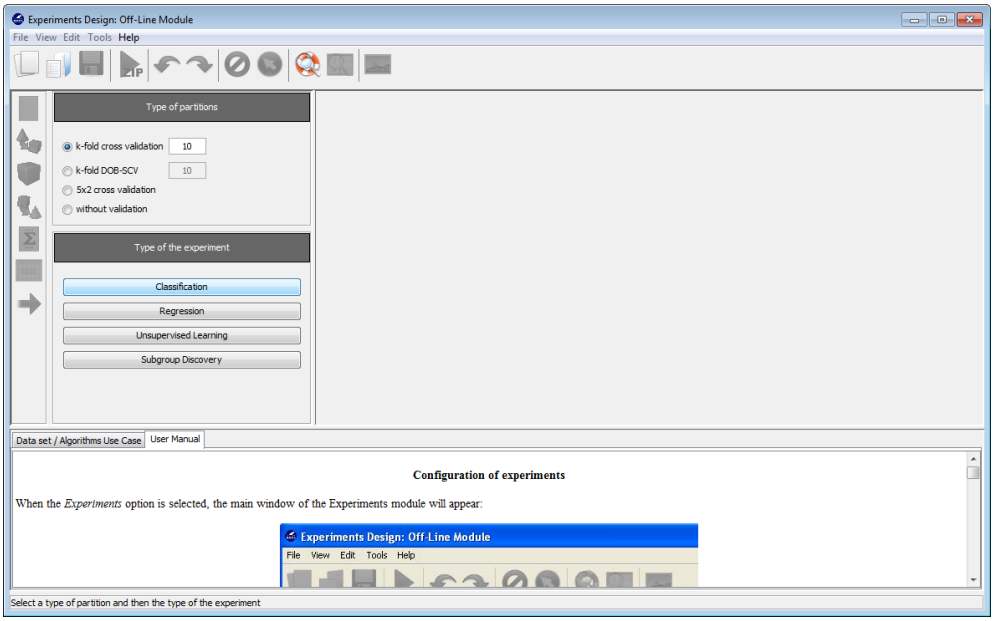

Figure A.9: The type of partitions and experiment

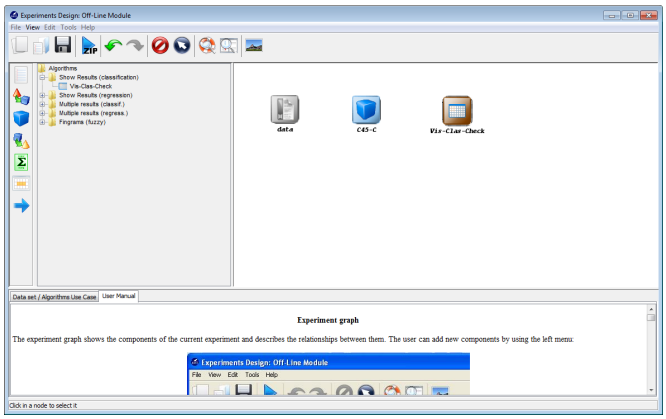

Figure A.10: Description of a sample experiment

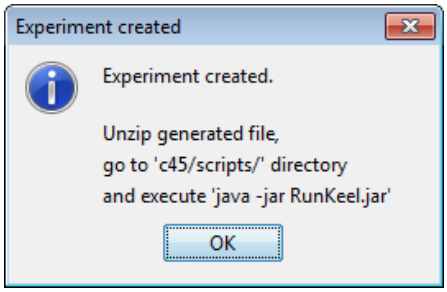

Figure A.11: Constructing experiment

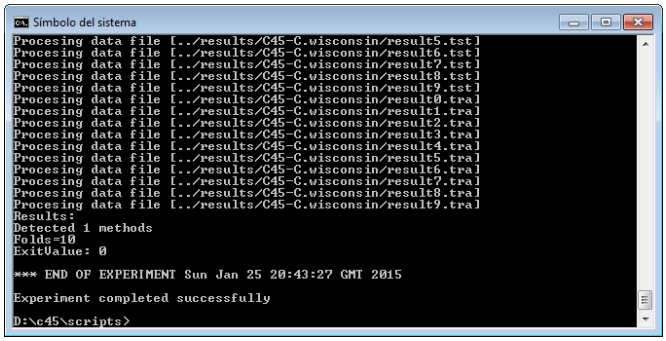

Figure A.12: Executing experiment

```
TEST RESULTS
============
Classifier= iris
Fold 0 : CORRECT=0.8666666666666667 N/C=0.0
Fold 1 : CORRECT=0.9333333333333333 N/C=0.0
Fold 2 : CORRECT=0.9333333333333333 N/C=0.0
Fold 3 : CORRECT=1.0 N/C=0.0
Fold 4 : CORRECT=1.0 N/C=0.0
Fold 5 : CORRECT=0.9333333333333333 N/C=0.0
Fold 6 : CORRECT=0.9333333333333333 N/C=0.0
Fold 7 : CORRECT=1.0 N/C=0.0
Fold 8 : CORRECT=1.0 N/C=0.0
Fold 9 : CORRECT=1.0 N/C=0.0
Global Classification Error + N/C:
0.039999999999999994
stddev Global Classification Error + N/C:
0.04422166387140534
Correctly classified:
0.96
Global N/C:
0.0

TRAIN RESULTS
============
Classifier= iris
Summary of data, Classifiers: iris
Fold 0 : CORRECT=0.9777777777777777 N/C=0.0
Fold 1 : CORRECT=0.9851851851851852 N/C=0.0
Fold 2 : CORRECT=0.9851851851851852 N/C=0.0
Fold 3 : CORRECT=0.9777777777777777 N/C=0.0
Fold 4 : CORRECT=0.9777777777777777 N/C=0.0
Fold 5 : CORRECT=0.9777777777777777 N/C=0.0
Fold 6 : CORRECT=0.9851851851851852 N/C=0.0
Fold 7 : CORRECT=0.9777777777777777 N/C=0.0
Fold 8 : CORRECT=0.9777777777777777 N/C=0.0
Fold 9 : CORRECT=0.9777777777777777 N/C=0.0
Global Classification Error + N/C:
0.02
stddev Global Classification Error + N/C:
0.0033945005147821075
Correctly classified:
0.98
Global N/C:
0.0
```

Figure A.13: Content of one the .stat output files