Dealing with classification irregularities in real-world scenarios

Payel Sadhukhan

Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
Kolkata - 700 108, India

Supervisor: Dr. Sarbani Palit

A thesis submitted to Indian Statistical Institute
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
1 July 2020

Dedicated

to

my family

&

late Prof. C. A. Murthy

ACKNOWLEDGEMENTS

From the beginning of my high school days, I wanted to have a research career. I was not quite sure about my exact passion but I knew that it had to be something logical, where a series of logical thoughts would lead me to something that I did not know or something that I knew but did not know why.

I had started my Ph.D in August 2014 under the supervision of late Prof. C. A. Murthy. He taught us during my M. Tech days at the same institute. His lecturers were immensely informative and probably one of the best in our country. Besides passing information to the students, he made them think about how things were working. I was influenced by him before joining my Ph.D. I had spent three years under his supervision till his death in March 2018. He taught me how to think, helped me with writing papers and rendered a fatherly supervision during this entire period.

The sudden death of Prof. Murthy had created a void which shook the very foundation of my Ph.D. Nonetheless, I was determined to continue and complete the unfinished ones and complete my thesis. In this tumultuous period, my present supervisor, Prof. Sarbani Palit stood by me as a rock, both academically and emotionally. With her help and supervision, I have been able to finish the works related to my Ph.D. To be honest, I don't know how to thank these two people. I can only say that without you two, I would not be writing the acknowledgement of this thesis.

My family has been my strength and anchor. My husband was there to support me from the beginning till the end of this work. My daughter was born during the early phase of my Ph.D and I could manage my home and work because of our joint efforts. My daughter has been a source of strength and cheer up and has helped me to preserve the very meaning of life during the pitfalls of my Ph.D journey. I also beg her pardon for missing out quite a few moments of her childhood while working towards my Ph.D (I sincerely hope that she understands my actions when she grows up). On an unofficial note, this Ph.D is the joint venture of all three of us, so I choose not to thank them. I would like to thank my parents because they have ensured me a strong academic foundation which enabled me to complete my higher studies.

I want to express my thankfulness to all the faculty members of my department. I would also like to acknowledge the support that I have received throughout from the office staff of our department during the tenure of my PhD. I express my sincere thanks to the authorities of ISI for the facilities extended to carry out research work and for providing me every support during this tenure. I want to thank all the others, whom I might have missed here, for their well wishes and support.

Payel Sadhukhan

List Of Related Articles Of The Author

- P. Sadhukhan and C. A. Murthy, "Multi-label Learning Through Minimum Spanning Tree-Based Subset Selection and Feature Extraction", Canadian Conference on AI, Lecture Notes in Computer Science, April 2017. *(Chapter 3)*

- P. Sadhukhan, "Learning Minority Class prior to Minority Oversampling", International Joint Conference on Neural Networks (IJCNN), July 2019. *(Chapter 2)*

- P. Sadhukhan and S. Palit , "Reverse-nearest neighborhood based oversampling for imbalanced, multi-label datasets", Pattern Recognition Letters, Volume 125, 2019. *(Chapter 4)*

- P. Sadhukhan and S. Palit, "Lattice and Imbalance Informed Multi-label Learning", IEEE Access, Volume 8, 2020. *(Chapter 3 and 4)*

- P. Sadhukhan, "Can Reverse Nearest Neighbors perceive unknowns?", IEEE Access, Volume 8, 2020. *(Chapter 5)*

- P. Sadhukhan and S. Palit, "Adaptive learning of minority class prior to minority oversampling", Pattern Recognition Letters, Volume 136, 2020. *(Chapter 2)*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data processing by the human sensory system comes naturally. This processing, commonly denoted as pattern recognition and analysis are carried out spontaneously by humans. In day to day life, in most cases, decision making by humans come without any conscious effort. From the middle of the past century, humans have shown interest to render their abstraction capabilities (pattern recognition and analysis) to the machine. The abstraction capability of the machine is 'machine intelligence' or 'machine learning' [87].

The primary goal of machine learning methods is to extract some meaningful information from the 'data'. Data refers to the information or attributes that are fed to a machine learning algorithm or method. The two main types of learning are – i] Summarization and ii] Generalization. An algorithm makes a summarization of the given information to understand the key components of the data. The algorithm might aim to learn the key features, key data point which can provide the particulars of the data. The other aspect of machine learning is generalization of data – extracting the underlying structure of the data to make correct prediction of upcoming new data. Classification of data by an algorithm requires it to make a generalization first.

Classification of objects into different categories is a fundamental element of decision making. The machine learning community has taken a keen interest in developing competent classification algorithms (classifiers) since it's outset. Thereupon, diversified paradigms like naive bayes [77], knn [65], neural networks [35] have emerged to facilitate effective classification of data. A classifier is modeled through the generalization of the given data or the training data. The purpose of a good classifier is to make correct predictions of the future data (or the test data) on the basis of the generated model. Consequently, the prediction capability of a classifier is dependent on how good one has modeled the classifier. The modeling in turn depends on the training data. To integrate and facilitate the intertwining of efficient modeling and predictions, some standard assumptions are made by the machine learning community.

Classification algorithms, being mathematical formulations are devised under several such assumptions on the datasets. The assumptions can be i] the different classes of a dataset have similar cardinalities (if the standard deviations in class cardinalities is high, we have the class-imbalance issue), ii] the training and test partitions of a dataset will have similar attribute profile (same number of classes and distributions), iii] an instance can belong to one class only, iv] information about all attributes of the instances are known and more. The continuous and intelligent efforts of the machine learning community has given (and is still providing) a class of robust classifiers in the past and present times. However, often, it is observed that a 'potent' classifier which gives accurate predictions under a given assumption fails to deliver optimal performance whenever it is exposed to situations where these assumptions are not fulfilled. To get admissible performance from the conventional classifiers, it is essential to comply with the assumptions.

Real-world data from diversified domains like medical, biology, security, banking, social networking, web-data, news articles show breaches of several of these assumptions. The weakness of the machine learning algorithm in failing to generalize becomes all the more clear when tried on the real word datasets. In current state of time, the involvement of machine learning in our day-to-day life has increased to an remarkable extent. To maintain and escalate its relevance, the classification paradigms have to be robust to the breach of the assumptions which exist in the real world data. In datasets from medical domain, a quantitative disparity exists in the cardinalities of different classes. A class may contain a significantly higher number of instances than another. Such a disproportion in the dataset biases the classifier towards the quantitatively abundant class. As a consequence, at the test phase, the instances from the under-represented class goes go vastly undetected by a regular classifier and we get suboptimal performance for that class.

The goal of this thesis is to address some of these data deviations in a classification context. In this thesis, we handle three such deviations and propose solutions to the problems. While analyzing the data deviations we have observed that at times two or more irregularities are intertwined with each other. In such a scenario, solving one gives an elegant solution to the other. In the following section, we discuss the motivation and the three types of deviation that we focus in this thesis. In the next subsection, Section

1.1, we discuss the problems that we address in this thesis. A more concise articulation of the research statements and research questions are presented in Section 1.2 .

1.1 Issues in Classification

As we have discussed above, data from a real-world domain has naturally occurring deviations or breaches of assumptions. Various types of deviation already exist and with time more such deviations are emerging from different fields. In this thesis, we have approached three types of deviations. These three deviations are – i] Class-imbalanced data, ii] Multi-label data and iii] Unknown class labels. We briefly discuss the technical aspects of these three deviations in the following paragraphs.

Class-imbalanced data: While modeling a conventional classifier for a given dataest, it is assumed that the cardinality of the different classes in the data are somewhat equal. A diverging scenario with largely varying class sizes is seen in data from domain like medical and fraud detection. In standard medical data [82], the ratio of the number of afflicted persons to that of healthy persons is highly skewed. We have greater number of healthy persons than diseased ones. A similar scenario is observed in fraud detection data in the banking sector [60]. The number of genuine transactions highly outnumber the fraudulent ones. Several other real-world domain are pertinent sources of such skewed or imbalanced data. A standard term for such data is "class-imbalanced data". In a class-imbalanced dataset, the number of examples in the possible classes differs by a noteworthy amount, affecting the functioning of a conventional classifier. In a two-class dataset, the quantitatively abundant and the under-represented classes are termed as majority class and minority class respectively. Conventional classifiers get biased towards the majority class of a class-imbalanced data. Though a good accuracy score is obtained on such data, such classifiers often fail to detect the minority class instances. In this thesis, we aim to achieve improvement in performance on the detection of both the majority class and the hard-to-learn minority instances.

Multi-label data: In a conventional classifier, the data points (examples) belong to exactly one of the possible classes. Data sources like a newspaper article can belong to one or more classes simultaneously. For example, a newspaper article (text categorization) can be classified as belonging to sports and entertainment together. Some other article may be classified as a piece on politics and entertainment. Similar classes of datasets are obtained from the domain of image, medical, tag recommendation systems and video. These data are characterized by the membership of their points in more than one overlapping classes. Such datasets are termed as **multi-label data**. The classes of a multi-label dataset are transformed into two or more overlapping labels. Each label represents a category and an instance is classified as 0 or 1 according to it's membership (belongingness) to a category. This process is repeated over the entire label set to get the information of the instance. Let us denote a multi-label dataset by $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{Y}_i), i = \mathit{1}, \mathit{2}, \ldots, n\}$ and the label set cardinality be $\mathcal{L}$. $\mathcal{Y}_i = \{y_{i1}, y_{i2}, \ldots, y_{iL}\}$. We assume that each label has exactly two classes positive (1) and negative (0) that is $\mathcal{Y}_{ij}$ can be either 1 or 0, $j = 1, 2, \ldots, \mathcal{L}$. The goal of the classifier is to rightfully classify $\mathbf{x}_i$ into either the positive (1) or the negative (0) class for $\mathcal{L}$ labels. In general, a conventional classification system does not provide for learning of multi-label data. This motivates a need for sophisticated classifiers which can accommodate multi-label learning. In this thesis, we analyze multi-label data and work on two aspects to provide competent multi-label learners.

Unknown class classification/ Open Set Recognition: Since the primal days of machine learning, a classifier is designed to efficiently classify the objects/ examples from classes that it has seen. Synchronizing with this aspect, a fascinating assumption is made. While modeling a classifier, it is assumed that the training data and test data has same number of classes. Technically, this assumption suggests that there is nothing unknown to the classifier. But this aspect does not conform to the characteristics of practical data. To develop an automated and self-sufficient system, we have to focus and fill up this critical gap. Besides regular (or known class classification), we have to facilitate unknown class detection. In machine learning context, Open set recognition/ classification is the detection of unknown (unseen during training) class instances and known class classification in an open world of known and unknown classes. Fraud detection [93], fingerprint spoof detection [71], impostor detection [31] constitute pertinent context for unknown class detection in real world domain. Open set classification is principally conducted in a multi-class framework. In the training or classifier modeling phase, we have one or more known classes on the basis of which the classifier is modeled. In the test phase, apart from the known classes seen during training, we encounter instances from some other classes unseen during training. The job of the classifier is to correctly classify the seen class instances to their rightful class and classify the unseen instances as belonging to the unknown class. In this thesis, we have presented a classifier which facilitates unknown class classification besides regular known class classification.

1.2 Thesis Contribution

In the previous section, we have presented a brief technical portrayal of three particular types of data irregularities – i] imbalanced data, ii] multi-label data and iii] open set classification. To obtain an efficacious performance on these classes of data, we need to provide special attention to them. These three types of data deviations are in three different directions and are largely different from each other in terms of technicalities. To have a potent learning and classification approach, we analyze the three types of data separately and work on the critical aspects of each of them. We have further noticed that multi-label data is by and large plagued by class-imbalance of the labels also. Consequently, we have added a study which deals with issue of class-imbalance in multi-label problems. Four specific research questions are raised and answered in the thesis. The specific research questions addressed in different sections of this thesis are presented in the following section.

1.2.1 Class-imbalance problem in a two-class dataset

In a class-imbalanced dataset, when we have exactly two classes, the class with higher and lower number of points are termed as majority and minority classes respectively. Class-imbalance ratio is ratio of points in the majority class to those of the minority class of a dataset. In a class-imbalanced dataset, the difference in cardinalities of the majority and minority classes are significant enough to affect the proper learning of such data. The minority instances get largely undetected in a regular classification framework.

This problem is handled by the machine learning community using various diversified techniques. The solutions to this problem can be classified into two types — i] data modification or ii] modifying the classifier to accommodate class-imbalance of data. The principal goal of the data modification techniques is to equate the cardinalities of the majority and the minority classes of the training data before building the classifier. There can be two ways of achieving this. The first way is to undersample the majority class– removing points from the majority class. The other way is to oversample or increase the minority class by adding synthetic minority points. Starting from SMOTE, the oversampling techniques have given incremental and commendable improvement in minority class performance over the years.

The objective of a class-imbalance focused learner is to improve the performance of the minority class instances. The techniques involving minority class oversampling have satisfied this criteria to some extent, often at the expense of majority class results. These learners shift the decision boundary away from the minority class instances and towards the majority class instances. This results in the detection of borderline minority instances but the majority class instances in the same region get misclassified into the minority class. Consequently, the classifiers focused towards minority class improve the F_1 scores of the minority class but the accuracy (taken together for both the classes) goes down in most of the cases. Accuracy goes down because the majority class instances are more in number. Yet, the betterment in minority class performance cannot be an affirmation to the true classes of the synthetic instances. This aspect raised the following question.

- **How to generate a synthetic minority point at the rightful location?**
- **Our Answer :** Estimate the minority set from the given points and generate the synthetic minority points from the estimated space.

We propose two schema for estimating the minority spaces. The first scheme proposes an estimation procedure where the volumes of estimated spaces are constant around the existing minority points. The second scheme has a layer of refinement over the first as it estimates an adaptive volume of space around different minority points. In both cases, the synthetic minority points are generated from the estimated spaces. We have carried out our experiments on more than 20 real-world class-imbalanced datasets. We have tried the oversampled datasets from the proposed and state-of-the-art methods on two base classifiers – C4.5 decision tree and logistic regression based classifier. Our empirical study shows that the proposed set estimation prior to generation model gives comparable to superior performance over the existing minority oversamplers.

1.2.2 Learning multi-label datasets — from the perspective of feature extraction

In a multi-label dataset, an instance belongs to more than one overlapping class. If the label set cardinality is $\mathcal{L}$, each instance has $\mathcal{L}$ membership values, one for each label. In a two class multi-label dataset of $\mathcal{L}$ labels, for an instance, each label membership can take exactly one of two values – 0 or 1. 0 and 1 denote negative and positive memberships respectively. In a multi-label dataset, the set of feature points are shared by all labels. Despite sharing the feature points, the memberships of the instances vary from label to label. This leads to a variable class partition and geometries for different labels. A multi-label dataset has exactly one instance set with different positive and negative class partitions for different labels. In this scenario, simply modeling a set of regular classifiers, one for each label will not provide for learning.

- **Question: How to properly learn a multi-label dataset?**
- **Answer:** Extract the label-specific class geometries of a multi-label dataset.

A meaningful learning of data characteristics is required in the context of each label. In order to achieve that, we learn the positive and negative lattice points for each label. Lattice points are key structural points of a dataset – we use them to learn the underlying class boundary of the dataset. The class-partition of a dataset varies across labels for a multi-label dataset, hence the set of lattice points will vary across the labels. We use two separate techniques for selecting the lattice points among the given feature points/ examples. One involves Minimum Spanning Tree and the other is based on Relative Neighborhood Graph. Finally, the lattice points obtained from the graphs are used to extract a label-specific feature set, one for each label. A single representation (of each instance) for all labels is transformed to a set of feature mappings, one for each label. Our experimental study has affirmed the utility of label-dedicated feature set in learning the multi-label datasets. We have employed 10 real-world datasets and five existing multi-label learners in the comparative study of our work.

1.2.3 Multi-label dataset: From the perspective of class imbalance

Another prominent characteristic of multi-label datasets is the class-imbalance of the labels. This is more compounded with the variable class partition across the labels. In a multi-label dataset, varying class partitions lead to varying number of points in the positive and negative classes of different labels. As a consequence, we get differing class imbalance ratio values for different labels.

- **Question: What are the ways of handling differential degrees of class-imbalance in a multi-label dataset?**
- **Answer 1:** Set varying costs of misclassification of minority instances for different labels.
- **Answer 2:** We can employ an imbalance adaptive oversampling for different labels.

In this piece of work, we propose two solutions to the critical problem of varying imbalance ratios. The first solution works on the aspect of modifying the classifier. A standard solution for handling class-imbalance problem is cost-sensitive learning. To counter the bias of a regular classifier towards the quantitatively abundant majority class, the misclassification cost of the minority instances is increased as compared to that of the majority instances. We extend this paradigm in context of multi-label datasets. We calculate the imbalance ratios of different labels. We select a set of misclassification cost values of the minority class, one for each label. This cost is made adaptive to the imbalance ratios. Between two labels with differing degree of imbalance, we set a higher misclassification penalty for the one with higher imbalance than that of the other.

In our second solution, we propose an imbalance-adaptive oversampling of the multi-label dataset. As we have said earlier, the minority-majority class configuration of the points and the imbalance ratios vary from label to label. To counter these two problems, we select different number of synthetic minority points for different labels. We add more synthetic minority points to a label with a higher imbalance ratio than that of another with a lower value. To select the locations of oversampling, we use the principles of reverse-nearest neighborhood. The neighborhood estimation given by reverse-nearest neighbor principles is adaptive to the density and distribution of the points. This takes care of the differential partitions of the positive and negative classes while generating the synthetic minority points. Empirical study indicates the contribution of the proposed techniques in handling class-imbalance issue multi-label learning and rendering admissible solutions.

1.2.4 Open set classification

Open set classification calls for the classification of seen (known) instances as well as the class of instances which are not seen (unknown and not used during training) by the classifier. A common mistake made by a conventional classifier is to mis-classify the unknown instances into one of the known classes. In such a scenario, it is required that the classifier should know what it does not know. To design a good open set classifier, one needs to balance and do well in both unknown class detection and known class classification.

- **Question: How to handle open set classification in an elegant fashion?**
- **Answer**: The classifier should have room for an open answer. By "open answer" we mean that – the classifier should have an option to say "I don't know" or "not among the ones seen" when encountered with unknown instances. We propose a reverse-nearest neighborhood based solution for dealing with open set classification.

We propose a scheme which simultaneously perform known class classification and unknown class detection. In the training phase, we will have instances from a certain number of known classes (seen during training). Let that number be c and let us have instances from these particular c classes only at training. During the test phase, the classifier can encounter instances from these c classes or from some other class/es also to which the classifier was not exposed in the training phase. We denote the unknown class by c+1. We note that there can be more than one unseen class, which we should not know because technically that information should be unknown also. Hence, we consider exactly one unknown class which we consider as $(c + 1)$. Hence, at training we do not have instances from $(c+1)^{th}$ class. The duty of the classifier is to correctly classify the instances from the seen c classes along with the detection of instances belonging to the unknown classes.

Following principles of reverse k-nearest neighborhood (RkNN), the neighbors of $\mathbf{p}$ are those points in whose nearest neighborhood $\mathbf{p}$ lies. It is easy to note that in a search space with cardinality n, the reverse nearest neighbor count of $\mathbf{p}$ can range from 0 to n. Unlike kNN, RkNN possesses an intrinsic capability to handle unknown class detection through it's zero neighbor count. The non-zero neighbor count of 1 to n can accommodate the known class classifications. The zero neighborhood property of RkNN gives the backbone for unknown class detection. We integrate two pieces of information – i] number of reverse nearest neighbors and ii] the distance of the nearest reverse neighbor (if at all present) to predict the class information of a test instance. Empirical study over 10 datasets and five competing methods indicate the capability of the proposed technique to address open set classification.

1.3 Organization

The rest of this thesis is organized as follows.

- Chapter 2 deals with the class-imbalance problem of machine learning in a generic context. We suggest that looking at the density and distribution prior to synthetic minority point generation can provide a better dataset for learning the minority class.

- In Chapter 3, we deal with multi-label datasets. We specifically work on the feature extraction perspective and suggest two schemes for extracting label-dedicated feature sets.

- Chapter 4 is also dedicated towards an effective learning of multi-label datasets. Here, we study the class-imbalance problem in multi-label datasets and propose two distinct solutions to this issue.

- We discuss the problem of open set classification in Chapter 5. An elegant solution to this problem is derived from the principles of reverse-nearest neighborhood.
- Chapter 6 concludes this thesis. We highlight our key observations and present the directions of future study.

Chapter 2

Class imbalance handling through estimation of minority class

2.1 Introduction

In this work, we address the class-imbalance problem prevailing in data analysis on the principles of set estimation. Learning a model of the minority class instances of an imbalanced dataset is often a difficult proposition. The minority class is the class with the least share of instances. The relative disparity in the instance shares of the minority class and its conjugate majority class biases a conventional classifier towards the quantitatively abundant majority class — minority instances mostly get undetected in such a scenario. Practical applications of class-imbalance learning are numerous and significant in present day context. Automated medical investigation [70], [57], customer classification [97], remote sensing [6], oil spill detection [46], fraud detection in banking and telephone calls [93] [22], text mining [113], [51], customer churn prediction [8], software defect prediction [39] constitute some relevant sources of class-imbalanced data.

So far, diversified techniques have been applied in the field of class imbalance learning approaches. The primary objective of a class imbalance classifier is to obtain better classification results for the minority class without perturbing majority class performance as much as possible. Several methods in the recent past have focused on generating synthetic minority samples from the information of the given minority instances. For generating the synthetic minority instances, the existing methods follow a general outline – they consider the original minority instances and their minority neighborhood.

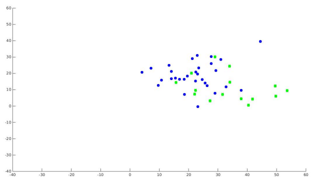
Starting with SMOTE [12], such techniques involving synthetic minority instance generation namely ADASYN [33], Borderline-SMOTE [32], NCN-SMOTE [26], MWMOTE [3] have given incredible improvement in minority class performance, which is the ultimate goal. Yet betterment in minority class performance cannot be an affirmation of the correctness of the class information of the synthetic data. Now the question is whether the minority instances are generated at their correct locations? So far, there is no clear answer given in the domain of class-imbalance learning and we address this in our work. We have used a Voronoi diagram to analyze the locations of synthetic minority points. Voronoi diagram of a set of objects is the decomposition of the space into a cell per object, such that cell associated with an object consists of all points that are closer to the object than any other given object. With reference to **Figure 2.1**, we have seen that the synthetic minority points generated by the existing oversamplers often lie within the Voronoi cell of an original majority point. This indicates that such synthetic minority points are encroaching the majority class spaces.

Our work is motivated to address this aspect by minority set estimation. A judicious estimation of the minority spaces and oversampling the minority points from those spaces will result in a sound synthetic minority set. It is important to note that minority instances alone are not sufficient to define the minority class spaces. Mutual separation of the majority and minority classes can help us define the minority spaces. We estimate the minority class spaces in light of this information. In this research work, we propose two procedures for estimating the minority spaces. The synthetic minority samples are generated from the estimated minority spaces. The original set of instances and the synthetic minority set is employed in class-imbalance learning.

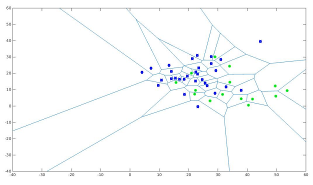
We have evaluated the competence of the proposed techniques, over five related methods namely SMOTE, ADASYN, NCN-SMOTE, Cluster-SMOTE and MWMOTE through empirical evaluations on 23 real-life datasets on two diversified metrics, AUC and minority class F_1.

2.2 Review of Existing Class Imbalance classifiers

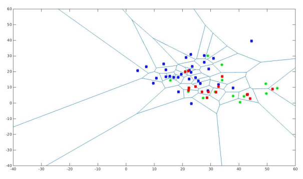
The class-imbalance problem has intrigued the data science community for more than a decade and substantial developments have been made in this domain. One approach concentrates on modifying the classifier for detecting those minority instances which remain unsighted by the regular classifiers. Cost sensitive learning was one of the foremost approaches in this domain. In this learning paradigm, the misclassification costs for different classes are set according to their shares of instances and importance. The main



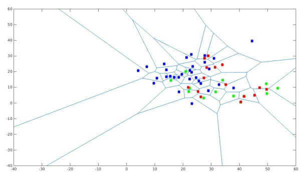
(a) Original Points



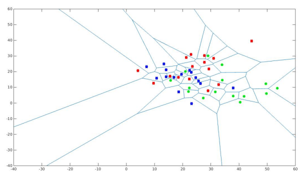
(b) Voronoi Diagram of the points



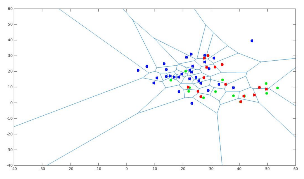
(c) SMOTE



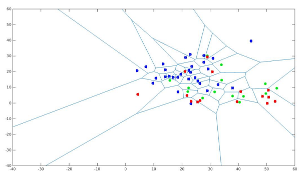
(d) ADASYN



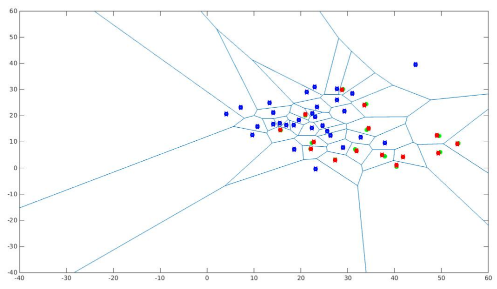
(e) Cluster-SMOTE



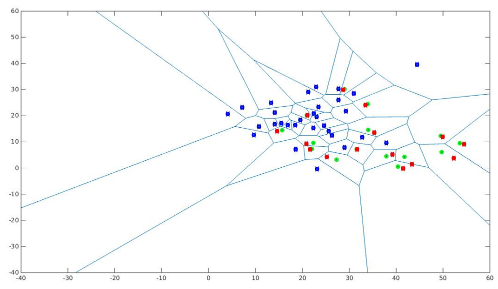
(f) MWMOTE



(g) NCN-SMOTE



(h) LMCMO



(i) ALMCMO

Figure 2.1: Locations of synthetic minority points generated by various methods in Voronoi diagram. Fig (a) illustrates a synthetic dataset, consisting of points from two classes. The green and blue points represent minority and majority points respectively. 50 majority points come from a Gaussian distribution with $\mu = [20, 20]$ and $\Sigma = \begin{pmatrix} 50 & 0 \\ 0 & 50 \end{pmatrix}$ respectively. 10 minority points belong to a Gaussian distribution with $\mu = [30, 10]$ and $\Sigma = \begin{pmatrix} 500 & 0 \\ 0 & 500 \end{pmatrix}$. Fig (b) shows the Voronoi diagram for the given set of points. Voronoi diagram of a set of objects is the decomposition of the space into a cell per object, such that cell associated with a object consists of all points that are closer to the object than any other given object. In our case, the set of majority and minority points taken together constitute the set of objects. In Figures (c)-(i), we show the locations of the synthetic minority points (represented by red) generated by the methods. The name of the generating algorithm is indicated in the corresponding figure. It is easy to note that a number of synthetic minority points generated between figures (c)-(g) are lying in the Voronoi cells of the majority class. Figures (h) and (i) illustrate the synthetic minority points generated by LMCMO and ALMCMO. In these two cases, the synthetic minority points lie within the minority Voronoi cells only. In LMCMO, the synthetic minority points are concentrated around the original minority point. But, in ALMCMO, the synthetic minority points are distributed in the entire Voronoi cell. **The minority points generated by LMCMO and ALMCMO maintain the non-encroachment into the majority class spaces (majority Voronoi cells) for all datasets.**

goal is to bias the classifier towards identifying the minority samples. Cost-sensitive neural networks, and cost-sensitive decision trees are two principal variants. Two other important variants are Active learning and Kernel-based methods. Querying the dataset to yield a balanced training set for learning the classifier through active learning has been done to address the class-imbalance issue [21]. In Kernel-based methods, the original dimensions are mapped to other dimensions better separating the instances to improve the minority class learning. A few examples are class-boundary alignment [94], kernel target alignment [95] and margin calibration [99].

The other way of achieving fair recognition of the minority class of a class-imbalanced dataset is to modify the dataset to equate the majority and minority class cardinalities. This can be realized in either way — through under-sampling of the majority class or by oversampling the minority class . Both under-sampling or oversampling can also be involved. A naive way has been to perform random under-sampling of the majority class instances or random duplication of the minority class instances [48]. Tomek link information is employed to undersample the majority class in a number of works [47], [18]. Cluster-based undersampling has been implemented in [100]. The other way is to increase the number of minority points by oversampling the minority class.

SMOTE came as a breakthrough in the domain of minority oversampling. It generated synthetic minority samples using minority neighbourhood information of the minority points. Variants in this regard have been borderline-SMOTE, ADASYN, RAMOBoost [13]. ADASYN presents a similar scheme with a different number of synthetic points being generated at different locations according to the minority densities. NCN-SMOTE is another small but significant increment in this regard. It identifies the symmetric centroid neighbourhood of each minority point and generates the synthetic points from those locations. A recent method MWMOTE, aims to learn the hard-to-learn minority samples through clustering of minority instances.

Apart from the approaches discussed above, mixed or hybrid paradigms have also been employed to facilitate class-imbalance learning in recent years. One example is [36] which uses quintuplet sampling followed by deep learning to recognize minority instances. Class-imbalance learning is also attempted through ensemble classifiers [25].

2.3 Principles of Set Estimation and estimating the Minority Set

In the first part of this section, we elaborate on some parameters for estimating a set from a given number of finite points. Next, in **Section 2.3.1** and **Section 2.3.2**, we propose two techniques for estimating the minority class.

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be the given collection of points belonging to a pattern class in R^M, where $M \geq 2$. Let the probability density function for the class be represented by f. Our objective is to estimate the set from this information. To be precise we want to get the *support* of f, namely α.

Suppose the estimated set is α_n. We will estimate it from the set of instances $\mathbf{x}_1, .., \mathbf{x}_n$.

For a good α_n and a given α, the set inconsistencies between the two should approach to zero and the symmetric difference between the two should be as less possible. The estimator α_n should converge to the original set α as the instance set cardinality approaches infinity [31].

2.3.1 Choosing an estimator α_n which has constant volume hyperspheres across all points

We want to estimate α_n corresponding to the set, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.

$$\alpha_n = \bigcup_{i=1}^{n} \{y \in R^M : \|\mathbf{x}_i - y\| \leq \varepsilon\} \tag{2.1}$$

The above defined α_n set defines a space around each $\mathbf{x}_i$, $i = 1, 2, \ldots, n$. It is the union of hyper-spheres of radius ε centered at $\mathbf{x}_i$, $i = 1, 2, \ldots, n$. When the ε value is 0, we get the set $\{\mathbf{x}_i, i = 1, 2, \ldots, n\}$ itself as the estimated set α_n. With increasing ε value, we include a larger volume of neighborhood into the estimated space.

We have to select a suitable ε that would estimate the minority set from the mixed space of minority and majority instances. A too small ε value will add redundant synthetic minority points in the later stage while a large value will cause the estimated minority set to encroach upon the majority class. The procedure for selection of ε value is discussed next.

2.3.1.1 Choosing a suitable ε

While estimating the minority set from the mixed space of majority and minority instances, one important characteristic should be to preserve the boundary of the two classes. Our ultimate goal is to bridge the gap between the cardinalities of the majority and the minority class but not at the cost of encroachment into the majority class region. We should not over-estimate the minority class into the majority class space. We would like to insist that while proposing an estimator for the minority class we have not delved into the consistency of the estimated set. We cannot really claim that we propose a consistent estimator of the minority class. As a result, in cases where we have a small dataset, we do not have adequate information to formulate a firm boundary separating the two classes. In such a case, it is difficult to assign a region to either the minority class or the majority class. Under the given circumstances, using our scheme, we may land up with an over-estimation or an under-estimation of the minority class.

Minimum Spanning Tree (MST) is an efficient (though computationally intensive) tool for studying the shape and the structure of a set of instances. MST edges show the connection of the instances to their respective neighborhood. Homogeneous edges or the edges between two instances from the same class show the connected or continuous regions of class-membership. On a similar note, a heterogeneous edge signifies a switching or transition region from one class to the other. The heterogeneous edge weights give a quantitative idea of the separation of the two classes. Minimum heterogeneous edge weight gives the minimum separation between the two classes.

As stated in the above section, at ε value 0, the estimated minority set is the set of minority points only. A non-zero ε ($\varepsilon > 0$) includes an adjoining neighborhood of the minority points into the estimated minority set. The volume of the estimated minority set increases with increasing ε value. Let $\mathrm{d_{min}}$ be the minimum heterogeneous edge weight of the MST. Since $\mathrm{d_{min}}$ is the minimum heterogeneous edge weight, the majority points are at least $\mathrm{d_{min}}$ distance away from the the set of minority instances. For $\varepsilon >= \mathrm{d_{min}}$, the estimated minority set will encroach upon the neighborhood of at least one majority point. ε value should be strictly less than $\mathrm{d_{min}}$ to prevent overestimation into the majority class. ε just less than $\mathrm{d_{min}}$ gives an over-the-fence estimation of the minority class where we include the entire heterogeneous edge length into the minority class hypersphere (and shadows the majority class spaces). $\varepsilon = \mathrm{d_{min}}/2$ is a fair choice as it allows to include exactly half of the edge length instead.

Hence, we select, $\varepsilon = \mathrm{d_{min}}/2$ and use this in (2.1) to estimate the minority set.

General remarks:

- *Minimum Spanning Tree construction is not required for calculating the proposed ε value*: Though the proposed ε calculation is theoretically motivated by Minimum Spanning Tree (MST) construction, MST implementation is not required for finding the minimal heterogeneous edge weight. Finding the minimum distance between the instances of the majority class and minority class is sufficient for calculating ε. Since MST is a connected graph with minimal edge weight sum, the minority and majority instances have to be connected by the least distance between the two (minority and majority) sets. Not creating a MST adds on to the improved time complexity as well.

- *Choices of* ε: In the previous paragraph, we have proposed a choice for ε, in light of which this scheme is presented. A number of other solutions can also be possible in this regard. Construction of the Minimum Spanning Tree of the entire training set followed by average edge weight calculations of the heterogeneous edges with *ε=0.5* average heterogeneous edge weight* is another possible solution. Unlike the previous choice, this would require the creation of a Minimum Spanning Tree and hence has added time complexity.

2.3.2 Choosing an α_n which has variable volumes of estimated spaces across $\mathbf{x}_i$, $i = 1, 2, \ldots, n$

In the above defined α_n, the estimated space around each given point is same for the entire set of points. It is equal to the volume of a hypersphere of radius ε. The distribution and density of the points vary from location to location, it may be useful to vary the estimated space for different points. Hence, we want to vary this estimated volume across the point set by selecting a specific radius for each point. For $\mathbf{x}_i$, we select a specific radius $\varepsilon_i,\ i = 1, 2, \ldots, n$. This adaptive estimation bears some relevance to our task of minority set estimation and consequent oversampling. We present the details in the next subsection. We re-define the estimated set α_n as $\bar{\alpha_n}$ with a varying ε as follows.

$$\bar{\alpha_n} = \bigcup_{i=1}^{n} \{\mathbf{y} \in R^M : \|\mathbf{x}_i - \mathbf{y}\| \leq \varepsilon_i\} \tag{2.2}$$

ε_i is a dedicated value for instance $\mathbf{x}_i$. $\bar{\alpha_n}$ is union of hyperspheres of variable volumes around the minority points. Let $\varepsilon_j > \varepsilon_k$ for two points $\mathbf{x}_j$ and $\mathbf{x}_k$. $\mathbf{x}_j$ spans a larger volume of estimated space than that of $\mathbf{x}_k$ by virtue of it's larger radius. The motivation and protocol for selection of ε_i is discussed in the following subsection.

2.3.2.1 Choosing a suitable set of ε_i

A properly estimated minority set (in the mixed space of majority and minority instances) should adhere to the boundary of the two classes. If the estimated minority set encroaches on the majority classes, it will be an over-estimation (which is undesirable).

In the first estimation procedure, we have presented a Minimum Spanning Tree based estimation of the minority class. We have constructed a Minimum Spanning Tree (MST) of the entire point set consisting of majority and minority points. Let $\mathrm{d}_{\min}$ be the minimum heterogeneous edge weight of the MST. We have selected ε equal to half of $\mathrm{d}_{\min}$. We had a constant radius for all minority points and we have selected blobs of constant volumes around all minority points.

Depending on the dataset, it may be appropriate to have a varying volume of estimated space across different minority points. A minority point which has it's neighboring point at a farther location can span a larger volume of estimated space than that of another with a denser neighborhood. In this dissertation, we design our estimation procedure in this light. Our estimation procedure is such that we select a varying volume across different minority points. Our goal is to compute a dedicated radius ε_i for each minority point $\mathbf{x}_i$. We first construct a Relative Neighborhood Graph (RNG) from the training points where the edge-weight between two points is represented by the euclidean distance between them. RNG shows the accessibility and connectivity of a data point or vertex to its adjoining neighborhood and gives an overall structure of the feature points. A RNG of a given set of points is an undirected graph where any two points $\mathbf{a}_i$ and $\mathbf{a}_j$ are connected whenever there is no third point $\mathbf{a}_k$ such that $\mathrm{d}(\mathbf{a}_i, \mathbf{a}_k) < \mathrm{d}(\mathbf{a}_i, \mathbf{a}_j)$ and $\mathrm{d}(\mathbf{a}_j, \mathbf{a}_k) < \mathrm{d}(\mathbf{a}_i, \mathbf{a}_j)$. For a given set of points, it's MST is a subgraph of it's RNG.

A non-zero ε_i ($\varepsilon_i > 0$) includes an adjoining neighborhood of the minority point $\mathbf{x}_i$ into the estimated minority set. The volume of estimated minority set around $\mathbf{x}_i$ increases with increasing ε_i value. Let e_i be the minimum edge weight of instance $\mathbf{x}_i$ in the Relative Neighborhood Graph (RNG). In the RNG, $\mathbf{x}_i's$ nearest neighbor can be a majority point. If so, the nearest majority point is at least e_i distance away from minority instance $\mathbf{x}_i$. If we select $\varepsilon_i >= e_i$, the estimated minority set corresponding to $\mathbf{x}_i$ can encroach upon the neighborhood of that majority point. Value of ε_i should be strictly less than e_i to prevent overestimation into the majority class spaces. The ε_i just less than e_i gives an over-the-fence estimation of the minority class where we include the entire edge length into the minority class hypersphere (and shadows the majority class spaces). $\varepsilon_i = e_i/2$ is a fair choice as it allows to include exactly half of the edge length into the estimated minority spaces. Hence we select ε_i as follows.

$$\varepsilon_i = e_i/2 \tag{2.3}$$

Additionally, this protocol allows us to select a varying radius across different points. Let us have two points $\mathbf{x}_i$ and $\mathbf{x}_j$ with minimum edge weights e_i and e_j respectively such that $(e_i > e_j)$. The nearest neighbor of $\mathbf{x}_i$ is at a greater distance than that of $\mathbf{x}_j$. Hence, $\mathbf{x}_i$ has a greater volume of neighborhood at its disposal than that of $\mathbf{x}_j$. Consequently, the estimated volume of minority space will be larger around $\mathbf{x}_i$ than that of $\mathbf{x}_j$. The ε_i selection protocol in Equation (2.3) also reflects the same.

Essential Remark: Outlier minority points can give negative results on the above given scheme of ε_i selection. They can encompass a large volume of feature space as the estimated minority space. In order to prevent that, we detect singleton minority points with just one edge in the RNG as outlier and do not let them participate in the estimation and subsequent oversampling. In a highly imbalanced and small sized dataset, we will have very few minority points. In this case, it is quite possible that a good fraction of minority points are having just one-edge and are seemingly outliers. In such a scenario, we suggest to construct the RNG with $k > 1$, till an admissible result is achieved.

2.3.3 Synthetic minority sample generation from estimated minority set

Through minority set estimation, we transform the discrete minority point set to a set of continuous blobs around the minority points. A blob is a hypersphere of a specified radius around each minority point. The estimated minority sets α_n and $\bar{\alpha_n}$ are the union of all such possible blobs respectively. Each blob has an infinite number of minority points (considering real feature space). We perform random and uniform sampling of each blob to generate elements of the synthetic minority set. By *sampling*, we refer to selection of individual observations. By uniform sampling we mean that the sampling is not biased towards a blob of any particular minority instance and random refers to the selection of any random point of the blob or the hypersphere. We propose two schema for generating the synthetic minority instances – the first one is from α_n and the second one is from $\bar{\alpha_n}$.

2.4 The Proposed method

Given an instance set $\mathbf{D}$, let us partition it randomly into a training set, $\mathbf{D}_{\text{tr}}$ and a test set $\mathbf{D}_{\text{te}}$. LMCMO consists of two steps, minority set estimation followed by selection of minority instances set $\mathbf{S}_{\text{minor}}$ from the estimated set. Augmented training set $\mathbf{A}_{\text{tr}}$ is obtained by combining the original training set $\mathbf{D}_{\text{tr}}$ and the synthetic minority set $\mathbf{S}_{\text{minor}}$. Learning is wrapped up by mandatory classifier modeling from the augmented training set $\mathbf{A}_{\text{tr}}$ and classification of test set $\mathbf{D}_{\text{te}}$.

2.4.1 First step: Minority set estimation:

Estimation of constant volume around all minority points: $\mathbf{D}_{\text{tr}}$ is the input of this stage and we want to compute ε as the output. $\mathbf{D}_{\text{tr}}$ consists of the class information of instances also. Initially, we partition $\mathbf{D}_{\text{tr}}$ into the majority instance set, $\mathbf{O}_{\text{major}}$ and minority instance set, $\mathbf{O}_{\text{minor}}$, according to their class labels. For this work, we have prefixed ε equal to $\text{d}_{\text{min}}/2$. For calculating d_{min} implementation of the Minimum Spanning Tree is not required. Calculating the minimum distance between the members of $\mathbf{O}_{\text{major}}$ and the members of $\mathbf{O}_{\text{minor}}$ gives d_{min}.

Now, $\varepsilon = \text{d}_{\text{min}}/2$.

ε and $\mathbf{O}_{\text{minor}}$ together defines the precise locations of minority set. The estimated minority set is the union of the hyperspheres of radius ε around each member of $\mathbf{O}_{\text{minor}}$.

Algorithm 1 Minority Set Estimation: constant volume across all minority points

```
Input: Training set $\mathbf{D}_{\text{tr}}$
Output: $\varepsilon$
1: Partition $\mathbf{D}_{\text{tr}}$ into $\mathbf{O}_{\text{minor}}$, $\mathbf{O}_{\text{major}}$ acc. to class labels.
2: Calculate the set of distances
       $\text{D}=\{distance(i,j) : i \in \mathbf{O}_{\text{major}}, j \in \mathbf{O}_{\text{minor}}\}$
3: Calculate $\text{d}_{\text{min}}$=minimum value of D
4: Calculate $\varepsilon = \text{d}_{\text{min}}/2$
5: Construct a hypersphere of radius $\varepsilon$ around each minority point of $\mathbf{O}_{\text{minor}}$. Estimated minority
   set is the union of all such blobs.
```

Adaptive estimation of minority spaces around the minority points:

$\mathbf{D}_{\text{tr}}$ consisting of the original training points is the only input of this stage. We will compute the set of a dedicated radii ε_i for a non-singleton minority point $\mathbf{x}_i$ as the output. $\mathbf{D}_{\text{tr}}$ consists of the class information of instances also. Initially, we partition $\mathbf{D}_{\text{tr}}$ into the majority instance set, $\mathbf{O}_{\text{major}}$ and minority instance set, $\mathbf{O}_{\text{minor}}$, according to their class labels. In ALMCMO, we will generate a relative neighborhood graph (RNG) from the majority and minority instances of $\mathbf{D}_{\text{tr}}$. From the RNG, we will compute a radius for each minority point of the RNG. We will ignore singleton minority points with exactly one edge. The reason for this is stated in the remarks of previous section. For each minority point $\mathbf{x}_i$, we will find and store it's minimum edge weight as e_i. The dedicated radius corresponding to x_i is denoted by ε_i and it's value is selected as $e_i/2$.

Now, $\varepsilon_i = e_i/2$.

ε_i and $\mathbf{x}_i$ together defines estimated minority space (an hypersphere) around $\mathbf{x}_i$. The estimated minority set is the union of the variable volumes of hyperspheres of radius ε_i around each non-singleton minority instance $\mathbf{x}_i$.

Algorithm 2 Minority Set Estimation: volume adaptive to the local configuration of a point

```
Input: Training set $\mathbf{D}_{\text{tr}}$
Output: $\varepsilon_i$ for each non-singleton minority instance $\mathbf{x}_i$
1: Partition $\mathbf{D}_{\text{tr}}$ into $\mathbf{O}_{\text{minor}}$, $\mathbf{O}_{\text{major}}$ acc. to class labels.
2: Construct the Relative Neighborhood Graph (RNG) of the entire training set $\mathbf{D}_{\text{tr}}$
       Let $\mathcal{G}$=RNG($\mathbf{D}_{\text{tr}}$)
3: $e_i$=Minimum edge weight of $\mathbf{x}_i$, $\mathbf{x}_i \in \mathbf{O}_{\text{minor}}$ and $\mathbf{x}_i$ has more than one edge in $\mathcal{G}$
4: Calculate $\varepsilon_i = e_i/2$
5: Construct a hypersphere of radius $\varepsilon_i$ around each non-singleton minority point $\mathbf{x}_i$. Estimated
   minority set is the union of all such hyperspheres.
```

2.4.2 Second step: Synthetic minority set generation:

In the first part of this algorithm, we have defined the minority set as the union of the union of hyperspheres of variable volumes around the non-singleton minority points. Now, we will generate the synthetic minority set $\mathbf{S}_{\text{minor}}$ by selecting random points from these hyperspheres. Depending on the method of estimation (fixed volume or variable volume), we will have two methods for generating the synthetic minority instances. Technically, the methods of point generation are the same in both cases.

The locations of the estimated minority set are given by the locations of non-singleton minority points in the feature space and their respective radii. This set of radii is the same in the case of Algorithm 1 and has a set of variable values in Algorithm 2. A synthetic minority point is generated by selecting an original non-singleton minority point and drawing a random point "New" from its respective hypersphere. An unbiased oversampling is done by drawing $\|\mathbf{S}_{\text{minor}}\|$ number of synthetic points "New" sequentially from the entire set of non-singleton minority points. Algorithm 3 articulates the steps for generating the synthetic minority instances from the estimated set corresponding to

Algorithm 1. In Algorithm 4, we describe the scheme on an estimated set with adaptive volumes of estimated space around the points.

The next concern is about selecting the synthetic minority points. We know the locations of the estimated minority set which is a collection of a hypersphere around each minority training point. The center of a hypersphere is a minority point itself and its radius is equal to ε or ε_i. A synthetic minority point is generated by selecting a minority point and drawing a random point "New" from its respective hypersphere. An unbiased oversampling is done by drawing $\|\mathbf{S}_{\text{minor}}\|$ number of synthetic points "New" sequentially from the entire minority instance set.

$\mathbf{S}_{\text{minor}}$ is the collection of all such synthetic minority points. The members of $\mathbf{S}_{\text{minor}}$ are generated from different minority points.
$\mathbf{A}_{\text{tr}}$ is obtained by taking union of $\mathbf{O}_{\text{minor}}$, $\mathbf{O}_{\text{major}}$ and $\mathbf{S}_{\text{minor}}$.
$\mathbf{A}_{\text{tr}} = \mathbf{O}_{\text{minor}} \cup \mathbf{O}_{\text{major}} \cup \mathbf{S}_{\text{minor}}$.
Classifier model, CM is modeled on $\mathbf{A}_{\text{tr}}$ and CM is invoked to classify the test set $\mathbf{D}_{\text{te}}$ in $\mathbf{Re}_{\text{te}}$.

Algorithm 3 Synthetic minority point generation (followed by training and testing) on estimated set corresponding to Algorithm 1

```
Input: Minority training set $\mathbf{O}_{\text{minor}}$, Majority Training set $\mathbf{O}_{\text{major}}$, Test set $\mathbf{D}_{\text{te}}$, $\varepsilon$
Output: Synthetic minority set $\mathbf{S}_{\text{minor}}$, Predicted test set $\mathbf{Re}_{\text{te}}$
 1: $\|\mathbf{S}_{\text{minor}}\| = 0.5 * (\|\mathbf{O}_{\text{major}}\| - \|\mathbf{O}_{\text{minor}}\|)$
 2: $\mathbf{S}_{\text{minor}} = \emptyset$
 3: p=1;                                  ▷ $p$ is used to loop over the entire minority point set
 4: for i=1 to $\|\mathbf{S}_{\text{minor}}\|$ do
 5:     Generate a random point New from the respective hypersphere centered at $\mathbf{x}_p$ at a radius
        $r$, $0 < r < \varepsilon$
 6:     $p = ((p+1) mod \|\mathbf{O}_{\text{minor}}\|) + 1$ ......          ▷ Selecting the next minority point
 7:     $\mathbf{S}_{\text{minor}} = \mathbf{S}_{\text{minor}} \cup$ New
 8: end for
 9: Augmented training set,
        $\mathbf{A}_{\text{tr}} = \mathbf{O}_{\text{minor}} \cup \mathbf{O}_{\text{major}} \cup \mathbf{S}_{\text{minor}}$
10: Model classifier $CM$ over $\mathbf{A}_{\text{tr}}$
11: Classification result, $\mathbf{Re}_{\text{te}} = CM(\mathbf{D}_{\text{te}})$      ▷ Predicts test set $\mathbf{D}_{\text{te}}$ invoking $CM$
```

2.5 Complexity Analysis of the proposed estimation procedures

We have proposed two minority set estimation procedures. The first procedure gives an estimated set, denoted by α_n. α_n is estimated from the Minimum Spanning Tree of the

Algorithm 4 Synthetic minority point generation (followed by training and testing) from estimated set corresponding to Algorithm 2

```
Input: Minority training set $\mathbf{O}_{\text{minor}}$, Majority Training set $\mathbf{O}_{\text{major}}$, Test set $\mathbf{D}_{\text{te}}$, $\varepsilon_i$ $i = 1, 2, \ldots$ for each $\mathbf{x}_i$
Output: Synthetic minority set $\mathbf{S}_{\text{minor}}$, Predicted test set $Prediction$
 1: $\|\mathbf{S}_{\text{minor}}\| = (\|\mathbf{O}_{\text{major}}\| - \|\mathbf{O}_{\text{minor}}\|)$
 2: $\mathbf{S}_{\text{minor}} = \emptyset$
 3: p=1;                                   ▷ $p$ is used to loop over the entire minority point set
 4: for i=1 to $\|\mathbf{S}_{\text{minor}}\|$ do
 5:     Check if $\mathbf{x}_p$ has just one edge,
          if true, $p = ((p+1) mod \|\mathbf{O}_{\text{minor}}\|) + 1$ and goto step 5 ▷ Ignoring minority points which
        can be a possible outliers
 6:     Generate a random point New from the respective hypersphere centered at $\mathbf{x}_p$ at a random
        radius $r$, $0 < r < \varepsilon_i$
 7:     $p = ((p+1) mod \|\mathbf{O}_{\text{minor}}\|) + 1$ ......          ▷ Selecting the next minority point
 8:     $\mathbf{S}_{\text{minor}} = \mathbf{S}_{\text{minor}} \cup$ New
 9: end for
10: Augmented training set,
        $\mathbf{A}_{\text{tr}} = \mathbf{D}_{\text{tr}} \cup \mathbf{S}_{\text{minor}}$
11: Model classifier $CM$ over $\mathbf{A}_{\text{tr}}$
12: Classification result, $Prediction = CM(\mathbf{D}_{\text{te}})$     ▷ Predicts test set $\mathbf{D}_{\text{te}}$ invoking $CM$
```

given set of points. The estimation depends on the minimum heterogeneous edge weight of the minimum spanning tree. The actual implementation of the Minimum Spanning Tree is not needed in this case. It reduces to calculating the minimum distance of the two sets (minority and majority). For n points, the worst case time-complexity of calculating the minimum distance is $\mathcal{O}$(n log n) and the best case is $\mathcal{O}(n)$. So the time-complexity of α_n is between these two intervals.

In the second estimation scheme, we propose an adaptive estimation procedure for which we need to implement the Relative Neighborhood Graph of the given set of points. The complexity of implementing Relative Neighborhood Graph of n points is $\mathcal{O}$(n log n) [85]. Hence, the time-complexity of implementing $\bar{\alpha_n}$ is $\mathcal{O}$(n log n).

2.6 Experimental Setup

We describe the layout of our empirical study in this section. The four integral components of the study — i] description of datasets, ii] evaluation metrics, iii] parameter configuration of proposed and comparing methods, the choice of classifiers for evaluating the class-imbalance classifiers and iv] Statistical test are provided as follows.

2.6.1 Datasets

We have carried out the experiments on 36 real-world, class-imbalanced datasets. In Table 2.1, we summarize their basic statistics consisting of instances cardinality, imbalance ratio and number of features. In our study, we have considered two-class datasets only. Datasets possessing diversified parameters – *instances cardinality ranging from 106 to 19020, number of features from 3 to 34 and imbalance ratio lying in the range 1.38-85.88* have been employed to test the relative efficacies of the class-imbalance classifiers. They are obtained from Keel repository [1]. Apart from these datasets, we have used a synthetic two dimensional dataset to analyze the issue that this work addresses. Please refer to **Figure 2.1**.

Table 2.1: Description of Datasets

Dataset	*Number of instances*	*Number of attributes*	*Imbalance ratio*
Abalone9_18	731	8	16.40
Appendicitis	106	8	4.05
Bands	365	19	1.70
Bupa	345	6	1.38
Cleveland-0vs4	177	13	12.62
Cleveland12vs345	297	13	2.34
Dermatology-6	358	34	16.9
Ecoli2	336	7	5.46
Ecoli3	336	7	8.60
Glass0	214	9	2.06
Glass1	214	9	1.82
Haberman	306	3	2.78
Heart	270	13	1.25
Ionosphere	351	33	1.79
Magic	19020	10	1.84
New-thyroid1	215	5	5.14
Page-blocks	5472	10	8.79
Phoneme	5404	5	2.41
Pima-indians	768	8	1.87
Poker-8vs6	1477	10	85.88
Poker-89vs5	2075	10	82.00
Segment0	2308	19	6.02
Vehicle1	846	18	2.9
Vehicle2	846	18	2.88
Vowel0	988	13	9.98
Wdbc	569	30	1.64
Winequality-red-8vs6	656	11	35.44
Winequality-white-3vs7	900	11	44.00
Winequality-white-39vs5	1482	11	58.28
Wisconsin	683	9	1.86
Yeast-0256vs3789	1004	8	9.14
Yeast-0359vs78	506	8	9.12
Yeast1	1484	8	2.46
Yeast3	1484	8	8.1
Yeast4	1484	8	28.1
Yeast5	1484	8	32.73

2.6.2 Evaluation metrics

For evaluating a conventional classification task, accuracy is a primary choice. On the contrary, to evaluate a class-imbalance classifier, accuracy alone is not sufficient and we need to consider a number of other aspects. We employ minority class F_1 and AUC for evaluating our work and the comparing class-imbalanced classifiers. For each of the metric, higher the value we obtain, the better the performance.

2.6.3 Comparing methods and Classifiers

The interest of the proposed work lies with the synthetic minority oversampling genre of class-imbalance learning – six comparable methods chosen from the same field. Their technical configurations are detailed below. For fair-play, the cardinality of the synthetic minority set is kept constant across all methods and the proposed method. It is equal to the cardinality difference of the majority and the minority class that is $|\mathbf{S}_{\text{major}} - \mathbf{S}_{\text{minor}}|$. *SMOTE* [12]: It is a base method in the field of class-imbalance learning. SMOTE uses random distances for minority sample generation and neighbourhood cardinality k is 5. *ADASYN* technique generates differential number of minority samples at different locations on the basis of minority density. It aims at learning hard-to-learn minority samples through decision boundary shifting. The only parameter is the number in neighborhood, k which is set to 5. *MWMOTE* [3] identifies the boundary and interior minority samples followed by clustering of the minority instances. Minority instances from the cluster are employed in adding the synthetic points. Parameter settings are as follows: For MWMOTE, the values for different parameters are, k1 = 5, k2 = 3, k3 $=0.5 \times \text{S}_{\text{min}}/2$; $\text{C}_p = 3$; $C_f(th) = 5$, and $CMAX = 2$. These values are recommended in their paper. *Surrounding neighborhood SMOTE* [26] computes surrounding centroid neighborhood of each minority point to generate the synthetic points. k is kept 5 as in the previous methods in order to maintain an equivalence across the comparing methods. We have set the parameters of Cluster-SMOTE [15] according to their recommendations in their paper.

The two proposed schemes are named as –i] *Learning Minority Class prior to Minority Oversampling (LMCMO)* [79] and ii] *Adaptive Learning of Minority Class prior to Minority oversampling (ALMCMO).* The schemes of LMCMO and ALMCMO do not have any user-provided parameter.

We have used two classifiers C4.5 Decision Tree and Logistic Regression based classifier. Both these classifiers are used in their default MATLAB settings.

2.6.4 Statistical Test

To test the difference in performance of two methods statistically, we have employed the Wilcoxon Signed Rank Test. We have conducted the tests for a pair of methods — proposed method, ALMCMO and a competing method. The null hypothesis in our case is as follows — *the median difference of the scores on evaluation metrics (F_1) or AUC between the two methods is 0.* If we fail to reject the null hypothesis, the result is statistically comparable performance of the two methods. If the medians of the two methods are different, we can reject the null hypothesis and infer that the performance of the two algorithms are statistically different. Depending on the evaluation, the method with lower rank sum emerges as the superior one. We have considered the level of significance as $p = 0.05$. According to this significance and Bonferroni correction, the decision p value is set to 0.008.

2.7 Results and Discussion

We have randomly equi-partitioned each dataset into a training set and a test set. These two sets are mutually exclusive. We have organized the results in the following fashion. For each dataset, we have conducted 20 independent runs and reported the mean values. Tables 2.2 and 2.3 report the comparative performance on F_1 and AUC respectively for C4.5 Decision Tree classifier. Similarly, Tables 2.4 and 2.5 report the results on F_1 and AUC scores respectively for the Regression-based classifier. Table 2.6 reports the findings of the statistical testing. Table 2.7 reports the time taken by the methods for computing the synthetic minority sets (for each dataset).

As reported in Table 2.2, ALMCMO has achieved the best F_1 scores on C4.5 Decision Tree classifier for 69.44% (25 out of 36) datasets. The remaining best scores are shared by Cluster-SMOTE (3 out of 36) datasets, NCN-SMOTE (1 dataset), ADASYN, SMOTE and MWMOTE (1 dataset for each), and LMCMO (4 datasets). For AUC, ALMCMO has obtained best scores on 63.89%(23 out of 36 datasets). The remaining thirteen top scores are shared by LMCMO (6 datasets), Cluster-SMOTE, SMOTE, ADASYN (2 datasets each) and NCN-SMOTE (1 dataset).

The performance of the proposed method on Regression-based classifier is similar to the above scenario. On F_1, ALMCMO has achieved best scores on 66.67% (24 out of 36) datasets. The remaining twelve best scores on F_1 are shared by LMCMO (3 datasets), SMOTE (3 datasets), ADASYN (3 dataset) and MWMOTE (3 datasets). ALMCMO has On AUC, ALMCMO has obtained best scores on 72.22% (26 out of 36) cases. The best

Table 2.2: F_1 Results on C4.5 tree

	Methods						
Dataset	ALMCMO	LMCMO	SMOTE	Cluster-SMOTE	ADASYN	MWMOTE	NCN-SMOTE
Abalone9-18	**0.912**	0.898	0.876	0.863	0.883	0.902	0.883
Appendicitis	**0.784**	0.746	0.736	0.772	0.768	0.674	0.771
Bands	**0.632**	0.625	0.621	0.615	0.628	0.627	0.618
Bupa	**0.643**	0.638	0.632	0.629	0.616	0.631	0.612
Cleveland0vs4	**0.484**	0.431	0.445	0.427	0.400	0.347	0.333
Cleveland12vs345	**0.655**	0.619	0.630	0.623	0.624	0.617	0.635
Dermatology-6	**0.944**	0.918	0.940	0.938	0.933	0.940	0.910
Ecoli2	**0.931**	0.925	0.913	0.918	0.914	0.912	0.920
Ecoli3	0.898	**0.907**	0.895	0.889	0.903	0.895	0.888
Glass0	**0.815**	0.768	0.755	0.746	0.755	0.794	0.791
Glass1	**0.756**	0.722	0.738	0.744	0.720	0.700	0.697
Haberman	**0.680**	0.646	0.643	0.641	0.628	0.661	0.637
Heart	**0.756**	0.732	0.742	0.736	0.702	0.730	0.735
Ionosphere	**0.896**	0.887	0.858	0.854	0.860	0.885	0.864
Magic	**0.823**	0.810	0.805	0.803	0.799	0.808	0.816
Newthyroid1	0.936	0.944	0.960	**0.968**	0.966	0.936	0.951
Page-blocks	**0.965**	0.962	0.958	0.959	0.958	0.961	0.949
Phoneme	0.835	0.830	0.835	0.838	0.829	0.822	**0.844**
Pima	**0.711**	0.679	0.686	0.700	0.684	0.686	0.682
Poker8vs6	0.253	0.388	**0.403**	0.282	0.262	0.320	0.343
Poker89v5	**0.097**	0.087	0.032	0.029	0.020	0.057	0.055
Segment0	**0.975**	0.970	0.967	0.970	0.969	0.967	0.973
Vehicle1	0.691	0.683	0.675	**0.693**	0.657	0.672	0.685
Vehicle2	0.868	0.889	0.875	0.876	0.886	**0.891**	0.875
Vowel0	0.970	0.972	0.975	**0.977**	0.970	0.968	0.969
Wdbc	**0.935**	0.928	0.918	0.929	0.926	0.929	0.923
Wine-red-8vs6	0.135	**0.229**	0.139	0.149	0.151	0.155	0.168
Wine-white-3vs7	**0.382**	0.342	0.158	0.147	0.119	0.276	0.282
Wine-white-39v5	0.137	**0.176**	0.060	0.059	0.071	0.083	0.154
Wisconsin	**0.951**	0.942	0.941	0.942	0.932	0.947	0.945
Yeast-0256vs3789	0.521	**0.544**	0.499	0.497	0.495	0.408	0.493
Yeast-0359vs78	**0.381**	0.364	0.317	0.321	0.330	0.283	0.367
Yeast1	0.490	0.504	0.514	0.512	**0.518**	0.496	0.516
Yeast3	**0.743**	0.731	0.676	0.696	0.716	0.681	0.689
Yeast4	**0.378**	0.346	0.284	0.289	0.322	0.282	0.303
Yeast5	**0.689**	0.675	0.647	0.673	0.661	0.671	0.658

scores on the remaining datasets are obtained by Cluster-SMOTE (2 datasets), SMOTE (3 dataset), ADASYN (4 datasets) and MWMOTE (1 dataset).

The above summarized results indicate a certain superiority of the proposed method, ALMCMO over other competing methods. It is also interesting to note that ALMCMO has also surpassed the performance of it's precursor LMCMO. To study and establish the statistical superiority of ALMCMO, we have conducted the Wilcoxon Signed Rank Sum test. We have 2 metrics and 2 classifiers. So, we have conducted 4 sets of such tests and reported the set of p values. We have carried out the statistical tests as described in the above section. At $p = 0.05$, we could reject the null hypothesis for all four cases. To be precise, the proposed method ALMCMO's performance was statistically superior to

Table 2.3: AUC Results on C4.5 tree

	Methods						
Dataset	ALMCMO	LMCMO	SMOTE	Cluster-SMOTE	ADASYN	MWMOTE	NCN-SMOTE
Abalone9-18	0.654	0.610	0.679	**0.713**	0.675	0.610	0.671
Appendicitis	**0.761**	0.719	0.648	0.692	0.651	0.656	0.730
Bands	**0.611**	0.601	0.605	0.596	0.603	0.595	0.601
Bupa	**0.652**	0.632	0.619	0.616	0.603	0.627	0.604
Cleveland-0vs4	**0.778**	0.769	0.699	0.673	0.676	0.664	0.660
Cleveland12vs345	**0.593**	0.560	0.588	0.579	0.558	0.562	0.584
Dermatology-6	0.975	**0.985**	0.977	0.975	0.972	0.977	0.964
Ecoli2	**0.881**	0.856	0.851	0.841	0.868	0.846	0.868
Ecoli3	**0.783**	0.766	0.747	0.733	0.761	0.762	0.739
Glass0	**0.787**	0.737	0.735	0.737	0.744	0.782	0.784
Glass1	**0.738**	0.712	0.726	0.725	0.722	0.685	0.686
Haberman	**0.604**	0.583	0.592	0.594	0.564	0.593	0.586
Heart	**0.762**	0.731	0.738	0.732	0.698	0.729	0.735
Ionosphere	**0.889**	0.878	0.852	0.844	0.853	0.876	0.855
Magic	**0.806**	0.797	0.794	0.795	0.793	0.790	0.799
Newthyroid1	0.918	0.920	0.922	**0.938**	0.930	0.911	0.920
Page-blocks	**0.929**	0.896	0.923	0.924	0.908	0.910	0.923
Phoneme	0.824	0.808	0.822	0.828	0.809	0.812	**0.830**
Pima	**0.689**	0.653	0.664	0.682	0.655	0.668	0.657
Poker-8vs6	0.681	0.683	**0.725**	0.676	0.677	0.679	0.653
Poker-89vs5	**0.548**	0.536	0.506	0.512	0.501	0.519	0.520
Segment0	**0.989**	0.988	0.985	0.986	0.986	0.978	0.985
Vehicle1	**0.742**	0.725	0.738	0.739	0.737	0.734	0.728
Vehicle2	0.930	**0.942**	0.922	0.928	0.932	0.926	0.921
Vowel0	0.944	**0.950**	0.928	0.929	0.901	0.912	0.927
Wdbc	**0.931**	0.928	0.916	0.926	0.922	0.927	0.923
Winequality-red-8vs6	0.570	**0.619**	0.591	0.596	0.591	0.583	0.589
Winequality-white-3vs7	0.687	**0.692**	0.626	0.614	0.598	0.642	0.647
Winequality-white-39vs5	0.570	**0.602**	0.546	0.542	0.554	0.542	0.567
Wisconsin	**0.951**	0.939	0.938	0.938	0.928	0.946	0.941
Yeast-0256vs3789	0.714	0.722	**0.749**	0.724	0.743	0.703	0.699
Yeast-0359vs78	**0.658**	0.649	0.640	0.642	0.652	0.617	0.651
Yeast1	0.649	0.657	0.662	0.638	**0.666**	0.646	0.664
Yeast3	**0.869**	0.852	0.825	0.828	0.843	0.809	0.822
Yeast4	0.677	0.659	0.675	0.684	**0.693**	0.655	0.632
Yeast5	**0.881**	0.866	0.831	0.845	0.832	0.847	0.830

all the competing methods for the two classifiers (Regression-based classifier and C4.5 Decision Tree classifier) for the two metrics.

We have carried out the statistical tests as described in the previous section. After the Bonferroni correction, $p = 0.008$, we can reject the null hypothesis for all twenty-four cases when directly comparing our approaches against the others for the two learning algorithms. Besides obtaining a p-value less than 0.008, the lower rank sum is obtained by ALMCMO in all 24 cases. These data establish that ALMCMO's performance is statistically superior to all the competing methods for the two classifiers (Regression-based classifier and C4.5 Decision Tree classifier) over the two evaluation metrics. Table 2.6 records these findings.

Table 2.4: F_1 Results on Regression based classifier

	Methods						
Dataset	ALMCMO	LMCMO	SMOTE	Cluster-SMOTE	ADASYN	MWMOTE	NCN-SMOTE
Abalone9-18	**0.924**	0.908	0.877	0.879	0.898	0.915	0.883
Appendicitis	**0.807**	0.792	0.800	0.805	0.797	0.798	0.781
Bands	**0.660**	0.637	0.638	0.638	0.611	0.643	0.596
Bupa	**0.668**	0.654	0.661	0.661	0.642	0.663	0.620
Cleveland-0vs4	0.415	**0.492**	0.446	0.364	0.407	0.294	0.295
Cleveland12vs345	**0.703**	0.656	0.652	0.645	0.637	0.686	0.623
Dermatology-6	**0.961**	0.949	0.909	0.912	0.920	0.903	0.930
Ecoli2	**0.941**	0.932	0.934	0.935	0.929	0.936	0.903
Ecoli3	0.927	0.925	0.907	0.903	0.919	**0.930**	0.881
Glass0	**0.825**	0.796	0.817	0.811	0.818	0.819	0.773
Glass1	0.773	**0.782**	0.752	0.753	0.773	0.769	0.702
Haberman	**0.789**	0.765	0.754	0.767	0.757	0.778	0.716
Heart	**0.817**	0.777	0.782	0.778	0.751	0.780	0.736
Ionosphere	0.906	0.905	**0.909**	0.896	0.892	0.885	0.864
Magic	**0.858**	0.847	0.838	0.835	0.842	0.852	0.804
Newthyroid1	**0.968**	0.955	0.959	0.959	0.960	0.943	0.953
Page-blocks	0.972	0.968	0.967	0.969	**0.978**	0.976	0.951
Phoneme	**0.889**	0.870	0.877	0.871	0.872	0.873	0.840
Pima	**0.738**	0.726	0.735	0.725	0.732	0.731	0.694
Poker8vs6	0.438	0.378	**0.589**	0.517	0.461	0.528	0.542
Poker89vs5	0.118	**0.129**	0.080	0.084	0.076	0.039	0.077
Segment0	0.965	**0.975**	0.969	0.970	0.972	**0.975**	0.973
Vehicle1	**0.775**	0.760	0.770	0.760	0.766	0.767	0.717
Vehicle2	**0.934**	0.930	0.905	0.902	0.918	0.893	0.927
Vowel0	**0.982**	0.977	0.978	0.976	0.976	0.975	0.976
Wdbc	**0.955**	0.936	0.941	0.942	0.936	0.944	0.934
Wine-red-8vs6	**0.204**	0.156	0.121	0.124	0.129	0.144	0.138
Wine-white-3vs7	**0.369**	0.249	0.090	0.104	0.126	0.173	0.314
Wine-white-39vs5	**0.171**	0.148	0.070	0.073	0.043	0.115	0.098
Wisconsin	0.953	0.958	**0.965**	0.962	0.961	0.955	0.938
Yeast-0256vs3789	**0.576**	0.553	0.470	0.482	0.463	0.446	0.518
Yeast-0359vs78	**0.414**	0.408	0.298	0.291	0.334	0.239	0.245
Yeast1	0.501	0.519	0.543	0.529	**0.569**	0.549	0.462
Yeast3	0.954	0.955	0.944	0.947	0.950	**0.957**	0.932
Yeast4	0.239	0.303	0.281	0.264	**0.330**	0.313	0.257
Yeast5	**0.985**	0.982	0.982	0.980	0.981	0.984	0.974

2.8 Summary

Our works, ALMCMO and LMCMO have presented a scheme of minority set estimation which acts as a subspace for synthetic minority oversampling. The novelty of this work lies with the estimation of the example minority set. The goal of ALMCMO is to adaptively estimate the minority class by allowing a varying volume of estimated minority spaces around different minority points. The relative distribution and densities of the points are taken into account while selecting the varying radii and volumes. This concept is an addition to what was proposed in LMCMO which followed a strictly constant estimated volume across all minority points. In ALMCMO, we have used a very simple

Table 2.5: AUC Results on Regression based classifier

	Methods						
Dataset	ALMCMO	LMCMO	SMOTE	Cluster-SMOTE	ADASYN	MWMOTE	NCN-SMOTE
Abalone9-18	0.661	0.618	0.707	**0.718**	0.656	0.608	0.672
Appendicitis	**0.717**	0.687	0.693	0.702	0.698	0.695	0.688
Bands	0.607	0.580	0.618	0.601	0.585	**0.624**	0.574
Bupa	**0.671**	0.653	0.650	0.662	0.630	0.657	0.620
Cleveland-0vs4	**0.789**	0.761	0.740	0.733	0.689	0.675	0.708
Cleveland12vs345	**0.600**	0.580	0.563	0.575	0.563	0.596	0.575
Dermatology6	**0.988**	0.976	0.967	0.959	0.963	0.961	0.982
Ecoli2	**0.893**	0.862	0.861	0.859	0.841	0.867	0.820
Ecoli3	**0.845**	0.815	0.799	0.788	0.814	0.794	0.762
Glass0	0.811	0.786	0.801	0.795	**0.814**	0.809	0.761
Glass1	0.731	0.731	0.713	0.718	**0.747**	0.731	0.682
Haberman	**0.772**	0.754	0.745	0.758	0.741	0.765	0.702
Heart	**0.812**	0.771	0.780	0.772	0.744	0.775	0.734
Ionosphere	**0.895**	0.889	0.889	0.882	0.879	0.873	0.849
Magic	**0.840**	0.834	0.829	0.828	0.822	0.837	0.798
Newthyroid1	**0.955**	0.903	0.906	0.904	0.898	0.932	0.878
Page-blocks	0.941	0.910	0.941	**0.946**	0.926	0.920	0.933
Phoneme	**0.870**	0.839	0.858	0.856	0.851	0.860	0.826
Pima	**0.715**	0.699	0.704	0.706	0.686	0.699	0.678
Poker-8vs6	0.702	0.673	**0.772**	0.743	0.730	0.733	0.728
Poker-89vs5	**0.571**	0.567	0.521	0.508	0.541	0.515	0.533
Segment0	**0.988**	0.987	0.984	0.982	0.984	0.985	0.984
Vehicle1	**0.719**	0.714	0.707	0.708	0.675	0.673	0.676
Vehicle2	**0.963**	0.958	0.934	0.927	0.944	0.925	0.954
Vowel0	**0.979**	0.972	0.947	0.955	0.945	0.947	0.960
Wdbc	**0.954**	0.933	0.937	0.940	0.933	0.944	0.938
Wine-red-8vs6	**0.603**	0.569	0.571	0.553	0.570	0.578	0.532
Wine-white-3vs7	**0.685**	0.626	0.552	0.578	0.563	0.559	0.623
Wine-white-39vs5	0.546	0.560	**0.564**	0.588	0.561	0.521	0.536
Wisconsin	0.945	0.952	**0.961**	0.956	0.955	0.952	0.928
Yeast-0256vs3789	**0.744**	0.723	0.721	0.718	0.713	0.701	0.710
Yeast-0359vs78	**0.667**	0.664	0.624	0.631	0.653	0.587	0.593
Yeast1	0.664	0.673	0.682	0.674	**0.701**	0.693	0.642
Yeast3	**0.902**	0.890	0.871	0.881	0.891	0.874	0.859
Yeast4	0.595	0.628	0.682	0.669	**0.712**	0.692	0.596
Yeast5	**0.892**	0.855	0.808	0.817	0.773	0.799	0.765

yet effective Relative Neighborhood Graph to get the neighborhood relations and configuration of the points. The performance of the proposed methods indicate superiority over all other competing methods on C4.5 Decision Tree and Regression-based classifier. It indicates that the estimation of minority set followed by generation of the synthetic minority points from these estimated spaces gives better guarantee of their classes. The results of empirical study also indicate the same. In future, we aim to formulate more refined strategies for estimating the minority sets which will in turn enable us to sample a good and prospective synthetic minority set.

Table 2.6: This table reports the p value at which we can reject the null hypothesis and claim that the performance of the proposed method (ALMCMO) is statistically different (in our case superior than) from that of a competing method. Lower the p value, more significant is the difference. At $p = 0.05$ level of significance and employing Bonferroni correction, $p = 0.008$ is the decision threshold. ALMCMO has achieved a value lesser than 0.008 in all the cases.. The cases where we could reject the null hypothesis are written in **bold-face**.

	LMCMO	SMOTE	Cluster-SMOTE	ADASYN	MWMOTE	NCN-SMOTE
C4.5 Decision Tree based classifier						
F_1						
ALMCMO	0.02045	**0.00014**	**0.00014**	**0.00006**	**0.00028**	**0.00009**
AUC						
ALMCMO	**0.00099**	**0.00079**	**0.00033**	**0.00024**	**0.00001**	**0.00002**
Regression based classifier						
F_1						
ALMCMO	**0.00196**	**0.00150**	**0.00013**	**0.00045**	**0.00075**	**0.00001**
AUC						
ALMCMO	**0.00003**	**0.00264**	**0.00167**	**0.00026**	**0.00027**	**0.00001**

Table 2.7: This table reports the average time taken to compute the synthetic minority oversampling by each method on each dataset. All the methods are executed on same platform (i7 processor, 8 GB RAM, OS - Ubuntu 14.04). Each method has been executed 10 times on each dataset. We have reported the average time taken by the methods. The unit of time used here is second (s). We can see that the two proposed methods, ALMCMO and LMCMO have taken the least amount of time to generate the synthetic minority sets. Time taken by LMCMO is the lowest among all the methods.

	Methods						
Dataset	ALMCMO	LMCMO	SMOTE	Cluster-SMOTE	ADASYN	MWMOTE	NCN-SMOTE
Abalone9-18	0.0036	0.0021	0.1497	0.1729	0.0403	0.0081	0.1656
Appendicitis	0.0014	0.0003	0.0155	0.0244	0.0068	0.0040	0.0211
Bands	0.0049	0.0007	0.0263	0.1437	0.1058	0.0270	0.0301
Bupa	0.0023	0.0004	0.0172	0.0860	0.1187	0.0252	0.0196
Cleveland0vs4	0.0016	0.0006	0.0358	0.0429	0.0043	0.0030	0.0861
Cleveland12vs345	0.0019	0.0005	0.0296	0.0752	0.0498	0.0154	0.0329
Dermatology-6	0.0032	0.0022	0.0733	0.0896	0.0160	0.0053	0.0903
Ecoli2	0.0021	0.0008	0.0608	0.0893	0.0376	0.0102	0.0582
Ecoli3	0.0020	0.0008	0.0665	0.0854	0.0207	0.0066	0.0736
Glass0	0.0017	0.0003	0.0177	0.0558	0.0410	0.0121	0.0194
Glass1	0.0016	0.0003	0.0173	0.0517	0.0449	0.0119	0.0201
Haberman	0.0027	0.0004	0.0314	0.0894	0.0571	0.0156	0.0343
Heart	0.0018	0.0004	0.0094	0.0692	0.0770	0.0219	0.0111
Ionosphere	0.0027	0.0012	0.0247	0.0921	0.2316	0.0238	0.0294
Magic	0.2780	0.1289	14.8876	71.3316	256.2103	4.7601	4.9004
Newthyroid1	0.0017	0.0005	0.0369	0.0644	0.0157	0.0065	0.0405
Page-blocks	0.0015	0.0004	0.0322	0.0519	0.0167	0.0066	0.0366
Phoneme	0.0430	0.0314	1.4952	2.8799	3.0597	0.1664	1.5722
Pima	0.0281	0.0131	1.2426	5.0769	23.1231	0.5376	0.7028
Poker-8vs6	0.0036	0.0009	0.0656	0.2317	0.3933	0.0505	0.0595
Poker-89vs5	0.0125	0.0066	0.3355	0.3503	0.0442	0.0076	0.4559
Segment0	0.0114	0.0081	0.4755	0.4984	0.0709	0.0117	0.5711
Vehicle1	0.0212	0.0115	0.3927	0.6042	1.0085	0.0781	0.4297
Vehicle2	0.0048	0.0020	0.0938	0.1974	0.3095	0.0449	0.1009
Vowel0	0.0045	0.0019	0.0878	0.2010	0.3443	0.0469	0.0946
Wdbc	0.0052	0.0031	0.1828	0.2327	0.1181	0.0190	0.2012
Winequality-red-8vs6	0.0035	0.0010	0.0349	0.1643	0.2548	0.0447	0.0489
Winequality-white-3vs7	0.0046	0.0023	0.0147	0.1263	0.0196	0.0050	0.2285
Winequality-whitw-39vs5	0.0047	0.0033	0.2113	0.2082	0.0212	0.0059	0.2751
Wisconsin	0.0087	0.0055	0.3269	0.3389	0.0370	0.0089	0.4110
Yeast-0256vs3789	0.0057	0.0008	0.0585	0.2157	0.4219	0.0442	0.0550
Yeast-0359vs78	0.0054	0.0024	0.1892	0.2667	0.1578	0.0196	0.1936
Yeast1	0.0028	0.0011	0.0913	0.1194	0.0389	0.0091	0.0998
Yeast3	0.0070	0.0028	0.2009	0.5636	1.5725	0.0923	0.1635
Yeast4	0.0070	0.0037	0.2890	0.4091	0.3353	0.0299	0.2817
Yeast5	0.0070	0.0043	0.3191	0.3573	0.0971	0.0119	0.3379

Chapter 3

Handling multi-label datasets – from a perspective of feature extraction

3.1 Introduction

Multi-label datasets differ from traditional datasets by virtue of the membership of instances to more than one overlapping label. In a regular dataset, an instance belongs to exactly one class. Hence, in regular datasets, the question of the multi-label nature of membership does not arise. Likewise, traditional classifiers dealing with single label data expect the instances to arise from a single and known class distribution.

In multi-label datasets, a single instance in a given input space can belong to one or more of the possible class labels. The need for efficient processing of multi-label data is backed by the availability of datasets with multi-label characteristics from several real-world applications. Beginning with text categorization [42] and [28], data with multi-label characteristics have emerged from different genres namely images [5] [63], music [50], bioinformatics [4], chemical data analysis [55], tag recommendation systems [44] and video [68]. Consequently, multi-label classification and learning grabbed the attention of the data science community.

Let a multi-label dataset be denoted by $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{Y}_i), i = 1, 2, \ldots, n\}$ and the label set cardinality be $\mathcal{L}$. The label set of $\mathbf{x}_i$ is $\mathcal{Y}_i = \{y_{i1}, y_{i2}, \ldots, y_{iL}\}$. Let us assume that each label has exactly two classes positive (1) and negative (0) that is $\mathcal{Y}_{ij}$ can be either 1 or 0, $j = 1, 2, \ldots, \mathcal{L}$. An instance $\mathbf{x}_i$ has to be rightfully classified into either positive

(1) or negative (0) class for $\mathcal{L}$ labels.

In a multi-label dataset, a single set of instances possess the same representation across all labels. But their positive or negative class memberships vary from label to label. This leads to a variable class partition of the same instance set across different labels. An efficient way of handling this problem is by generating label-specific features. For $\mathcal{L}$ labels, a single representation of an instance is given L different mappings, one for each label. A supervised feature extraction framework which relies on the label specific class information has proved to be a very effective tool for learning the multi-label datasets. In recent years, quite a number of works have focused on selecting or extracting dedicated features for tackling the multi-label problem. Label specific feature extraction was proposed in LIFT [105]. In LIFT, following the clustering of the positive and negative classes of each label, the authors extract a label-specific feature set.Works dealing with feature extraction and selection from multi-label datasets include [98] and [49]. A detailed account and comparative analysis of the extant works in multi-label feature extraction and selection in first-order framework can be found in [83]. Joint feature selection and classification (JFSC) [37] and [102] performs label-correlated feature selection of multi-label datasets.

In this work, we propose two techniques of dedicated feature set extraction for different labels of a multi-label dataset. In the previous chapter, we have used the techniques of Minimum Spanning Tree (MST) and Relative Neighborhood Graph (RNG) for estimating the minority class spaces. In this chapter, we use these same two techniques in a different context, for extracting the dedicated features.

3.2 Approach I - A Minimum Spanning Tree (MST) based feature extraction

A feature extraction scheme can be effective if it can competently capture the dissimilarities which separates the different classes of an instance set. A multi-label dataset has a single instance set with varying (likely to be different) partitions for different labels. To be precise, it has dissimilar positive class and negative class structures for different labels (considering two-class multi-label datasets). Here, we present a feature extraction scheme which banks on this characteristic of such datasets to generate a discriminative feature set for each label by initially selecting a shape-related discriminative subset of instances per label. In this work, we follow a *first-order approach* of multi-label learning where we learn a single classifier for each label. The discriminative feature sets obtained by our schema are used to learn the classifiers. The following three steps are sequentially performed for each label to obtain its corresponding distinctive feature set.

- *Shape of the classes*: Different partitions of the same instance set for different labels is the motivation of our schema. We obtain the shapes of both positive as well as negative classes of a label following this step twice, once for each. Here, we assume that the set of all points in a class (i.e, not only the given points belonging to the class, but also all those points which belong to the class but are not given) is a connected set. Note that we are given finitely many points from a class, even though the original set is uncountable (considering real space). The minimal connected set that can be obtained from the given points in the class is the Minimal Spanning Tree of the given points, where the edge weight is taken as the distance between the points. We consider Minimal Spanning Tree (MST) to represent the geometry or shape of the class.

- *Key Point Subset / lattice of a shape*: Once we have the positive and negative class geometries of a label, we would like to select the 'key points or lattice' of their respective geometries. In a graph, a node or point can be denoted important if its removal along with its connected neighbors results in substantial distortion to the original geometry. By this rationale, it is evident that the potential candidates for the Key Point Subset are the higher degree vertices of the MST. We select the higher degree vertices of the both the positive class MST and negative class MST individually into the Key Point Subset for a label (till they cover a predefined fraction of edges). Selection of the points is done from the actual set of instances. Hence, from a single instance set for all labels we select a dedicated set of instances for each label.

- *Feature extraction*: An instance is likely to be closer to the members of the class to which it belongs than it is to the members of the other class. A natural extension of this is followed by extracting a distance-based feature vector for each instance. The transformed feature vector of an instance for a label is obtained by calculating its distance from the Key Point Subset members of that label. If there are k labels, an instance gets k transformations, one for each label.

The transformed feature set for the training instance set with respect to a label is employed to model its corresponding classifier and this process is repeated across all labels. While classifying a test instance over a label, its transformed mapping (extracted feature vector) is obtained by calculating its distance from the Key Point Subset of that label and it is fed to the respective label classifier to obtain the class prediction. The action is repeated across all labels to obtain classification over the entire label set.

3.2.1 Algorithm

Proposed Method I has two stages. First stage is the the feature extraction and classifier modelling step, where we derive the transformed feature set for each label from a single instance set. In this stage, we employ the set of training instances to select the Key Point Subset for each label from both positive and negative Minimum Spanning Trees followed by extraction of a distance based feature set. Next, we model a set of classifiers, one for each label, modelled upon the distance-based feature set. In the second stage, classification of a test point is done according to its new set of mapping for each label and feeding it to the respective label classifier.

- *First stage*: We shall assume that we are given a set of n training examples. Let x_i denote the ith training sample point, $i = 1, 2, ..., n$. Let the maximum number of labels to be attributed to a point be k. Let Y_{ij} denote membership of ith point to jth label. That is

$$\begin{aligned} Y_{ij} = &\quad 1, \quad \text{if ith point has jth label} \\ = &-1, \quad \text{if ith point does not take the jth label} \end{aligned} \tag{3.1}$$

  Initially, for each label j, we segregate data points in the training set into positive and negative sets P_j and N_j respectively according to their membership to class j.

$$P_j = \{x_i : Y_{ij} = 1\} \qquad j = 1, 2, .., k \tag{3.2}$$

$$N_j = \{x_i : Y_{ij} = -1\} \quad j = 1, 2, .., k \tag{3.3}$$

  Once we have the segregated positive and the negative sets of label j, we construct two Minimum Spanning Trees (MSTs) corresponding to positive set and negative set for the concerned label j where edge-weight of an edge joining two nodes a_1, a_2 is the Euclidean distance between them. For nominal data, we considered the Hamming Distance between the points as the edge weight.

$$\text{Tree}_{P_j} = \text{MST}(P_j) \tag{3.4}$$

Algorithm 5 Proposed Method I

```
Input: Training attribute set $S$, Training label set $Y$, $covG$
Output: Classifiers – $C_1, C_2, ..., C_k$, Key Point Sets – $KPS_1, KPS_2, .., KPS_k$
 1: procedure TRAINING($S$, $Y$, $covG$)
 2:     for each label $j \rightarrow 1, k$ do
 3:         Segregate $P_j, N_j$ as in Eqn (1)
 4:         Form $Tree_{P_j}, Tree_{N_j}$ as in Eqn (2)
 5:         Sort elements of $P_j, N_j$ into $P_{j\_sorted}, N_{j\_sorted}$ according to $norm\_degree$ values as in Eqn (3)
 6:         Select $KPS_{j_p}, KPS_{j_n}$ as in Eqn (4)
 7:         Obtain $KPS_j$ by taking union of $KPS_{j_p}$ and $KPS_{j_n}$ as in Eqn (5)
 8:         Transformed training feature set with respect to label $j$, $\beta_j(S)$ is obtained from $KPS_j$ as in Eqn (6)
 9:         Classifier $C_j$ is modelled on $\beta_j(S)$ and $Y$
10:     end for
11: end procedure
    ------------------------------------------------- Input: Classifier – $C_1, ..., C_k$, Key Point Sets – $KPS_1, .., KPS_k$, test point $p$
    Output: Predicted labels for $p$
12: procedure TEST($C_1, C_2, ..., C_k$, $KPS_1, KPS_2, .., KPS_k$, $p$)
13:     for each label $j \rightarrow 1, k$ do
14:         $\beta_j(p)$ is obtained
15:         $C_j$ is fed with $\beta_j(p)$ to get the $j^{th}$ label prediction for $p$
16:     end for
17: end procedure
```

$$\text{Tree}_{N_j} = \text{MST}(N_j) \tag{3.5}$$

The degrees of data points of P_j, N_j in $Tree_{P_j}$, $Tree_{N_j}$ are used in selection of KPS (Key Point Subset) for positive and negative sets respectively. Degree of the nodes is normalized with respect to the number of edges in the MST to get *norm_degree* value.

Node centrality of a node is its degree value normalized with respect to the number of edges of the network .

$$\text{norm\_degree}(P_j)_i = \text{degree}((P_j)_i)/(|P_j| - 1) \tag{3.6}$$

$i = 1, 2, ..|P_j|$, where $(P_j)_i$ denotes i^{th} element of set P_j.

$$\text{norm\_degree}((N_j)_i) = \text{degree}((N_j)_i)/(|N_j| - 1); \tag{3.7}$$

i=1,2,..,$|N_j|$ and $(N_j)_i$ denotes i^{th} element of set N_j.

The data points with higher degrees are more relevant candidate points of the subset. In order to select such points sequentially, we sort the elements of P_j in decreasing order according to their degree norm_degrees in $Tree_{P_j}$ into P_{j_sorted}.

$$P_{j\_\text{sorted}} = \text{sort\_decreasing\_degree}(P_j) \tag{3.8}$$

Similarly,

$$N_{j\_\text{sorted}} = \text{sort\_decreasing\_degree}(N_j) \tag{3.9}$$

Based on P_{j_sorted} and N_{j_sorted}, Key Point Subset(KPS) is selected for label j, for which we first construct KPS_{j_p} for positive set and KPS_{j_n} for negative set respectively. Elements of P_{j_sorted} are included in KPS_{j_p} sequentially starting from $(P_{j\_sorted})_1$ till the cumulative sum of *norm_degree* values of the included points equals or just exceeds the edge coverage value, covG. covG is the only user-specified parameter used in the algorithm and its value is generally 0.6.
Let the first N nodes of P_{j_sorted} be selected.

$$\sum_{i=1}^{n} \text{norm\_degree}(P_{j\_sorted})_i \geq \text{covG} \tag{3.10}$$

$$\sum_{i=1}^{n-1} \text{norm\_degree}(P_{j\_sorted})_i < \text{covG} \tag{3.11}$$

KPS_{j_n} is constructed in similar fashion.

Let m_{j_p}, m_{j_n} be the cardinalities of KPS_{j_p}, KPS_{j_n} respectively.

KPS_j is constructed by combining KPS_{j_p} and KPS_{j_n}. Hence we get consistent edge-coverage of positive and negative classes in the Key Point Subset. Let m_j be the cardinality of KPS_j.

$$(KPS)_j = (KPS)_{j_p} \cup (KPS)_{j_n} \tag{3.12}$$

Since KPS_{j_p} and KPS_{j_n} are mutually exclusive,

$$\begin{aligned} |(KPS)_j| &= |(KPS)_{j_p}| + |(KPS)_{j_p}| \\ &= m_{j_p} + m_{j_n} \\ &= m_j \end{aligned} \tag{3.13}$$

- *Second stage*: For each label we will construct transformed feature set from its corresponding KPS. Each KPS is derived from its respective MSTs which is built upon inter-point distances between elements of set. So, for each label we construct the transformed feature set by taking distance of training points from the respective KPS.
  Transformed feature inherent to label j for a training sample point x is as follows:

  $$\beta_j(x) = [d(x, KPS_{j1}), d(x, KPS_{j2}), ..., d(x, KPS_{jm_j})] \tag{3.14}$$

  $$\text{where } j = 1, 2, ..., k$$

  where KPS_{j1} denotes the 1st element of KPS_j and $d(x, KPS_{j1})$ denotes euclidean distance between x and 1st element of KPS_j.

- *Second stage*: We will train k single-label classifiers, $C_1, C_2, ..., C_k$ for k labels from β, label-based transformed features by invoking the respective KPS. Classification and learning of an unknown instance follows a similar label-specific path. Initially,for a label j, label-specific α is calculated for the test instance from KPS_j. Based on that, the classification result is obtained by invoking classifier C_j.

Remarks:

- **Complexity of Proposed Method I**: Initially, we have to implement the MST of all points of a dataset. The complexity for MST implementation of n points is $\mathcal{O}(\text{n log n})$. In feature extraction step, we have to extract the lattice points followed by distance computation. For each label and n feature points,the number of lattices is at most n. Hence, the overall complexity of feature extraction for $\mathcal{L}$ labels is $\mathcal{O}(n \times \mathcal{L})$.

3.3 Proposed Approach II- A Relative Neighborhood Graph (RNG) based feature extraction

- Extracting the class geometries of labels

  Our first goal is to extract the positive and negative class geometry of each label. To perceive the geometry, we generate a Relative Neighborhood Graph (RNG) of the entire set of training dataset where the edge weights are the distance between the points. RNG shows the connectivity of a data point or vertex to its adjacent neighborhood and the interconnectivity of the points gives the overall anatomy of the feature points. A RNG of G is an undirected graph defined from G where $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected whenever there is no third point $\mathbf{x}_k$ such that $\text{d}(\mathbf{x}_i, \mathbf{x}_k) < \text{d}(\mathbf{x}_i, \mathbf{x}_j)$ and $\text{d}(\mathbf{x}_j, \mathbf{x}_k) < \text{d}(\mathbf{x}_i, \mathbf{x}_j)$. For a given set of points, it's MST is a subgraph of it's RNG. We may note that this tree will be same for all the labels. But the membership of the vertexes or the data points vary from label to label and leads to different positive and negative class structures for the labels. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be the training data and $\mathcal{Y}_i = \{y_{i1}, \ldots, y_{i\text{L}}\}$ be the class label vector associated with instance $\mathbf{x}_i$. We have assumed that there are $\mathcal{L}$ labels in the dataset.

  We will extract the positive and negative class geometries (with respect to each label) from the RNG. To extract the above-said, we need to look at the membership of the vertexes to each label. For a label, the membership of a vertex can be either positive (1) - if it belongs to that label or negative (0) - if it does not belong to that label. The class-memberships of the data points will likely vary across the labels.

  Let us consider an edge e_{ij} between two vertices, v_i and v_j. If the class-membership of $\mathbf{x}_i$ and $\mathbf{x}_j$ with respect to label k are the same (both 0 or both 1), we call edge e_{ij} a homogeneous edge. If y_{ik} and y_{jk} (the memberships of $\mathbf{x}_i$ and $\mathbf{x}_j$) are both 1, we term e_{ik} to be a positive homogeneous edge. If the class-memberships (y_{ik} and

y_{jk}) are 0, we call it negative homogeneous edge. So, for each label, we will have a set of homogeneous edges which is a subset of the RNG edges. We can further partition this homogeneous edge set into two mutually exclusive sets of positive homogeneous edges and negative homogeneous edges. For each label, we will have a set of positive homogeneous edges and a set of negative homogeneous edges which is described in the next paragraph. We will extract the positive and negative class lattices of the label from its respective sets of homogeneous edges. Homogeneous edges lie in the regions of the same class memberships.

A homogeneous edge (belonging to a certain class) with smaller weight will likely be a better representative of that class than another with higher weight. It is because, with increasing edge weight, the vertexes (associated with the edge) become sparser in the feature space and eventually overlap with the vertexes associated with a different class. But a vertex associated with a shorter edge will have another vertex near its vicinity which affirms its class-membership. Hence, to get the positive class lattice (for a label), we arrange the positive homogeneous edges in increasing order of their weights. For a certain label k, we select a N_{Pk} number of positive homogeneous edges in increasing order of their weights and compute their midpoints. The set of N_{Pk} midpoints represents the positive lattice of label k. Similarly, we compute N_{Nk} lattice points to represent the negative lattice of label k. In case of extreme imbalance when we do not get any positive homogeneous edge, we select the positive points themselves to represent the positive lattices.

We determine the values of N_{Pk} and N_{Nk} in light of the degree of class-imbalance of label k. Let the degree of imbalance of label k be imb_k , which is the ratio of the cardinality of the negative class of label k to that of the positive class of label k. For the negative class (generally the majority class) of label k, we select the value of N_{Nk} as N_{Pk}*$(\log_2(\text{imb}_k) + 1)$. The logarithm function allows us to add deviations in the N_{Nk} cardinalities in a controlled manner. Let us consider two scenarios to analyse this aspect. If there is no imbalance in two classes of label k, that is $\text{imb}_k = 1$, $(\log_2(\text{imb}_k) + 1)$ will be 1 and we will select N_{Pk} points as the negative class. On the contrary, if imb_k is 16 (dataset is highly imbalanced with respect to label k), $(\log_2(\text{imb}_k) + 1)$ will be 5 and we will select $5\times$ N_{Pk} points to represent the negative class. We can also select different figures of N_{Pk} and N_{Nk}. We present a discussion in Remarks 1 at the end of this section.

- Extracting the features

Now, we extract the features for each label. For that, we obtain the distance of a data point from the sets of positive and negative lattice points of a label. In order to make the positive information stand out in a pool of negative data, we multiply the distances from the positive lattices with the respective class imbalance ratio of that label. The above computed distances give the imbalance-informed mapping of a data point for that label. The set of N_{Pk} positive distances and N_{Nk} negative distances gives the transformed mapping of $\mathbf{x}$ with respect to label k.

Remarks:

1. **Values of N_{Pk} and N_{Nk}:** The number of lattice points for the positive class and the negative class are given by N_{Pk} and N_{Nk} respectively. The extracted feature set cardinality will increase with the increase in the number of lattice points. Increasing the number of lattice points will give better discernible and classification capabilities to the classifier. But this is accompanied with an increase in computational complexity. While setting the values of N_{Pk} and N_{Nk}, we have to make a trade-off between complexity and performance. Experiment 4 in the empirical study explores this aspect.

2. **Distance function used:** We have used Euclidean distance and Hamming distance functions for numeric and nominal datasets respectively.

3.4 Algorithm

Let the multi-label dataset be denoted by D and the number of class labels for $\mathcal{D}$ be $\mathcal{L}$. $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{Y}_i), |1 \leq i \leq n$, $\mathcal{Y}_i$ denotes class label vector of $\mathbf{x}_i\}$. $\mathcal{Y}_{\mathrm{i}} = \{y_{i1}, y_{i2}, \ldots, y_{il}\}$. y_{ij} is 1 when label j is positive for instance $\mathbf{x}_i$, otherwise the value of y_{ij} is 0. Let each $\mathbf{x}_{\mathrm{i}} \in \mathbf{R}^{\mathbf{p}}$. We randomly equi-partition $\mathcal{D}$ into a training set, $\mathcal{D}_{tr}$ and a test set, $\mathcal{D}_{te}$. Let $\mathcal{X}$ be the set of training instances (without the label information).

$$\mathcal{X} = \{\mathbf{x}_i, \text{i=1, 2,}\ldots\text{, n}\} \tag{3.15}$$

We calculate class-imbalance ratio of each label j, $j = 1, 2, \ldots, l$ denoted by imb_j.

$$\begin{aligned} \text{imb}_\text{k} &= \frac{\text{Number of negative training instances for label } k}{\text{Number of positive training instances for label } k} \\ \Rightarrow \text{imb}_\text{k} &= \frac{||\{\mathbf{x}_i \text{ such that } \mathcal{Y}_{ik} = 0, i = 1, 2, \ldots, n\}||}{||\{\mathbf{x}_i \text{ such that } \mathcal{Y}_{ij} = 1, i = 1, 2, \ldots, n\}||} \end{aligned} \tag{3.16}$$

In multi-label datasets, we have a single set of observations covering all labels. We construct a Relative Neighborhood Graph (RNG) whose vertices are represented by the members of $\mathcal{X}$.

$$\text{Tree} = \text{RNG}(\mathcal{X}) \tag{3.17}$$

To extract more refined information about the class structures, we have to extract a label-specific lattice from these graphs. Firstly, we extract the homogeneous edges of the graph. As explained earlier, homogeneous edge for which both vertices belong to the same class, there are two classes of homogeneous edges, positive and negative.

Let $\mathbf{x}_i$ denote the i^{th} vertex of the graph and $c_j(\mathbf{x}_i)$ denote the class-membership of $\mathbf{x}_i$ to label j.

$$c_j(\mathbf{x}_i) = \begin{cases} 1 & \text{if } y_{ij} = 1 \\ 0 & \text{else} \end{cases} \tag{3.18}$$

Let an edge of *Tree* between two vertices $\mathbf{x}_i$ and $\mathbf{x}_k$ be represented by e_{ik}. w_{ik} denotes the edge-weight of e_{ik}.

$$e_{ik} = \{(\mathbf{x}_i, \mathbf{x}_k), w_{ik}\}, i, k = 1, 2, \ldots, n, i \neq k \tag{3.19}$$

$\mathbf{S}_{pj}$ and $\mathbf{S}_{nj}$ are the sets of positive and negative homogeneous edges of label j respectively.

For each label j, $j = 1, 2, \ldots, \mathcal{L}$,

$$\mathbf{S}_{pj} = \{e_{ik}, c_j(\mathbf{x}_i) = c_j(\mathbf{x}_k) = 1\} \tag{3.20}$$

Similarly,

$$\mathbf{S}_{nj} = \{e_{ik}, c_j(\mathbf{x}_i) = c_j(\mathbf{x}_k) = 0\} \tag{3.21}$$

We arrange the elements of $\mathbf{S}_{pj}$ and $\mathbf{S}_{nj}$ in increasing order of their edge-weights to get the ranks of their respective elements. Let, for an edge e_{ik}, its rank in its respective set (the set where it belongs) be denoted by $R(e_{ik})$. We obtain the ranks of the edges because we will select the lattice points from the shorter homogeneous edge weights. Shorter homogeneous edges have lower ranks than longer edges.

The mid-points of edges in $\mathbf{S}_{pj}$ are stored in $\mathbf{M}_{pj}$. $\mathbf{M}_{nj}$ stores the mid-points of $\mathbf{M}_{nj}$. Let N_{Pj} and N_{Nj} denote the number of negative and positive lattice points of label j

respectively.

$$\mathbf{M}_{pj} = \bigcup_{\substack{e_{ik}\in\mathbf{S}_{pj}\\ R(e_{ik})\le N_{pj}}} \frac{\mathbf{x}_i + \mathbf{x}_k}{2} \tag{3.22}$$

$$\mathbf{M}_{nj} = \bigcup_{\substack{e_{ik}\in\mathbf{S}_{nj}\\ R(e_{ik})\le N_{nj}}} \frac{\mathbf{x}_i + \mathbf{x}_k}{2} \tag{3.23}$$

Let the representations of $\mathbf{M}_{pj}$ and $\mathbf{M}_{nj}$ be as follows.

$$\mathbf{M}_{pj} = \{\mathbf{m}_{1j},\ \mathbf{m}_{2j}, \ldots,\ \mathbf{m}_{k_{pj}}\} \tag{3.24}$$

$$\mathbf{M}_{nj} = \{\mathbf{m}^{'}_{1j},\ \mathbf{m}^{'}_{2j}, \ldots,\ \mathbf{m}^{'}_{k_{nj}}\} \tag{3.25}$$

$\mathbf{m}_{1j},\ \mathbf{m}_{2j}, \ldots,\ \mathbf{m}_{k_{pj}}$ represent the individual members of $\mathbf{M}_{pj}$.
Similarly, $\mathbf{m}^{'}_{1j},\ \mathbf{m}^{'}_{2j}, \ldots,\ \mathbf{m}^{'}_{k_{nj}}$ represent the elements of $\mathbf{M}_{nj}$. It is easy to note that the number of elements of $\mathbf{M}_{pj}$ and $\mathbf{M}_{nj}$ depend on data distribution and are likely unequal. We have represented their cardinalities by k_p and k_n respectively.

The transformed mapping of instance $\mathbf{x}_i$ with respect to label j denoted by $\mathbf{z}_{ij}$ is as follows:

$$\mathbf{z}_{ij} = f_j(\mathbf{x}_i) = \{d(\mathbf{x}_i, \mathbf{m}_{1j}), \ldots,\ d(\mathbf{x}_i, \mathbf{m}_{k_{pj}}),\ d(\mathbf{x}_i, \mathbf{m}^{'}{}_{1j}), \ldots,\ d(\mathbf{x}_i, \mathbf{m}^{'}{}_{k_{nj}})\} \tag{3.26}$$

$\mathbf{z}_{ij}$ is a $k_p + k_n$ dimensional vector and it serves as the transformed feature point (i^{th}) for label j. Its first k_p components are generated by taking distance from the midpoints of the positive homogeneous edges and multiplying them with the imbalance ratio of label j. The remaining k_n components by taking distance from the negative homogeneous edges.

Let $\mathcal{Z}_j$=$\{\mathbf{z}_{ij}, i = 1, 2, \ldots, n\}$. $\mathcal{Z}_j$ represents the transformed feature mapping of the training instances in $\mathcal{D}_{tr}$ for label j.

For each label j, we train a classifier $\mathcal{W}_j$ by invoking $\mathcal{Z}_j$. For classifying a test instance $\mathbf{t}$ with respect to label j, we first obtain its transformed mapping for label j and invoke $\mathcal{W}_j$ to predict it's class. We have used a linear SVM classifier implementation of LIBSVM ([11]) for modeling and classification.

Remarks

- **Complexity of Proposed Method II**: Let us assume that the dataset has n instances and $\mathcal{L}$ labels. The first step of the scheme is RNG implementation of the given points and we need to implement it just one time (for all labels). The complexity for RNG implementation of n points is $\mathcal{O}(\text{n log n})$ The complexity of feature generation for each label is of $\mathcal{O}(n)$. So, the overall complexity of feature extraction for all labels is $\mathcal{O}(\text{n} \times \mathcal{L})$.

3.4.1 Datasets

Two principal categories of real-life multi-label datasets are used: *five* datasets with numeric attributes and *five* datasets containing nominal attributes. In each of these data sets, for every instance, there is a positive or negative labeling with respect to every category in the data set. That is for each label, we have exactly one *two* class classification problem. Multi-label datasets are described in terms of standard parameters like domain, number of instances, attribute types, number of attributes, number of labels as well as two additional multi-label parameters, label cardinality and label density which give average number of labels per example and normalized label cardinality with respect to the possible number of labels respectively.

The description of the *ten* benchmark datasets is given in Table 3.1. The parameters associated with the datasets are explained as follows – D denotes the cardinality of a dataset. L and F denote the cardinalities of the label set and feature set respectively. The average number of positive labels across all points of a dataset is is given by $L.Card$. $L.Uniq$ gives the number of unique label combinations in a dataset.

Table 3.1: Description of Datasets

Dataset	domain	att.type	(D)	(L)	(F)	$(L.Card)$	$(L.Uniq)$
Corel5k	image	nominal	5000	374	499	3.522	3175
Enron	text	nominal	1702	53	1001	3.378	753
Lang Log	text	nominal	1460	75	1004	1.180	286
medical	biology	nominal	978	45	1449	1.245	94
Slashdot	text	nominal	3782	22	1079	1.181	156
CAL500	music	numeric	502	174	68	26.044	502
Image	image	numeric	2000	5	294	1.236	20
Music	music	numeric	592	6	72	1.869	27
Scene	image	numeric	2407	6	294	1.047	15
Yeast	biology	numeric	2417	14	103	4.037	198

Table 3.2: **Predictive performance of methods in terms of multi-label evaluation metrics** — ↑ indicates higher is better and ↓ indicates lower is better, best outcome is indicated in bold-face

	Corel5k	Enron	LLOG	Medical	Slash	CAL500	Image	Music	Scene	Yeast
Hamming Loss ↓										
	mean ± std	mean ± std	mean ± std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std
B_R	0.010±0.001	0.060±0.002	0.017±0.002	0.013±0.001	0.050±0.001	0.138±0.003	0.187±0.007	0.286±0.002	0.108±0.004	0.202±0.005
CLR	0.011±0.001	0.055±0.001	0.018±0.001	0.038±0.002	0.052±0.001	**0.137**±0.003	0.185±0.006	0.272±0.002	0.106±0.004	0.201±0.005
LIFT	0.010±0.001	0.047±0.001	**0.015**±0.002	0.014±0.001	**0.040**±0.001	0.138±0.003	0.166±0.004	0.194±0.001	0.083±0.002	0.198±0.003
MLKNN	0.010±0.001	0.054±0.003	0.016±0.002	0.017±0.002	0.046±0.001	0.140±0.003	0.181±0.004	0.209±0.002	0.093±0.002	0.197±0.004
RAKEL	0.013±0.001	0.075±0.003	0.018±0.003	**0.012**±0.001	0.047±0.001	0.198±0.004	0.172±0.008	0.299±0.001	0.141±0.003	0.201±0.006
Proposed Method I	**0.009**±0.001	0.048±0.002	**0.015**±0.001	**0.012**±0.001	**0.040**±0.001	**0.137**±0.003	0.156±0.002	0.190±0.001	**0.081**±0.002	0.192±0.003
Proposed Method II	**0.009**±0.001	**0.045**±0.002	0.017±0.002	**0.012**±0.003	**0.040**±0.001	0.138±0.003	**0.152**±0.002	**0.176**±0.001	**0.081**±0.002	**0.190**±0.003
Average Precision ↑										
	mean std	mean std	mean std	mean std	mean std	mean std	mean std	mean std	mean std	mean std
B_R	0.115±0.002	0.427±0.009	0.184±0.009	0.785±0.007	0.564±0.004	0.493±0.003	0.713±0.005	0.741±0.002	0.782±0.004	0.568±0.002
CLR	0.275±0.003	0.673±0.008	0.368±0.007	0.502±0.004	0.672±0.003	0.498±0.004	0.786±0.004	0.736±0.001	0.844±0.003	0.652±0.001
LIFT	**0.287**±0.004	0.683±0.010	**0.382**±0.007	0.843±0.004	0.670±0.003	0.492±0.003	0.812±0.004	0.808±0.001	0.876±0.002	0.762±0.001
MLKNN	0.239±0.002	0.609±0.008	0.290±0.006	0.777±0.007	0.589±0.004	0.484±0.003	0.774±0.004	0.789±0.002	0.853±0.003	0.758±0.002
RAKEL	0.124±0.003	0.556±0.008	0.192±0.010	0.777±0.006	0.609±0.004	0.383±0.004	0.794±0.005	0.722±0.002	0.847±0.002	0.622±0.003
Proposed Method I	0.265±0.003	0.656±0.008	0.328±0.012	**0.863**±0.003	0.667±0.003	0.497±0.004	**0.824**±0.003	0.816±0.001	**0.882**±0.002	**0.769**±0.002
Proposed Method II	0.268±0.002	**0.684**±0.006	0.341±0.010	0.843±0.003	**0.681**±0.003	**0.499**±0.004	0.820±0.003	**0.848**±0.002	**0.882**±0.003	0.768±0.003
Coverage ↓										
	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std
B_R	0.894±0.004	0.582±0.007	0.425±0.005	**0.044**±0.005	0.117±0.005	0.952±0.008	0.296±0.002	0.362±0.001	0.111±0.001	0.626±0.002
CLR	0.316±0.008	**0.223**±0.009	**0.154**±0.008	0.088±0.006	**0.108**±0.003	0.763±0.005	0.185±0.004	0.348±0.002	0.128±0.002	0.472±0.004
LIFT	0.319±0.007	0.245±0.008	0.194±0.006	0.058±0.005	0.111±0.004	0.762±0.004	0.181±0.003	0.291±0.002	0.070±0.002	0.456±0.003
MLKNN	0.316±0.009	0.263±0.011	0.182±0.011	0.075±0.006	0.186±0.005	0.762±0.005	0.207±0.004	0.311±0.003	0.084±0.002	0.458±0.006
RAKEL	0.882±0.002	0.527±0.007	0.438±0.004	0.081±0.005	0.196±0.004	0.967±0.004	0.212±0.002	0.389±0.002	0.118±0.002	0.526±0.003
Proposed Method I	**0.289**±0.006	0.251±0.007	0.165 ±0.005	0.050±0.004	0.113±0.004	**0.756**±0.004	0.168±0.003	0.288±0.002	0.067±0.002	**0.449**±0.003
Proposed Method I	0.310±0.008	0.245±0.006	0.174 ±0.005	0.051±0.005	0.122±0.005	0.757±0.003	**0.165**±0.003	**0.264**±0.003	**0.065**±0.002	0.453±0.004
One Error ↓										
	mean± std	mean± std	mean± std	mean± std	mean± std	mean± std	mean± std	mean± std	mean± std	mean± std
B_R	0.829±0.013	0.506±0.014	0.856± 0.011	0.252±0.012	0.502±0.009	0.283±0.010	0.408±0.009	0.357±0.011	0.346±0.014	0.248±0.012
CLR	0.708±0.012	0.265±0.010	0.718± 0.008	0.386±0.007	0.433±0.009	0.124±0.010	0.329±0.009	0.276±0.011	0.252±0.008	0.228±0.010
LIFT	0.687±0.011	0.253±0.010	**0.704**±0.013	0.191±0.012	0.425±0.013	0.123±0.011	0.286±0.008	0.254±0.010	0.208±0.009	0.235±0.011
MLKNN	0.744±0.014	0.330±0.012	0.816± 0.012	0.312±0.008	0.642±0.010	0.125±0.011	0.345±0.009	0.283±0.013	0.247±0.014	0.229±0.009
RAKEL	0.823±0.012	0.406±0.014	0.845±0.014	0.254±0.009	0.462±0.010	0.312±0.012	0.303±0.015	0.388±0.008	0.247±0.009	0.253 0.013
Proposed Method I	0.706±0.010	0.240±0.008	0.765±0.011	**0.170**±0.010	0.441±0.012	0.118±0.009	0.270±0.011	0.251±0.010	**0.198**±0.009	**0.218**±0.009
Proposed Method II	**0.660**±0.009	**0.232**±0.007	0.768±0.010	0.198±0.010	**0.413**±0.010	**0.099**±0.008	**0.265**±0.008	**0.193**±0.006	0.202±0.005	0.229±0.009
Ranking Loss ↓										
	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std	mean±std
B_R	0.624±0.004	0.248±0.005	0.244±0.006	0.040±0.005	0.158±0.010	0.412±0.005	0.293±0.004	0.232±0.003	0.168±0.006	0.309±0.005
CLR	**0.118**±0.003	0.078±0.004	**0.134**±0.007	0.052±0.004	**0.096**±0.005	0.182±0.003	0.173±0.003	0.229±0.004	0.076±0.004	0.171±0.002
LIFT	0.132±0.003	0.086±0.004	0.184±0.004	**0.035**±0.005	0.098±0.004	0.189±0.003	0.157±0.003	0.154±0.002	0.068±0.004	0.171±0.003
MLKNN	0.138±0.002	0.100±0.006	0.175±0.003	0.042±0.006	0.172±0.006	0.184±0.005	0.192±0.002	0.175±0.003	0.086±0.003	0.168±0.004
RAKEL	0.603±0.004	0.226±0.007	0.325±0.006	0.078±0.005	0.285±0.003	0.467±0.008	0.192±0.003	0.252±0.001	0.103±0.004	0.226±0.006
Proposed Method I	0.124±0.003	0.091±0.005	0.156±0.007	0.036±0.003	0.101±0.004	0.182±0.004	**0.143**±0.003	0.148±0.002	0.065±0.003	**0.163**±0.003
Proposed Method II	0.135±0.003	**0.076**±0.005	0.158±0.007	0.036±0.003	0.104±0.005	**0.179**±0.004	0.148±0.003	**0.123**±0.002	**0.063**±0.003	0.166±0.003

3.4.2 Comparison against State-of-the-art approaches and the experimental settings

We have considered five popular diversified multi-label schemes for empirical comparison with the proposed methods. The methods are as follows – i] *Binary Relevance (BR)* – a first order approach, ii] *Calibrated Label Ranking (CLR)* – second order approach, iii] *LIFT* – a feature transformation approach with *r=0.1*, iv] *MLKNN* – an algorithm adaptation approach with *k=10,11 or 12* and v] *RAKEL* – a higher order approach following parameter settings *k=3 and number of subsets equal to twice the number of labels.*

The two proposed methods are implemented in the framework of Binary Relevance (BR), that is one classifier for each label. LIBSVM [11] implementation of linear SVM is used as the base classifier. Euclidean distance is considered for numeric datasets and for nominal datasets Hamming distance is taken. In Proposed Method I, the only user supplied parameter is *covG* and its value is set to 0.6 for all datasets. This value is chosen via a two-fold cross-validation (Hold Out setting). The set of values explored was {0.3, 0.4, 0.5, 0.6, 0.7}. We have split each dataset into a training set and a test set. For each value of *covG* in the above given range, we performed 10 such runs and noted the results. We chose covG=0.6 as it gave superior performance on most datasets. Proposed Method II has just one user-provided parameter — number of positive lattice points (N_{Pk}) which is used for mapping the features.

3.4.3 Evaluation Metrics

Evaluation metrics: Five metrics namely *Hamming Loss, Coverage, One Error, Ranking Loss, Average Precision* are employed to evaluate the relative efficacies of all methods [91]. Let $\mathbf{x}_i$, i=1,2,. . . ,N be the set of N test instances and $\mathcal{Y}_i$ be the L-dimensional label vector of $\mathbf{x}_i$. Let $\overline{\mathcal{Y}_i}$ be the complement label set of $\mathbf{x}_i$. Let α_i be the label prediction vector for $\mathbf{x}_i$. We denote the label specific predicted score of $\mathbf{x}_i$ for label j by $f_j(\mathbf{x}_i)$.

- **Hamming Loss**: It measures the fraction incorrect predictions for all instances across the entire label set. Lower the value achieved by a classifier, better is its performance.

$$\text{Hamming Loss} = \frac{1}{\text{NL}} \sum_{\text{i=1}}^{N} \mathcal{Y}_i \oplus \alpha_i \tag{3.27}$$

- **Average Precision**: It calculates the fraction of relevant labels ranked higher (predicted) than a particular label[105]. $\mathrm{r}_i(j)$ denotes the rank of label j for $\mathbf{x}_i$ instance predicted by a classifier.

$$\text{Average Precision} = \frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\frac{1}{|\mathcal{Y}_i|}\sum_{\gamma\in\mathcal{Y}_i}\frac{|\{\gamma\in\mathcal{Y}_i : r_i(\gamma)\leq r_i(\gamma')\}|}{\mathrm{r_i}(\gamma)} \tag{3.28}$$

- **One-Error**: One error counts the number of instances for which the predicted top-rank label is not present in the actual label set. Lower the value of one-error, better is the performance of the classifier.

$$\text{One Error} = \frac{1}{N}\sum_{i=1}^{N}[\arg\max_{\text{label}_j\in\alpha_i} f_j(\mathbf{x}_i)\notin\mathcal{Y}_i] \tag{3.29}$$

- **Coverage**: Let us consider an ordered list of predicted labels for each instance, where the top-ranked label is the label number one. Coverage evaluates the number of steps we need to move down the list to get the set of all true labels of the instance. It can easily be intuited that the lesser the value of coverage the better is the performance. Let l_j be the j^{th} label. Let rank$(\mathbf{x}_i, l_j)$ be the rank of j^{th} label w.r.t. instance $\mathbf{x}_i$.

$$\text{Coverage} = \frac{1}{L}(\frac{1}{N}\sum_{i=1}^{n}\max_{l_j\in\alpha_i}\text{rank}(\mathbf{x}_i, l_j) - 1) \tag{3.30}$$

- **Ranking Loss**: Ranking Loss calculates the average percentage of mis-ordered pairs of labels. Lower value of ranking loss is desirable for a classifier. The numerator of the following term represents the number of mis-ordered label pairs. Let us assume, $\mathbf{x}_i$ has positive label k and negative label j. But, the classifier has wrongly assigned more confidence (membership score f) to the j^{th} label than that of k^{th} label of $\mathbf{x}_i$. Such a situation contributes to the Ranking Loss. The numerator calculates the number of such mis-ordered pairs.

$$\text{Ranking loss} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\{(l_k, l_j),\ f_k(\mathbf{x}_i)\leq f_j(\mathbf{x}_j), (l_k, l_j)\in Y_i\times\overline{Y}_i\}|}{|\mathrm{Y}_i||\overline{Y}_i|} \tag{3.31}$$

3.4.4 Results and Discussions

Comparisons in terms of *five* evaluation metrics is done for the proposed method, and *five* other baseline approaches. For each dataset, *50* percent of the samples are randomly selected and used as the training set while the remaining samples are used as the test set. Mean and standard deviation values on the metric outputs are computed over *ten* independent runs and reported in the table. Table 3.2 presents the performance of the methods on accuracy-based, ranking-based and label-based metrics in sequence. The estimated mean values for *ten* runs as well the corresponding standard deviations are also given. The best performance on each dataset is indicated in bold face in the column for the corresponding method. From Table 3.2, the number of best outcomes achieved by each method (comparing and proposed) is calculated. Note that there should essentially be 50 best scores for 5 metrics and 10 datasets. But on *Hamming Loss* and *Average Precision*, on more than one occasion the best score is shared by two or more methods. As a consequence the total number of best scores obtained is 59. The summarization shows the competence of the two proposed methods. Proposed Method I and Proposed II achieve *18* and *26* best results. Together they obtain 44 out of 59 best scores (74.57%). CLR and LIFT obtain 7 and 6 best scores respectively. RAKEL and BR achieve one best score. The above stated results require some more analysis with respect to individual performances on each of the *seven* evaluation metrics to provide a vivid picture.

The two proposed methods have shown better performance with respect to the other competing methods on metrics like *Hamming Loss, Average Precision* and *One Error*. On three of these metrics, either or both of these two methods (Proposed I and Proposed Method II) have achieved best scores on 80-100% of the datasets. The remaining two metrics, Ranking Loss and Coverage deal with the correct ranking of labels. The results indicate that on these two metrics, the efficaciousness of the two proposed techniques is not as superior as that of the previous case. On each of these metrics, Proposed method I and Proposed Method II achieve best scores on 60% datasets. CLR (which operates on the ranking of the labels) achieves best scores on 30% of the remaining datasets and remaining cases are shared by LIFT and BR.

In the previous paragraph, we have discussed the performance of the two proposed techniques in a unified fashion. We may note that the performance of Proposed method II is slightly better than that of Proposed method I. On *Hamming Loss*, *Average Precision* and *Coverage*, the performance of the two methods are comparable. But on *One Error* and *Ranking Loss*, Proposed Method II achieves twice the number of best scores with respect to Proposed method I. Hence, overall we can say that Proposed Method II is a better technique than that of Proposed Method I.

3.5 Summary

In this chapter of the thesis, we have focused on multi-label datasets, where the instances belong to more than one overlapping labels. We have tackled this problem by introducing a couple of schemes for dedicated feature extraction. We have used graph based techniques MST and RNG to find the differential class structures (varying over labels) followed by label-specific feature extraction. The two proposed schemes have delivered competent to superior performance on 10 real-world multi-label datasets.

Distinct positive and negative class geometries for different labels is a typical characteristic of multi-label datasets. In addition to this, the multi-label datasets are also found to be class-imbalanced. Coupling these two attributes leads to another important property – differential class-imbalance ratio of labels. In the next chapter, **Chapter 4** of this thesis, we work on that aspect to achieve a fruitful learning of multi-label datasets.

Chapter 4

Handling multi-label datasets – addressing imbalance of the labels

4.1 Introduction

Multi-label datasets have gained the attention of the machine learning community in the past two decades. A multi-label dataset is characterized be the membership of an instance to more than one overlapping label. Let $\mathcal{D}$ be a multi-label dataset, $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{Y}_i) : i = 1, 2, \ldots, n\}$, where $\mathbf{x}_i$ denotes the i^{th} instance and $\mathcal{Y}_i$ represents its label vector. If a dataset $\mathcal{D}$ has $\mathcal{L}$ labels, $\mathcal{Y} = \{y_{i1}, y_{i2}, \ldots, y_{iL}\}$, y_{ij} denotes the j^{th} label membership of instance $\mathbf{x}_i$. We have considered two class multi-label problems, each label can take exactly one of two values, 0 (negative) or 1 (positive). Inception of multi-label research was with text categorization [28]. Other real-world multi-label sources include music [50], images [63] [5], bioinformatics [4], codebook construction [19] and medical [92].

For effective handling of multi-label datasets, we have to analyze the intrinsic characteristics of the multi-label datasets. A prevalent characteristic of multi-label datasets is class imbalance of the labels. In imbalanced datasets, quantitative disproportion exists in the distribution of different classes. If we consider a two-class dataset, the quantitatively abundant class and the quantitatively scarce class are termed as majority class and minority class respectively. Let imb be the imbalance ratio a dataset. It is the ratio of number of instances in the majority class of a dataset to the number of instances in it's minority class. In regular datasets we have exactly one label, hence we have a single imbalance ratio. In a multi-label dataset, class-membership of the instances vary from label to label. Different labels deal with differing aspects associated with a dataset. As we have discussed in **Chapter 3** of this thesis, this leads to different positive and negative class cardinalities for different labels. Consequently different imbalance ratios are obtained for different labels. Generally speaking, for most of the labels of the multi-label datasets, the positive class remains under-represented and becomes the minority class. But at times, in a few cases, the negative class also becomes the minority class. The differential imbalance ratios coupled with the above stated class-switching for different labels makes the class imbalance problem more critical as well as interesting. For example, in the yeast dataset [20] (with 14 labels), the minimum imbalance ratio and maximum ratio is 1.32 (for label 12) and 50.74 (for label 14) respectively. A single framework with a single set of parameters may not work well across the two diversified labels. In recent years, a few works like COCOA [103] and [17] have given attention to this pertinent aspect.

In this chapter, we present two different and distinct techniques to address the problem of class imbalance in multi-label datasets. The techniques are i] label-specific minority class oversampling for different labels and ii] an imbalance adaptive cost-sensitive learning framework.

Oversampling the minority class has been a popular technique for handling imbalance of datasets and learning the minority instances. Starting with SMOTE [23], several and diversified oversampling techniques have been developed over the years to address the issue of imbalance of datasets. In oversampling, synthetic points are added to increase the cardinality as well as representation of the under-represented minority class. In this paper, we introduce an imbalance-adaptive oversampling scheme for multi-label datasets. We oversample a set of label-specific synthetic minority points, one for each label and use them (along with original training set) to train a set of first-order classifiers. The novelty of our scheme lies with the use of reverse-nearest neighborhood or reverse k-nearest neighborhood (RkNN) exploration. Reverse k-nearest neighbor of an instance **p** are all those instances which have **p** as one of their k-nearest neighbors. The number of RkNNs of **p** is not necessarily k and its value is auto-regulated by the local density and distribution of points around **p**. Reverse nearest neighborhood principles are used to demarcate the neighborhood as well as neighbors of a candidate minority point. Selecting an existing minority point as the seed point, a synthetic minority point is oversampled in it's reverse nearest neighborhood. In regular SMOTE and SMOTE-inspired oversampling techniques, a generic 'k' value or neighborhood size is selected across the entire dataset. Reverse nearest neighborhood allows us an adaptive neighborhood size for performing the oversampling. For each label, the synthetic minority points are oversampled from the reverse-nearest neighborhood of the minority instances. To cope with the differential imbalance problem of multi-label datasets, we oversample a dedicated set of synthetic instances for each label. For each label we train a classifier using the original set of instances and the label specific synthetic minority set. We have presented and compared our results with several state-of-the-art multi-label classifiers as well as class-imbalance handling classifiers.

Cost-sensitive learning [34] was one of the fundamental techniques for tackling the issue of class-imbalance and consequently detecting the 'hard-to-learn' positive or minority instances. To nullify the natural bias of the classifier towards the quantitatively abundant majority class, a higher mis-classification penalty is set for the quantitatively scare minority class. The main goal is to bias the classifier towards identifying the minority samples. In the second solution proposed in this research, to address differential class-imbalance further, we adopt a cost-sensitive learning scheme where the misclassification cost adapts to the imbalance ratio of the labels. A multi-label dataset has differing values of imbalance ratio across the labels. In such a situation, selecting a single misclassification cost for the minority class across will not yield proper learning. Instead, we select a set of misclassification cost values of the minority (positive) class, one for each label. Between two labels with differing degree of imbalance, we set a higher misclassification penalty for the one with higher imbalance than that of the other.

4.2 Related Work

Multi-label learning methods are broadly classified into two types –*Problem transformation approach (PT)* and *Algorithm Adaptation Approach (AA)* [27]. On the other hand, studies such as the one in [61] differentiate the multi-label classifiers into three groups, namely *Problem transformation*, *Algorithm Adaptation* and *Ensemble of multi-label classifiers.*

Problem transformation approaches modify or decompose a multi-label dataset to fit it in a framework of regular decomposition. Depending on the number of decompositions and number of labels involved in a classifier, this class of classifiers are further sub-divided into *first order*, *second order* and *higher order* paradigms [107]. In first-order PT, only one label is involved in a classifier while for second-order and higher-order approaches, two and more number of labels are involved in a classifier respectively. Notable problem transformation approaches are namely Binary Relevance [54], power set of labels [5], pruned problem transformation[73] and calibrated label ranking [24]. Binary relevance, (BR) is the most primitive form of *PT* approach, where a series of binary classifiers is generated, one for each label. Though computationally sound (linear with label cardinality), BR is criticized for its inability to capture label correlations [89]. The solution proposed in [5] accommodated label correlations by employing Label Powerset. It generated $2^8 = 256$ classifiers for learning 8 labels. Despite involving label correlations, the scheme lacked computational feasibility as it generated an exponential number of classifiers. A more feasible approach was given in RAKEL [88], which considered random subsets of labels. Calibrated Label ranking [24] scheme provided multi-label outputs on the basis of pair-wise classification, considering a synthetic label to distinguish the relevant and irrelevant groups of labels. Ensemble of classifiers like RAKEL [88], ensembles of classifier chains [72] [14] [74] and ensembles of pruned sets are also popular and effective in learning multi-label datasets. In addition to these, a number of feature selection and extraction methods transform the features in context of each label and follow first-order approach to complete the learning. They are discussed in the detail in the next paragraph. In Algorithm-Adaptation approach, an existing classifier is adapted in the context of a multi-label scenario. Quite a number of classifier paradigms like k-nearest neighborhood classifier [106], naive bayes algorithm [104], back-propagation neural networks [108] are adapted to facilitate multi-label learning. In [106], the k-nearest neighbors of a test point are identified. Following that, their label configurations and the principles of maximum posteriori are used to determine the label predictions of the test instance. In [108], the usual back-propagation algorithm is used with small modifications to accommodate the multi-label characteristics. The error function of the back-propagation algorithm is replaced with a ranking loss minimization function which operates on the fact that a relevant label of an instance is ranked higher than another label to which the instance does not belong. Another scheme [62] uses the cross-entropy error function in back-propagation neural network for facilitating multi-label learning.

Apart from the above, multi-label datasets are analyzed from newer perspectives like feature or dataset preprocessing and class distribution of the labels. Label specific feature extraction was proposed in LIFT [105]. In LIFT, following the clustering of the positive and negative classes of each label, the authors extract a label-specific feature set. Feature selection is also done by a number of works on the basis of class characteristics of the labels. Works dealing with feature extraction and selection of multi-label datasets include [98] and [49]. A detailed account and comparative analysis of the extant works in multi-label feature extraction and selection in first-order framework can be found in [83]. Joint feature selection and classification (JFSC) [37] and [102] performs label-correlated feature selection of multi-label datasets. Class distribution of a multi-label dataset is a veritable data mine and gives a number of pertinent information. As said earlier, multi-label datasets are class-imbalanced and often differently imbalanced. COCOA [103] has addressed the imbalance issue in their work by considering an ensemble of classifiers using pair-wise label correlation. A few more works, [96] and [38] have used a cost-adaptive paradigm to address multi-label problems. In [76], authors have integrated the data gravitational model with a multi-label lazy classifier for improving the minority class performance of imbalanced multi-label datasets. Several other techniques like convex relaxation [29], ensembles of random graph [84] and graph classification [45] have also been employed to address multi-label classification.

4.3 Proposed Approach I

4.3.1 Working principles – Principles of reverse nearest neighborhood

Definition 1: Given a set of instances $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ ($\mathcal{X} \subset \mathbb{R}^N$) and a point $\mathbf{p}$ ($\mathbf{p} \in \mathbb{R}^N$), a Reverse Nearest Neighbor query concerning $\mathbf{p}$ in search space $\mathcal{X}$ retrieves all the points $\mathbf{x}_i \in \mathcal{X}$ that have $\mathbf{p}$ as their nearest neighbor. Thus, a Reverse k-Nearest Neighbor (RkNN) search returns all those points $\mathbf{x}_i \in \mathcal{X}(i = 1, 2, \ldots, n)$ whose k-nearest neighborhood contain $\mathbf{p}$.

In k-nearest neighborhood of a point, exactly k points can be present in its neighborhood. These k points are closest to $\mathbf{p}$ than other points in the search space. On the contrary, when the cardinality of the search space is n, reverse k-nearest neighborhood of $\mathbf{p}$ can contain 0 to n points. RkNN does not define any fixed number of neighbors around query point $\mathbf{p}$. The number of neighbors around a point varies from location to location and it depends on the relative configuration of the query point and the points in the search space. The RkNN count of $\mathbf{p}$ is adaptive to the density and distribution of points around $\mathbf{p}$ and gives a more versatile estimation of neighborhood than k-nearest neighborhood based scheme. An example is illustrated in Figure 4.1.

4.3.2 Proposed method

Motivated to handle the differential class imbalance of multi-label learning, we present an oversampling scheme which generates a dedicated set of synthetic minority instances for each label. A dedicated set of synthetic minority instances for each label is more effectual than a single synthetic minority set for all labels. We can generate the minority set according to the requirement of a label. It allows us to set the size of the synthetic minority set of a label in consonance with the imbalance ratio of that label. Let us

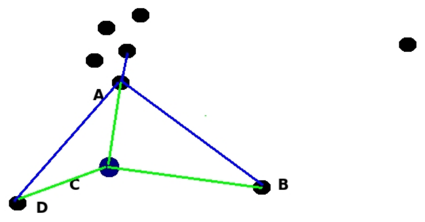


Figure 4.1: We want to find the RkNNs of point C. We consider k=1. It can be found visually that the Euclidean distance between A and C is less than between B and C as well as D and C. If we were to find 1-nearest neighbor of C, A would have been the answer. Now, let us analyse with respect to RkNN. Though A is closest to C but C is not closest to A, A has some nearer points or neighbors in its vicinity. Hence, A is not a RkNN of C. Though B may seem far off from C, it becomes a RkNN of C because C is nearer to B than any other given point in the feature space. The same holds for D also, hence D becomes a RkNN of C. At k value 1, C has 2 RkNNs $\{B, D\}$. RkNN gives a density based neighborhood estimation of the points.

have two labels a and b with different imbalance ratios imb1 and imb2 respectively (where imb1 $<$ imb2). We need to add more points in the synthetic minority set of b than that of a in order to equate the cardinalities of their respective majority and minority classes. We assume that the positive class is the minority class across all labels. The number of synthetic minority points we generate is dependent on the difference in cardinality between the majority (negative) and minority (positive) sets. If the number of points in the positive class is more than or equal to the number of points in the negative class for a label, we consider the imbalance ratio to be 1 and do not add any synthetic minority points for that label. Our scheme is somewhat motivated by SMOTE, but the neighborhood exploration of our scheme is entirely different from SMOTE [12]. SMOTE relies on k-nearest neighborhood of points whereas our scheme uses a more versatile reverse-nearest neighborhood principles to identify the neighborhood of points. (We have included SMOTE in the comparative study. RkNN principles accommodate a varying number of neighbors for different points in a dataset. The RkNN count of a point depends on its proximity or presence in the neighborhood of remaining points in the search space. The significant improvement delivered by RkNN-based neighborhood exploration over k-NN based SMOTE is manifested there.)

Here, we describe the scheme for one label, say j. Let $\mathcal{D}$ be the training set. For $\mathcal{L}$ labels, the procedure is repeated $\mathcal{L}$ times to generate $\mathcal{L}$ sets of synthetic minority points, one for each label. Let $\mathbf{Majority}_j$ and $\mathbf{Minority}_j$ be the sets of majority and minority instances of label j respectively.

$$\mathcal{D} = \mathbf{Majority}_j \cup \mathbf{Minority}_j \tag{4.1}$$

Let N_j be the cardinality difference of $\mathbf{Majority}_j$ and $\mathbf{Minority}_j$.

$$N_j = |\mathbf{Majority}_j| - |\mathbf{Minority}_j| \tag{4.2}$$

To equate the cardinalities of the majority and minority sets, we will add N_j points in the synthetic minority set of label j. Here, $\mathbf{S}_j$ denotes the synthetic minority set of label j.

- **Reverse k-nearest neighbors of original minority points**: For each (minority) point $\mathbf{x}_i$ in $\mathbf{Minority}_j$, we will find its RkNN from the same set. Let $\mathbf{R}_j(\mathbf{x}_i)$ and $RC_j(\mathbf{x}_i)$ denote the set of reverse k-nearest neighbors of $\mathbf{x}_i$ and its cardinality respectively.

$$\mathbf{R}_j(x_i) = \{\mathbf{x}_k, \mathbf{x}_k \text{ is a RkNN of } \mathbf{x}_i,\ \mathbf{x}_k \in \mathbf{Minority}_j\} \tag{4.3}$$

$$RC_j(x_i) = |\mathbf{R}_j(\mathbf{x}_i)| \tag{4.4}$$

Since we are querying the minority points in context of $\mathbf{Minority}_j$, the RkNN count of each query point will be at least 1. We will mark the points whose RkNN count is exactly 1. Such points are isolated from the other points in the search space and do not have any RkNN other than itself. Let the set of such points for label j be $\mathbf{M}_j$.

$$\mathbf{M}_j = \{\mathbf{x}_i, \mathbf{x}_i \in \mathbf{Minority}_j \text{ and } RC_j(\mathbf{x}_i) = 1\} \tag{4.5}$$

As each member of $\mathbf{M}_j$ lies all by itself (without any minority neighbors in the vicinity) in the feature space, they do not serve as good candidates for adding the synthetic minority points and we do not consider them as seed points for adding the synthetic minority points. In a random dataset, it is quite possible that the original minority points do not find any minority reverse-nearest neighbor. In such a situation, we suggest to increase the value neighborhood size k and perform the operation. However, for all the datasets that we have included in the empirical study, we have not encountered this situation. Let $\mathbf{Aug\_Minority}_j$ be the set of points around which we will perform the oversampling. We compute

$\mathbf{Aug\_Minority}_j$ by removing the elements of $\mathbf{M}_j$ from $\mathbf{Minority}_j$.

$$\mathbf{Aug\_Minority}_j = \mathbf{Minority}_j \setminus \mathbf{M}_j \tag{4.6}$$

- **Generating synthetic minority points**: To generate a synthetic minority point, we random select a point from $\mathbf{Aug\_Minority}_j$, $\mathbf{q}$ (say) and find its RkNNs. We randomly select one of the RkNN of $\mathbf{q}$. Let that point be $\mathbf{r}$. We must maintain $\mathbf{q} \neq \mathbf{r}$, otherwise an original minority point will be selected in the synthetic minority set. Let d($\mathbf{q}$,$\mathbf{r}$) be the euclidean distance between $\mathbf{q}$ and $\mathbf{r}$. We generate a random value v such that $0 < v < \mathrm{d}(\mathbf{q}, \mathbf{r})$. The synthetic point, **new** is generated at a random distance v from $\mathbf{q}$ on the direction vector of $\mathbf{q}$ and $\mathbf{r}$.

  $$\mathbf{new} = \mathbf{q} + (\mathbf{r} - \mathbf{q}) * v \tag{4.7}$$

  We generate N_j number of such points to form $\mathbf{S}_j$, the synthetic minority set of label j. While selecting each synthetic point, we sequentially consider each point of $\mathbf{Aug\_Minority}_j$. This is done to maintain the uniformity of locations of the synthetic minority points. The same procedure is repeated across the entire label set to select $\mathcal{L}$ sets of synthetic minority points $\mathbf{S}_1,\ \mathbf{S}_2, \ldots, \mathbf{S}_{\mathcal{L}}$.

- **Classifier modeling**: We run the classifiers in first-order setting, one classifier for each label. For each label j, we get the oversampled instance set, $\mathbf{O}_j$ taking the union of original instance sets $\mathbf{Majority}_j$ and $\mathbf{Minority}_j$ along with the synthetic minority set $\mathbf{S}_j$.

  $$\mathbf{O}_j = \mathbf{Majority}_j \cup \mathbf{Minority}_j \cup \mathbf{S}_j \tag{4.8}$$

  We train a Linear SVM classifier for label j using $\mathbf{O}_j$ to get the classifier model $\mathrm{M_j}$. The label predictions of test set corresponding to label j are obtained by invoking $\mathrm{M_j}$. The procedure is repeated over L labels.

4.4 Proposed Approach – II

4.4.1 Multi-label nature of data, its consequences and our thoughts

A multi-label dataset is characterized by the membership of a set of feature points to more than one label. A class-imbalanced dataset is typified by the quantitative disproportion in the number of instances representing its classes. Multi-label datasets are often found to be class-imbalanced. In this work, we deal with binary multi-label dataset where each label can take exactly one of the two classes (1 - positive class and 0 - negative class). Typically, class 1 and class 0 are the minority and majority classes respectively. The class membership of each instance varies from label to label. An instance which is positive for some label A can be negative for some other label B. A similar phenomenon for all the instances will lead to a different combinations of positive and negative sets for each label (even though the union of the positive and negative set of instances is same for all labels). The quantitative consequence of this phenomenon is the variable degree of class imbalance across the labels. The key idea of this approach is to design an imbalance-informed scheme. For each label, we set an imbalance adaptive mis-classification cost framework different labels. The issue of differential class geometry is already taken care of in the previous section. **We may note that the integration of the information of class geometry and imbalance of the labels gives the best outcomes. We have included this study in the empirical evaluation section later in this chapter**.

4.4.2 Handling imbalance further - Cost sensitive classification

Cost-sensitive learning is one of the ways of handling imbalanced data. As stated earlier, in multi-label datasets, the degree of imbalance varies across labels. After generating the imbalance-informed representations for each data point, we proceed with a cost-sensitive linear SVM based classification. Let the misclassification cost of a minority instance to the majority class for label k be denoted by Cost_k. We denote the imbalance ratio of label k as $\mathrm{imb_k}$ To improve the detection of minority class (generally the positive class), for label k, Cost_k value is fixed to $cf \times (\log_2(\mathrm{imb}_k) + 1)$. cf is a cost-factor whose increasing value gives increasing misclassification cost for the minority class. Remark 5 discusses the details on choice of cf. The misclassification cost of a majority instance to the minority class is set to 1 for all labels. The misclassification cost Cost_k value increases with increase in imbalance value of a label and is adaptive to various and diversified ranges of imbalance values in a single dataset. For a label which has no imbalance or the imbalance ratio is 1, the misclassification costs of both classes (no class is minority or majority to be precise) is 1. For an imbalanced label with $\mathrm{Cost}_k > 1$, the misclassification cost of the minority instances to the majority class is greater than 1 and it increases with increase in imbalance value. Hence, we have an imbalance-informed misclassification cost for each label. The $\log_2$ function allows us restrict the misclassification costs within an admissible yet varying limit depending on the imbalance ratios.

4.4.3 Algorithm

Let the multi-label dataset be denoted by $\mathcal{D}$ and the number of class labels for $\mathcal{D}$ be $\mathcal{L}$. $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{Y}_i), |1 \leq i \leq n$, $\mathcal{Y}_i$ denotes class label vector of $\mathbf{x}_i\}$. $\mathcal{Y}_\mathrm{i} = \{y_{i1}, y_{i2}, \ldots, y_{il}\}$. y_{ij} is 1 when label j is positive for instance $\mathbf{x}_i$, otherwise the value of y_{ij} is 0. Let each $\mathbf{x}_\mathrm{i} \in \mathbf{R}^\mathbf{p}$. We randomly equi-partition $\mathcal{D}$ into a training set, $\mathcal{D}_{tr}$ and a test set, $\mathcal{D}_{te}$. Let $\mathcal{X}$ be the set of training instances (without the label information).

$$\mathcal{X} = \{\mathbf{x}_i, \text{i=1, 2,..., n}\} \tag{4.9}$$

We calculate class-imbalance ratio of each label j, $j = 1, 2, \ldots, l$ denoted by imb_j.

$$\mathrm{imb_j} = \frac{\text{Number of negative training instances for label } j}{\text{Number of positive training instances for label } j}$$
$$\Rightarrow \mathrm{imb_j} = \frac{||\{\mathbf{x}_i \text{ such that } \mathcal{Y}_{ij} = 0, i = 1, 2, \ldots, n\}||}{||\{\mathbf{x}_i \text{ such that } \mathcal{Y}_{ij} = 1, i = 1, 2, \ldots, n\}||}$$

$\mathbf{z}_{ij}$ is a $k_p + k_n$ dimensional vector or feature. Its first k_p components are generated by taking distance from the midpoints of the positive homogeneous edges and multiplying them with the imbalance ratio of label j. The remaining k_n components are computed by taking distances from the negative homogeneous edges.

Let $\mathcal{Z}_j$=$\{\mathbf{z}_{ij}, i = 1, 2, \ldots, n\}$. $\mathcal{Z}_j$ represents the transformed feature mapping of the training instances in $\mathcal{D}_{tr}$ for label j.

Let Min and Maj be the minority and majority classes of a label respectively. Let Cost_j(Min, Maj) and Cost_j(Maj, Min) denote the misclassification costs of a minority instance to the majority class for label j and vice versa. For each label, $\mathrm{Cost}_j(\mathrm{Min}, \mathrm{Maj})$ is equal to the product of a cost factor (*cf*) and logarithm of the it's imbalance ratio. In this work, we have fixed the value of cf to 1.

For each label j,

$$\mathrm{Cost}_j(\text{Min, Maj}) = cf \times \max((\log_2(\mathrm{imb}_j) + 1), 1) \tag{4.10}$$

$$\mathrm{Cost}_j(\text{Maj, Min}) = 1, \qquad i = 1, 2, \ldots, n \tag{4.11}$$

For each label j, we train a classifier $\mathcal{W}_j$ by invoking the original feature set $\mathcal{X}$

and the above defined cost function for label j. For classifying a test instance $\mathbf{t}$ with respect to label j, we invoke $\mathcal{W}_j$ to predict it's class. We have used linear SVM classifier implementation of LIBSVM ([11]) for modeling and classification.

4.5 Experimental Setup

We present the details of the empirical study in this subsection. Eleven real-world multi-label datasets were used. We have done some pre-processing for the class-imbalance focused learning. In all these datasets, we have removed the labels whose imbalance ratio (number of negative instances / number of positive instances) is more than 50 or the number of positive instances is less than 20. A similar protocol has been suggested in [103]. For the nominal datasets, we have performed reduction in the feature set according the same recommendation. The attribute information of the datasets with respect to this experiment is presented in Table 4.1. Since this work deals with differential class imbalance ratios of multi-label datasets, we have also showed the minimum (min IR), maximum (max IR) and average imbalance (avg IR) statistics of each dataset in Table 4.1. These datasets are obtained from MULAN [90] and MEKA [75] repositories.

Table 4.1: Description of Datasets for experiment on imbalance

Dataset	domain	att.type	(D)	(L)	(F)	$(L.Card)$	$(L.Uniq)$	min IR	max IR	avg IR
Corel5k	image	nominal	5000	44	499	3.522	3175	3.46	50	17.86
Enron	text	nominal	1702	24	1001	3.378	753	1.0	43.48	5.35
medical	bio-NLP	nominal	978	14	1449	1.075	42	2.67	43.48	11.24
Slashdot	text	nominal	3782	22	1079	1.134	118	5.46	35.71	10.99
Tmc2007	text	nominal	28596	15	500	2.100	637	1.447	34.483	5.848
CAL500	music	numeric	502	124	68	25.058	502	1.04	24.39	3.85
RCV1 Subset1	text	numeric	6000	43	472	2.458	574	3.34	50.0	15.15
RCV1 Subset2	text	numeric	6000	39	472	2.170	489	3.22	47.62	15.87
Emotions	music	numeric	592	6	72	1.869	27	1.25	3.0	2.15
Scene	image	numeric	2407	6	294	1.047	15	3.52	5.62	5.56
Yeast	biology	numeric	2417	13	103	4.233	189	1.32	12.5	2.78

For comparative analysis, we consider RAKEL ([88]), LIFT ([105]) and CLR ([24]) which are multi-label classifiers. In addition to that, we have also included COCOA [103] and RML [66]. COCOA specifically addresses the class-imbalance problem in multi-label datasets. MLKNN invokes a set of k-nearest neighbor based classifiers for multi-label datasets. Besides these, we have included a couple of methods - namely SMOTE ([12]) and Random Undersampling (USAM) which are dedicated to the general class-imbalance problem and used them in a multi-label setting.

We have included our both approaches — i] Proposed Method I – Reverse nearest neighborhood based oversampling and ii] Proposed Method II – Cost-sensitive learning adaptive to the imbalance ratios. **It should be noted that we have presented the results of Proposed Method II in conjunction with the extracted features corresponding to the second algorithm of the previous chapter. The intrinsic efficiency of the cost-sensitive learning paradigm of Proposed Method II is further investigated in the next experiment.**

For evaluating their performances we have employed Macro-averaging $\mathrm{F_1}$ and Macro-averaging AUC. They are described below.

- **Macro-averaging $\mathbf{F_1}$**: It calculates the average of F_1 values across all labels. Let tp_j, tn_j, fp_j and fn_j denote the number of true positive, true negative, false positive and false negative predictions for label j respectively.
  We calculate $\mathrm{F_1}$ for label j,
$$\mathrm{F_{1j}} = \frac{2 \times tp_j}{2 \times tp_j + fp_j + fn_j} \tag{4.12}$$
$$\text{Macro-averaging } \mathrm{F_1} = \frac{1}{\mathrm{L}} \sum_{\mathrm{j=1}}^{\mathrm{L}} \mathrm{F_{1j}}, \qquad \text{where L is the number of labels} \tag{4.13}$$

- **Macro-averaging AUC**: Let AUC_j be the AUC score for label j. We calculate the average AUC score of all labels in Macro-averaging AUC. Higher the value of Macro-averaging AUC, better is the performance of the classifier.
$$\text{Macro-averaging AUC} = \frac{1}{L} \sum_{i=1}^{L} \mathrm{AUC}_i \tag{4.14}$$

4.5.1 Second experiment: Competence of the imbalance adaptive misclassification cost

We analyze the utility of the proposed scheme of imbalance-adaptive misclassification cost in this study. We consider two first-order methods LIFT and BR in their default settings where the misclassification costs of the classes are equal. We compare their performances with an enhanced cost version of each of them, LIFT-cost and BR respectively, where the cost of misclassification of the minority instances is set according to the proposed scheme. Note that BR evaluates the intrinsic capability of Proposed Method II in addressing the class-imbalance problem of multi-label datasets. We evaluate the difference in performances using Macro-averaging $\mathrm{F_1}$ metric.

4.5.2 Third experiment: Parameter optimization

In this experiment, we have studied the effect of variation of cost factor and number of lattice points on class-imbalance focused multi-label learning. Cost factor is varied between 0.5, 1, 2 and 4. Variation of the number of positive and negative lattice points is also explored.

4.5.3 Fourth: Statistical significance test

We have conducted Wilcoxon Signed Rank Sum Test to measure the statistical significance of the difference in performance given by the proposed method, LIIML with respect to a competing method. In this work, we have a number of experiments and each is evaluated with more than one metric. Experiments 1 and 2 are the key ones for this work. We have conducted the statistical tests for these two experiments. We report the p value at which the performance of the two methods are different. Lower the p value, more significant is the difference or more certain we are about rejecting the null hypothesis. The null hypothesis assumes that the performance of two methods are same. p value 0.05 or 5% significance level is the standard threshold for rejecting or accepting a null hypothesis. We have used p value 0.05 as the threshold for statistical significance of difference.

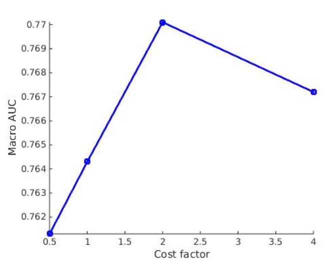
4.6 Results and Analysis

Experiment 1: Table 4.2 shows the performance of the proposed and compared methods on Macro-averaging F_1 and Macro-averaging AUC. Proposed Method II achieves the best score on a total of 8 out of 11 cases (**72.72%** cases) on Macro-averaging F_1. On one remaining dataset *Corel5k*, RML has obtained the best results of Macro-averaging F_1. On 2 datasets, Proposed Method I has performed best. On Macro-averaging AUC, Proposed Method II performs better than all other methods on 7 out of 11 datasets (**63.63%**). The remaining 4 best scores of Macro-averaging AUC are shared by COCOA (2), CLR (1) and Proposed Method I (1).

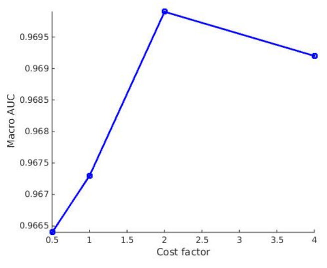
Table 4.2: This table records the observations of experiment on class-imbalance aspect of multi-label dataset (Experiment 2). Result are reported for 2 metrics (Macro-averaging F_1 and Macro-averaging AUC), 11 datasets and 9 methods (including LIIML). For both the metrics, a higher value means better result (as indicated by the ↑). On Macro-averaging F_1 and Macro-averaging AUC, LIIML has achieved best scores among all methods on 8 and 7 datasets respectively.

	Corel5k	Enron	Medical	Slashdot	TMC	CAL500	RCV1 Subset1	RCV2 Subset2	Emotions	Scene	Yeast
						Macro-averaging F_1 ↑					
USAM	0.143±0.004	0.263±0.011	0.673±0.013	0.260±0.008	0.606±0.003	0.216±0.006	0.357±0.006	0.342±0.005	0.594±0.012	0.621±0.007	0.431±0.008
SMOTE	0.131±0.003	0.263±0.005	0.671±0.018	0.323±0.006	0.608±0.003	0.238±0.005	0.312±0.003	0.308±0.004	0.586±0.021	0.618±0.005	0.431±0.003
RML	**0.215±0.007**	0.308±0.008	0.670±0.016	0.315±0.002	0.568±0.003	0.209±0.006	0.313±0.004	0.302±0.004	0.586±0.003	0.623±0.006	0.429±0.004
COCOA	0.197±0.003	0.327±0.007	0.690±0.011	0.326±0.009	0.668±0.003	0.208±0.010	0.362±0.006	0.339±0.008	0.665±0.014	0.731±0.010	0.455±0.014
LIFT	0.072±0.003	0.290±0.005	0.588±0.005	0.382±0.007	0.416±0.003	0.076±0.006	0.212±0.004	0.162±0.005	0.639±0.006	0.758±0.006	0.377±0.005
RAKEL	0.091±0.006	0.249±0.004	0.577±0.012	0.248±0.004	0.642±0.003	0.195±0.002	0.274±0.005	0.267±0.005	0.611±0.011	0.684±0.007	0.421±0.006
CLR	0.051±0.003	0.223±0.005	0.653±0.011	0.234±0.006	0.626±0.003	0.086±0.006	0.226±0.004	0.226±0.005	0.593±0.015	0.631±0.012	0.414±0.007
Proposed Method I	0.200±0.003	0.345±0.004	**0.768±0.005**	0.444±0.005	0.630±0.004	0.246±0.003	**0.478±0.003**	**0.475±0.002**	0.683±0.002	0.745±0.005	0.488±0.004
Proposed Method II	0.179±0.003	**0.367±0.006**	0.721±0.005	**0.477±0.010**	**0.672±0.005**	**0.255±0.006**	0.459±0.008	0.457±0.003	**0.692±0.004**	**0.766±0.005**	**0.509±0.004**
						Macro-averaging AUC ↑					
USAM	0.574±0.005	0.605±0.011	0.852±0.014	0.620±0.004	0.801 ± 0.004	0.515±0.003	0.675±0.011	0.673±0.009	0.707±0.015	0.792±0.007	0.579±0.006
SMOTE	0.601±0.006	0.619±0.007	0.872±0.006	0.685±0.006	0.789±0.003	0.514±0.004	0.623±0.006	0.621±0.005	0.701±0.008	0.771±0.012	0.589±0.008
RML	–	–	–	–	–	–	–	–	–	–	–
COCOA	0.717±0.003	0.735±0.005	0.955±0.004	0.732±0.004	0.930±0.002	0.554±0.004	0.890±0.003	0.884±0.003	0.842±0.007	**0.944±0.004**	**0.712±0.003**
LIFT	**0.742±0.003**	0.756±0.004	0.946±0.010	0.830±0.006	0.911±0.003	0.529±0.006	0.898±0.005	0.893±0.005	0.844±0.007	0.942±0.007	0.680±0.005
RAKEL	0.550±0.003	0.637±0.003	0.831±0.005	0.613±0.003	0.862±0.002	0.525±0.003	0.730±0.004	0.721±0.005	0.795±0.011	0.892±0.003	0.641±0.005
CLR	0.742±0.001	0.662±0.004	0.801±0.008	0.697±0.008	0.905±0.003	**0.559±0.003**	0.892±0.005	0.883±0.002	0.793±0.009	0.897±0.005	0.652±0.004
Proposed Method I	0.726±0.004	0.735±0.006	**0.977±0.007**	0.822±0.003	0.921±0.004	0.558±0.004	0.912±0.003	0.914±0.005	0.841±0.005	0.918±0.006	0.661±0.003
Proposed Method II	0.718±0.006	**0.765±0.003**	0.967±0.004	**0.838±0.005**	**0.923±0.006**	0.524±0.002	**0.920±0.002**	**0.918±0.004**	**0.847±0.003**	**0.944±0.004**	0.694±0.003

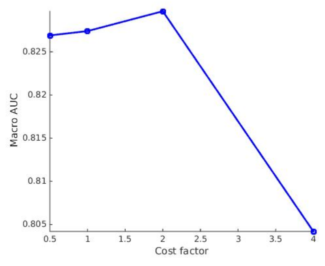
??



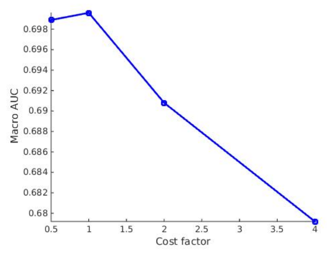
(a) Enron



(b) Medical

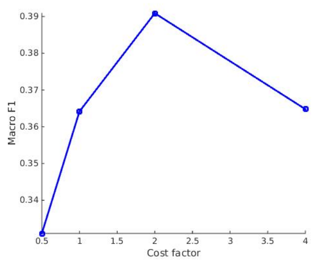


(c) Yeast

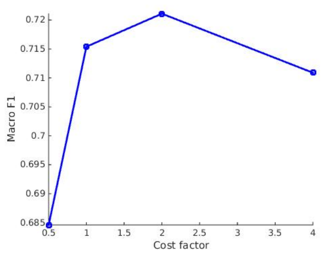


(d) Slashdot

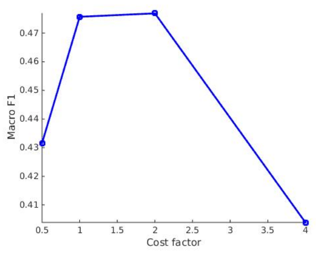
Figure 4.2: Macro-averaging AUC results of four datasets subject to varying and increasing misclassification costs for the minority class. We have varied the cost factor between 0.5, 1, 2 and 4. It can be observed that increasing the cost factor value upto 2 improves the learning of minority classes of each label. The graphs of these figures indicate a loss of performance on cost factor beyond 2 on all the four datasets. On using a value beyond 2, the classifier is getting over-biased towards the minority class. The optimal cost factor value is 2 for three datasets and equals 1 for one dataset.



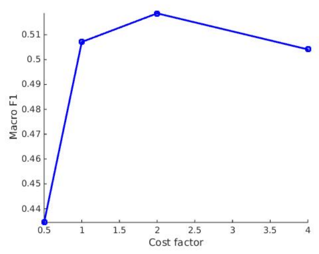
(a) Enron



(b) Medical

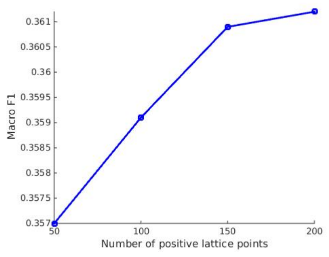


(c) Yeast

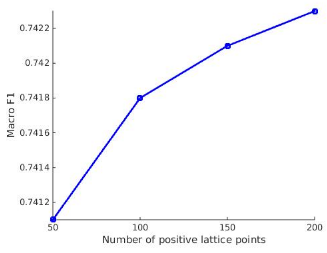


(d) Slashdot

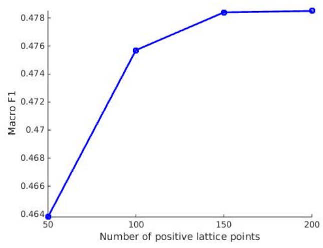
Figure 4.3: Macro-averaging F_1 results of four datasets subject to varying and increasing misclassification costs for the minority class. We have varied the cost factor between 0.5, 1, 2 and 4. The observation and analysis of this figure's data is in congruence with our findings from Figure 4.2. On all four cases, Macro-averaging F_1 value increases on increasing the cost factor value up to 2. It marks the optimal value for learning the given datasets. An increase beyond cost factor value 2 is observed to cause a loss of minority performance.



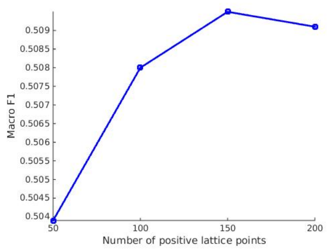
(a) Enron



(b) Medical

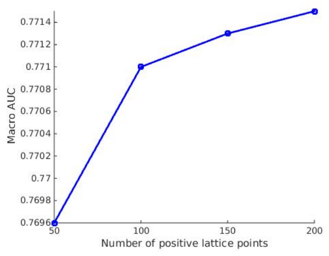


(c) Yeast

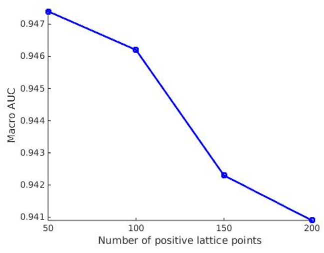


(d) Slashdot

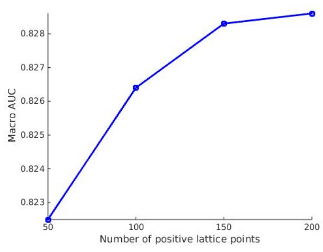
Figure 4.4: Macro-averaging F_1 results of four datasets subject to varying number of lattice points. We have varied the number of positive lattices between 50, 100, 150 and 200. Number of negative lattices is proportional to the number of positive lattice points and vary accordingly. The figures indicate that increasing the lattice points result in improvement in macro-averaging F_1 performance.



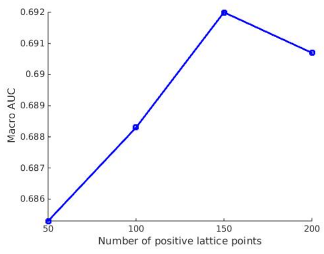
(a) Enron



(b) Medical



(c) Yeast



(d) Slashdot

Figure 4.5: Macro-averaging AUC results of four datasets subject to varying number of lattice points. We have varied the number of positive lattices between 50, 100, 150 and 200. Number of negative lattices is proportional to the number of positive lattice points and vary accordingly. For three datasets (Enron, Yeast and Slashdot) Macro-averaging AUC scores increase with increasing the lattice points. We get an exception with Medical dataset, where the performance degrades as number of lattices.

Table 4.3: This table records the results of applying the proposed cost-sensitive learning paradigm on two first order approaches LIFT and BR. We have used Macro-averaging F_1 and Macro-averaging AUC as the evaluating metrics. The original results (without added cost), results with enhanced cost and the corresponding improvement on each dataset are reported in the table. The instances where the enhanced cost version achieves an improvement of greater than 20% over the original result are highlighted through darkening of their backgrounds. — ↑ indicates higher is better and ↓ indicates lower is better, best outcome is indicated in bold-face

	Corel5k	Enron	Medical	Slashdot	TMC	CAL500	RCV1 Subset1	RCV2 Subset2	Emotions	Scene	Yeast
Macro-averaging F_1 ↑											
LIFT-with cost	0.154	0.397	0.704	0.463	0.643	0.245	0.387	0.379	0.682	0.758	0.490
LIFT	0.071	0.290	0.588	0.381	0.417	0.076	0.212	0.167	0.639	0.755	0.377
% of improvement	**116%**	**36%**	19%	**21%**	**54%**	**222%**	**82%**	**127%**	7%	0.4%	**30%**
Macro-averaging AUC ↑											
BR-with cost	0.206	0.357	0.797	0.459	0.583	0.275	0.482	0.486	0.651	0.674	0.476
BR	0.029	0.194	0.742	0.346	0.365	0.076	0.258	0.247	0.556	0.635	0.337
% of improvement	**610%**	**84%**	7%	**33%**	**60%**	**261%**	**87%**	**97%**	17%	6%	**42%**

Table 4.4: This table corresponds to the outcome of statistical tests on results of Experiment 1 (Table 4.2). It reports the p value at which LIIML's performance is statistically superior to that of a comparing method for a given metric. Each row corresponds to a method and each column to a metric. A lower p value more significant difference in performance. We have selected $p = 0.05$ as the threshold for statistical significance. Outcomes at which $p < 0.05$ are indicated in boldface. On Macro-averaging F_1, LIIML achieves statistical superior performance over 7 out of 8 methods. On Macro-averaging AUC metric, LIIML's performance is statistically superior to 4 out of 7 methods.

Metrics→ Methods↓	Macro-averaging F_1	Macro-averaging AUC
USAM	**0.003**	**0.003**
SMOTE	**0.003**	**0.003**
RML	**0.004**	–
COCOA	**0.006**	0.184
LIFT	**0.003**	0.075
RAKEL	**0.003**	**0.004**
CLR	**0.003**	**0.018**
Proposed Method I	0.424	0.308

Experiment 2: Table 4.3 records the *Macro-averaging F_1* scores of LIFT and BR in regular cost framework and enhanced misclassification cost framework (Proposed Method II). The results indicate certain effectiveness of the enhanced cost scheme in handling class-imbalance and recognition of the positive (minority) class of the multi-label datasets. *Macro-averaging F_1* performance of LIFT has improved by over 20% on 8 out of 11 datasets using the misclassification cost-enhancement learning. On BR, the improvement using this scheme is also pronounced as we witness the improvement in results by over 20% on 8 out of 11 datasets also. For two datasets *Corel5k* and *CAL500* the percentage of improvement is more than 100 (w.r.t both BR and LIFT).

Experiment 3: Figure 4.2 and Figure 4.3 show the variation of Macro-averaging AUC and Macro-averaging F_1 scores on varying ranges of cost factor. Increasing the value of the cost factor promotes the recognition of minority class instances at the cost of majority class performance dataset. Macro-averaging AUC and Macro F_1 scores also indicate the same. Increasing the cost factor beyond 2 results in sharp fall of Macro-averaging F_1 and Macro-averaging AUC scores for all four datasets. Figures 4.4 and 4.5 illustrate the variation of Macro-averaging F_1 and Macro-averaging AUC scores with varying number of lattice points. Unlike the cost factor, we could not establish a relation between number of lattice points and Macro-averaging AUC scores.

Statistical Tests: Table 4.4 shows the results of statistical significance testing on outcomes of Experiment 1. For analyzing results of Experiment 1, we have conducted 15 tests for 8 comparing methods (RML did not output Macro-averaging AUC scores) and 2 metrics. On 11 (73.33%) cases, Proposed Method II has achieved statistically superior performance. Note that, as indicated by this analysis, the performances of Proposed Method I and Proposed Method II do not differ statistically.

4.7 Summary

In this chapter, we proposed two divergent techniques for addressing the differential class-imbalance problem of multi-label datasets. The first technique works on the principles of reverse nearest neighborhood based minority oversampling. The second technique is basically extending the cost-sensitive learning framework in context of multi-label datasets. The methods are also extensible in conjunction with other classifiers. We have presented the results of Proposed Method II as the conjunction of the proposed cost-sensitive learning with the feature extraction method (Proposed Method II) of the previous chapter. Empirical evidences indicate that considerable improvement in performance has been delivered by both the methods. It is also evident that the performance of Proposed Method II is slightly better than that of Proposed Method I. Proposed Method I significantly outperforms the remaining multi-label classifiers on class-imbalance oriented metrics.

Chapter 5

Open Set Classification

5.1 Introduction

A conventional classification task aims to assign the instances to one out of the *known* classes whereas *unknown* class detection deals with recognition of the instances belonging to *unknown* classes in addition to the known ones. An unknown class is discriminated from the known classes on the basis of the non-availability of its (unknown class's) instances during the training phase. Though classification and detection are performed simultaneously by humans, machines often fail to accomplish the latter efficaciously. Perception and consequent detection of unknowns pose a serious challenge for the machine, which is designed to operate in a 'closed' world. Classifier design and presumptions made by us primarily account for such a disparity. We grow and learn in an unknown world with an incrementally growing known subspace. On the contrary, our classifiers are trained in a 'closed' setting of known distributions and classes. Furthermore, it is considered ideal when the training set and the test set have as similar distributions as possible. On assuming the above, a classifier is forced to restrict its prediction into the set of training classes. While predicting a test set consisting of seen and unseen class instances, the unseen instances get camouflaged as seen instances and thus get misclassified.

The above mentioned problems can be generalized as follows. At the training phase, we have instances belonging to any one of the c possible classes where $c \geq 1$. Unlike regular classification, during testing, the instances can be a member of any one of the $c+u$ classes, $u \geq 1$, the c *known* classes are seen during the training as well as test phase while the remaining u classes which constitute the set of *unknown* class/es appear in the test phase only. In an open set classification scenario, we have an universe. During training, we are provided information about only a few aspects (known classes) of the universe but in the test phase we have to classify what we have seen before (known classes) and detect the ones that that we have not encountered earlier (unknown classes). We may also have some classes which we do not encounter in either training or test phases. Openness of a dataset is the degree of unknownness in the dataset. For quantifying this characteristic the following definition is provided by [81].

$$Openness = 1 - \sqrt[2]{\frac{2 *|\text{Training classes}|}{|\text{Target classes}| + |\text{Test classes}|}}$$

The target classes consist of all the training and test classes as well as the leftover unknown classes that do not participate in the training and testing. The task of open set classification is much more complicated than simply rejecting the uncertain test points. Here, one has to correctly classify the known class points besides detecting the unknown instances. In unknown class detection, a significant fraction of points can come from the unseen class/es and follow a class structure. In such a scenario, for efficacious performance, one needs to balance and do well in both unknown class detection and known class classification. A scheme addressing this task should be self-contained, and be able to decide what it does not know, more favorably without any human intervention.

Extant classifiers predict 'closed' class-memberships in terms of the known classes only. As we have said earlier, a true open set solution has to possess the capability of saying 'no' or 'unknown' when a test point is coming from an unknown class. In this work, we attempt to answer this by raising a simple question. Instead of querying a test instance **p** about its nearest neighbors in a given search space, we query about the **reverse k-nearest neighbors of p**. Reverse k-nearest neighbors of an instance **p** are all those points in a given search space whose k-nearest neighborhood contains **p**. **Note that, when n is the total number of training points, p's RkNN count can be anything between n and** 0. When all the instances in the search space have **p** as one of its k-NN, R-kNN count of **p** is n and it indicates sufficient belongingness of **p** to the given search space. On the contrary, when **p** does not lie in any point's neighborhood, it indicates significant disharmony between **p** and all others members of the search space. The latter situation is our motivation for rejecting and extending the prediction into the unknown class. In this dissertation, we present a novel reverse k-nearest based classification scheme which performs simultaneous classification into the known classes as well as to the unknown class. The key aspect of our work is the simplicity of the scheme. We do not need the availability of any information other than the training or known class instances and their respective class labels. The proposed scheme does not require any distance based thresholding for demarcation of the known and unknown spaces. The only user-modulated parameter is neighborhood size k. In the next section, we discuss the technical aspects (inputs and outputs) of the problem of open set classification and also compare with other classification problems.

5.2 Open set classification

A closed set classifier makes its prediction within the set of classes that it encounters in the training phase. It assumes that all classes of the test data (queries) were well represented at the training phase. Closed set classifiers, mostly built on a Bayesian Optimal Posterior Probability model assume, that a fixed set of classes share the real space and it (the classifier) has to predict to any of these classes according to the class boundaries. If the number of classes is c, it computes $\mathrm{P}(\mathrm{C}_j|\mathbf{x})$ for $j = 1, 2, \ldots, c$ and assigns the query instance to the class i which gives the maximum value of $\mathrm{P}(\mathrm{C}_i|\mathbf{x}),\ i = 1,\ 2, \ldots, c$.

Open set classification is a type of classification problem where an instance belonging to the unknown class appears at the test phase. Unknown class denotes a class which had zero or no representation at the training time. An open set classifier can encounter instances from such un-represented class/es at the test phase and should rightfully predict them as 'unknown' instead of classifying them into the known classes. To yield admissible output, the modus operandi of an open set classifier should largely differ from that of a closed set classifier. As said in the previous paragraph, a closed set classifier can create the class boundaries during the training phase. But, the same mode of action by an open set classifier will lead to the assignment of the unknown class instances into any one of the known classes, which will contradict with their true classes. Open set classification is different from anomaly detection as well as incremental learning. In incremental learning, the goal is to add the newly encountered classes to the database of seen classes on encountering it's instance. On the contrary, in open set classification it is not desirable for us to add the unknown/ unrepresented classes in the seen domain. What is unknown should remain unknown but should be recognized as unknown. Anomaly detection is a task in which a rare event or observation like outliers is identified as different from the regular ones. In anomaly detection, we do not need to discriminate the known classes. Unlike anomaly detection, in open set classification, the classifier does not make any assumption about the cardinalities of the unknown/ unknowns, no information is available about the unknowns at the training time (as well as the test time). The problem of open set classification requires us to have provision for the unknown and

Table 5.1: Categorization of different problems

Task	Objective	Training Data	Prediction of test data
Closed set classification	Discrimination between the classes	Data from all classes	To any one of the existing classes
Anomaly Detection	Detecting abnormal data like outlier	Adequate normal class data, few outliers	Classification as outlier- yes or no
Incremental Learning	Dynamic classifier modeling in a changing world	Adequate and sequential training and test data	To any one of the existing classes as well as classifier updation
Open set Recognition	**Discrimination between known classes and identifying unknown class instances**	**Data from all known classes only**	**To any one of the known class or to the unknown class**

unrepresented class besides discriminating between, and correctly predicting the known classes. In an efficient open set classifier, two types (known classification and unknown prediction) of the results should be consequences of the scheme. Table 5.1 provides a tabular presentation of the same. Principles of reverse nearest neighborhood provide an elegant way of solving these two issues simultaneously. Our scheme based on reverse nearest neighborhood principles is presented in Section 5.5. We discuss the existing works on open set classification in the next section.

5.3 Literature Review

This work deals with open set classification using the principles of *reverse nearest neighborhood*. Reverse k-nearest neighbor (RkNN) principle has been used in various applications but RkNN based classification has not been implemented or addressed in any existing piece of work so far. Keeping in mind these two aspects, the literature review of this work is presented in two contexts. First, we discuss extant works in the field of open set classification. In the second part, we present a brief discussion on works that have used principles of reverse k-nearest neighborhood to achieve some machine learning goals.

Open set recognition in a mixed bag of seen and unseen classes has appealed to the data science community for quite some time. Despite the number of works not being numerous till date, the techniques applied are quite diverse. [30] implemented unknown class recognition through estimation of prior probability of the known classes and posterior probabilities for the known as well as unknown classes. One class classifiers which try to model a class only through its positive instances has been one of the foremost solutions to deal with open world problem. Though it is sufficient to deal with a setup having one known class and the rest as unknown class, the need for a more refined scheme which can tackle two or more known classes along with the unknown is natural. [81] addressed this issue by implementing open set recognition in the context of two known and the remaining as unknown class. They modified the conventional SVM for this. Besides drawing a decision boundary between the two known classes, [81] added one more hyperplane which separated the unknown class from the known subspace. The learning of the classifier model followed by incorporating Compact Abating Probability (CAP) is another solution. An amalgamation of extreme value theory and the probabilistic CAP model is implemented in [80] to classify the instances from the known class/es and subsequently recognize the unknowns. CAP model considers a decreasing confidence of class membership as one moves away from a known class instance into the unmarked space. Regions beyond a thresholded radius are subsequently categorized as the unknown or open space. In [40], a posterior probability estimator is implemented for each training class. A test instance is predicted into a known class only if the maximum probability surpasses the threshold. If none is found, the point is recognized as unknown. Distribution learning of the known classes through Extreme Value Theory (EVT) and incremental learning are incorporated in [78] to implement open set classification. Object detection under open set constraints are solved using the drop-out sampling approach in [59].

A few recent schema have incorporated neural networks to recognize samples from unseen classes along with classification of samples into seen or known classes. The scheme by [56] is based on an ensemble of Convolutional Neural Networks with a provision for open set recognition. It separates plant images from unknown non-plant images. Open set recognition through a weightless neural network has been explored in [10]. In [9], a neural network based classifier detects the unknown samples through comparison and computation of the similarity between the unknown data and the stored or bounded knowledge. [43] on the other hand proposed a theoretically sound method to estimate the 'sampling window' of the training data. Samples generated from regions outside the sampling window are used to represent the unknown world (class). They have trained a neural network to learn the known and unknown classes. In [2], a Generative Adversarial Network (GAN) based approach is to separate the differential identity components of face to generate an-identity preserving open set face synthesizer.

[58] has tweaked the traditional k-NN based classifier to facilitate open set recognition. It has proposed two schema. In the first variant, an instance is classified as unknown on non-agreement in class labels of its first two neighbors, while upon agreement the instance is assigned to the class of its first (as well second) neighbor. The second captures the

distances of the test instance from its two nearest neighbors belonging to different classes and calculates their ratio (nearer/ farther). If the ratio is beyond a threshold, the instance is classified as unknown and vice versa. [16] has employed a data fusion technique by integrating open-set graph-based optimum-path forest (OSOPF) classifier with genetic programming (GP) and majority voting fusion techniques for open set recognition. [101] explores the technique of classification-reconstruction learning for open set recognition.

Reverse k-nearest neighborhood might just seem a flip side of the k-nearest neighborhood, but it has been used to solve a number of data mining subtasks. Outlier detection in an unsupervised context and in data streams is implemented using reverse-nearest neighborhood by [69] and [67] respectively. Efficient reverse nearest search in metric spaces is achieved by [86]. [64] explored reverse nearest neighbor principles for protein information mining in bioinformatics. Problems on spatial data search are also addressed by the same in [41]. Reverse-nearest neighbor based algorithms have solved spatio-temporal query and range queries in [53]. A work by [7] has implemented a data clustering algorithm via RkNN. RkNN explores the locality of the instances to obtain meaningful data mining. In recent years, the techniques of local information exploration, feature embedding and lower rank and sparse subspace recovery have been used as a backbone in a number of diversified domains. In [110], a technique of adaptive embedded label propagation with weight learning is used for classification of real-world image datasets. For efficient classification of images, [111] integrates incorporation of embedded low-rank and sparse principal features with feature coding error and classification error. [109] uses analysis-based trained dictionary learning model for retrieval of query images. [112] is another important work on the same context. It introduces a structured and scalable dictionary learning framework to handle image analysis.

A technical elaboration of the backbone of our work, the *reverse-nearest neighbor principles* is presented in the next section.

5.4 Reverse k-nearest neighborhood

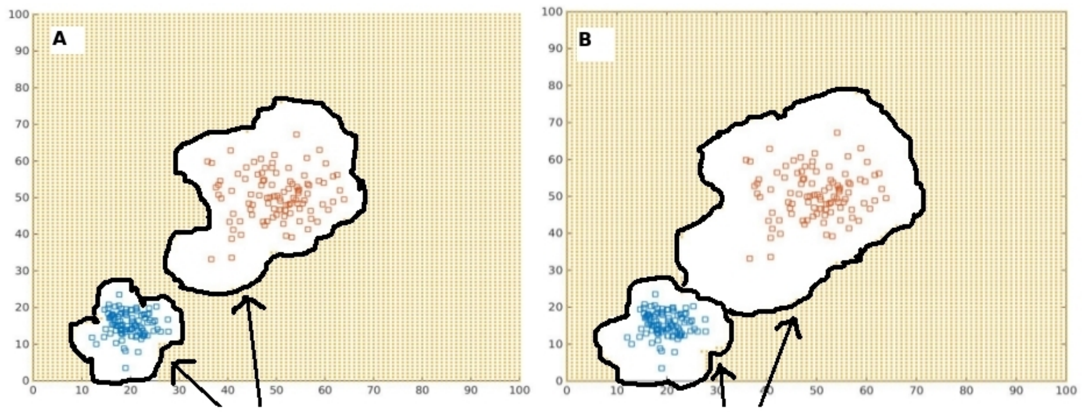
The principles of reverse nearest neighborhood is the backbone of this work. We have provided a definition of this principle in Definition 1 of the previous chapter (**Chapter 4**). Let us assume that we have a query point $\mathbf{p}$ in a given feature (search) space $\mathbf{p} \in \mathbb{R}^N$). We have points $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ ($X \subset \mathbb{R}^N$) in the search space. We also assume the neighborhood cardinality to be k.

Extant neighborhood estimators estimate the neighborhood of a query instance $\mathbf{p}$

through the distribution of the neighboring instances around it. Neighborhood demarcation is made via a surrounding hypersphere or through the encompassment of a fixed number of nearest instances around the query point. They do not take into account the locale of the query instance $\mathbf{p}$ in the neighborhood of the other search points. Reverse k-nearest neighborhood realizes the neighborhood paradigm in the latter light. To obtain reverse k-nearest neighbors of a query point $\mathbf{p}$, all points in a given search space are queried about their k-nearest neighbors to find if $\mathbf{p}$ is one of them. It is interesting to note that unlike k-NN (where a query point has exactly k neighbors), the number of RkNNs of query instance $\mathbf{p}$ can be anything between 0 and n (the search space cardinality). Depending on the data distribution, a query instance $\mathbf{p}$ can remain absent from the k-nearest neighborhood of all the instances in the training data, subsequently the RkNN count of $\mathbf{p}$ would be zero, if distance($\mathbf{p}, \mathbf{x}_\mathrm{i}$) $>$*distance of* $\mathbf{x}_i$ *from its* k^{th}*-nearest neighbor*, $\forall i$. The other extreme case arises when the query point $\mathbf{p}$ has the RkNN count of n, the size of search space by virtue of its presence within the k-nearest neighborhood of all the instances in the search space. $0 \leq$ RkNN count $\leq$ n is the possible range of RkNN values. For $\mathbf{p}$, its RkNNs constitute its neighborhood in the given search space X. More the RkNN count of $\mathbf{p}$ in X, more is its agreement with the instances in X. A zero RkNN cardinality indicates a significant disharmony between the query point $\mathbf{p}$ and the instances in the training set, and it will be fair to assume that $\mathbf{p}$ comes from an entirely different distribution. This is our principal motivation for predicting the unknown class instances (along with the usual prediction for the known classes) in a mixed bag of known and unknown class instances. Neighborhood size k is a critical component of our work. It is also the only parameter that the user needs to find out and fix to get the optimal performance on a dataset. In a later part of this chapter, Section 5.6.2, we have explained the procedure for selecting the k value for a particular dataset. We have carried out an empirical study on this aspect and reported our findings in Figures 5.37-5.40 and Section 5.8. The findings indicate that a single k value will not provide optimized performance across all datasets.

5.4.1 Known and unknown space modulation

According to our scheme, a region of positive RkNN count constitutes the known subspace (subspace covered by the known classes). We have a search space X (as defined in the previous subsection) and a query instance $\mathbf{p}$. Let $d_k(\mathbf{x}_\mathrm{i})$ be the distance of $\mathbf{x}_i$ from its k^{th} nearest neighbor in the given search space X (excluding itself). A hypersphere $S_{k\mathbf{x}_\mathrm{i}}$ of radius $d_k(\mathbf{x}_i)$, centered at $\mathbf{x}_i$ is assumed as the k^{th}-nearest neighborhood of $\mathbf{x}_\mathrm{i}$. $S_{k\mathbf{x}_\mathrm{i}}$



Red points and blue points come from class 1 and class 2 respectively. They come from two different distributions with different covariance values. As shown by arrows, the span of known class subspace around more compact class 2 points is much less than that spanned by the relatively sparser class 1 points.

Figure 5.1: This figure shows known-unknown subspace for a toy example.
Red points and blue points denote two different known classes. Let red points and blue points denote class 1 and class 2 respectively. Class 1 points come from a Gaussian distribution with $\mu = [50, 50]$ and $\Sigma = \left(\begin{smallmatrix} 49 & 0 \\ 0 & 49 \end{smallmatrix}\right)$. Similarly, Class 2 come from a Gaussian distribution with $\mu = [20, 15]$ and $\Sigma = \left(\begin{smallmatrix} 9 & 0 \\ 0 & 9 \end{smallmatrix}\right)$. 100 points from each of these classes are shown in the figures. White space denotes the known subspace and the yellow colored region denotes the unknown subspace. It can be noted that spread of known subspace around each class increases with the sparsity of the distribution. Class 2, being a dense class with lower value of sigma spans a smaller area representing the known subspace. On the other hand, known space volume around class 1 points is high since the relative distribution of the points is sparser. 1 Fig A and Fig B shows the known-unknown subspace delimitation at $k = 5$ and $k = 10$ respectively for the same set of data points. It can be noted that the known space volume increases with increasing the k value. At k value 10, known subspaces of the two classes expand and we get an overlap between the two.

constitutes the known space corresponding to instance $\mathbf{x}_\mathrm{i}$. If $\mathbf{p}$ lies inside $S_{k\mathbf{x}_\mathrm{i}}$, $\mathbf{x}_\mathrm{i}$ is a RkNN of $\mathbf{p}$. Let $d(\mathbf{p}, \mathbf{x}_\mathrm{i})$ be the distance between $\mathbf{p}$ and $\mathbf{x}_\mathrm{i}$. $\mathbf{p}$ can lie within $S_{k\mathbf{x}_\mathrm{i}}$ if $d_k(\mathbf{x}_\mathrm{i}) > d(\mathbf{p}, \mathbf{x}_\mathrm{i})$. Let S be the subspace that is covered by the known class.
$\mathrm{S} = \bigcup_{\mathbf{x}_\mathrm{i} \in X} S_{k\mathbf{x}_\mathrm{i}}$.

If $\mathbf{x}_\mathrm{i}$ is a vector in $\mathbb{R}^N$, then S is a subset of $\mathbb{R}^N$. Here, S implicitly defines the sampling window of the training data and hence can be viewed as defining the boundary of the known classes. The volume of $S_{k\mathbf{x}_\mathrm{i}}$ or the known subspace spanned by $\mathbf{x}_\mathrm{i}$ is dependent on $d_k(\mathbf{x}_\mathrm{i})$. In Figure 5.1, we scatter-plot 100 points each from two Gaussian distributions $\mathrm{N}_1(\mu_1, \Sigma_1)$ and $\mathrm{N}_2(\mu_2, \Sigma_2)$ where $\mu_1 = [50, 50]$, $\mu_2 = [20, 15]$, $\Sigma_1 = \left(\begin{smallmatrix} 49 & 0 \\ 0 & 49 \end{smallmatrix}\right)$ and $\Sigma_2 = \left(\begin{smallmatrix} 9 & 0 \\ 0 & 9 \end{smallmatrix}\right)$. The points of N_1 are labeled in red while the ones from N_2 are labeled in blue. The k^{th} nearest neighbor distance or $d_k(\mathbf{x}_i)$ for points in N_1 are usually greater than that of points in N_2. Accordingly, the points from N_1 span a larger volume of known space than that of N_2. Thus, the RkNN gives an automatic modulation of the known class spaces depending on the local distribution of the training data points. In Figure 5.1, the spaces marked with yellow color correspond to the unknown region. It is auto-adaptive to the class boundaries which vary from class to class. This is a desirable property while dealing with variable data distributions.

5.4.2 Principles of Reverse k-nearest neighborhood and classification

Mathematically, the principles of reverse-nearest neighborhood provides another approach of quantifying the neighborhood of the points. But reverse k-nearest neighbor principles have not been used for handling problems of classification. k-nearest neighborhood principles has a framework of classifying test data points. In k-nearest neighborhood based classifier, the confidence of the contending classes is calculated from the class membership of the k nearest neighbors. A test point is likely to belong to a class which has the highest number of it's (test point's) neighbors. The working principles of reverse k-nearest neighborhood is analogous to that of k-nearest neighbor's. We can easily extend a similar classification protocol using reverse k-nearest neighborhood. For a certain k value, we can find the reverse k-nearest neighbors of a test point $\mathbf{p}$ and give more confidence of classifying $\mathbf{p}$ to the class with highest number of reverse k-nearest neighbors. It is indeed true that getting a zero reverse k-nearest neighborhood is also possible. A RkNN based classifier has to possess proper strategies for handling the zero neighborhood count. The zero RkNN count allows us to solve the issue of open set recognition in a natural manner, hence we allow it as it is in our scheme.

The approach and its algorithm is elaborated in the next section.

5.5 Proposed Work

5.5.1 Approach

While classifying a test instance, classifiers operating on principles of density estimation predict the class having the highest density estimate (that is the class with the highest number of neighbors) as the test class. Now, let us assess their potential for addressing the classification task of an unknown class. For a window based classification paradigm [65], the number of neighbors inside the window volume can vary from zero to maximum cardinality of the search space. Though a zero neighbor count can be used for unknown class detection, when the density distribution is highly skewed, a single volume threshold is not expected to work well across the entire dataset. In addition to this, the volume thresholding is not automatic and needs empirical and manual modulations. In k-NN based classification motivated by [52], the k-nearest neighbors of a query point are searched in the training space. Consequently, k-NN classifier can predict only one of the known classes. There is no provision for unknown class detection in this scheme unless some thresholding is involved.

An efficient neighborhood based solution of open set detection should detect test instances which fall into zero neighborhood zones of a given feature space and subsequently reject them as unknown class instances. On a similar note, a positive neighbor count of a test instance indicates a finite known class membership and should be predicted to a class from the training instances. It is desirable that both these tasks (unknown class rejection and known class classification) should be done by the scheme without any thresholding. The scheme should be uniform as well as robust to non-uniform class distributions in a dataset. In order to design a scheme satisfying the said requirements, we propose a neighborhood based classifier where the neighborhood definition is a bit different from the one assumed in the above paragraph. Reverse k-nearest neighbors (RkNN) of a query instance $\mathbf{p}$ is searched in the training space X. When the RkNN count of $\mathbf{p}$ is zero, we classify $\mathbf{p}$ to the unknown class. In other words, if $\mathbf{p} \in \mathrm{S}^{\mathrm{C}}$ (the complement of the known subspace or sampling window, S), then $\mathbf{p}$ will likely belong to some unknown class. When RkNN count of $\mathbf{p}$ is > 0, then the class-specific membership scores are computed. Membership score of $\mathbf{p}$ for a class depends on the number of RkNNs count from that class and the distance between $\mathbf{p}$ and the nearest RkNN in that class. The membership value increases with increase in the RkNN count and a decrease in the distance of the nearest RkNN. The instance $\mathbf{p}$ is assigned to the training class with the highest membership score.

5.5.2 The Proposed Method

We have an open instance set D, consisting of two mutually exclusive partitions D_{tr} and D_{te}, representing the training set and test set respectively. The respective number of classes in D_{tr} and D_{te} are c and $c+u$. The extra u classes in D_{te} are the ones that remain unseen during the training. We consider u unseen classes together as a single unknown class resulting in $c+1$ classes for the test set, D_{te}. Classes $1, 2, \ldots, c$ correspond to the known classes and $c+1^{th}$ class corresponds to the unknown class. We also assume that the neighborhood size is a fixed positive integer k.

We will classify a test instance $\mathbf{p} \in \mathrm{D}_{\mathrm{te}}$ in $\mathbb{R}^N$ into any one of the known classes $1, 2, \ldots, c$ or to the unknown class, $c+1$ on the basis of the training set D_{tr} only.

Let $\mathrm{D}_{\mathrm{tr}}=\{(\mathbf{x}_{\mathrm{i}}, \mathbf{y}_{\mathrm{i}})|\ 1 \leq \mathrm{i} \leq \mathrm{n}\}$, where $\mathbf{x}_{\mathrm{i}}$ is a training instance vector in $\mathbb{R}^N$ and y_{i} is its corresponding class label. Hence, the number of training instances is n. The instances in D_{tr} belong to the known classes only, hence their memberships lie in $\{1, 2, \ldots, c\}$. Next we provide a stepwise description of the algorithm. Algorithm 6 depicts the same.

Step 1: We find the RkNN of $\mathbf{p}$ in D_{tr}. The outputs of the lookup are stored in $R_{\mathbf{p}}(.)$ and $M_{\mathbf{p}}(.)$.

$$R_{\mathbf{p}}(\mathrm{i}) = \begin{cases} 1, & \text{if } \mathbf{x}_{\mathrm{i}} \text{ is a } RkNN \text{ of } \mathbf{p} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

$$M_{\mathbf{p}}(\mathrm{i}) = \begin{cases} \mathrm{distance}(\mathbf{p}, \mathbf{x}_{\mathrm{i}}), & \text{if } \mathbf{x}_{\mathrm{i}} \text{ is a } RkNN \text{ of } \mathbf{p} \\ \infty, & \text{otherwise} \end{cases} \tag{5.2}$$

Remarks: $R_{\mathbf{p}}(\mathrm{i})$ is a vector which can take only two values 0 or 1. $M_{\mathbf{p}}$ is a vector in R^n.

Step 2: Now, we obtain the class-wise RkNN statistics for $\mathbf{p}$ in $N_{\mathbf{p}}(\mathrm{j})$ and $\mathrm{Mem}_{\mathbf{p}}(\mathrm{j})$, i=1, 2, $\ldots, j$. By 'class' only the seen training classes are meant. We calculate the distance of $\mathbf{p}$ from its nearest RkNN in class j and store the same in $N_{\mathbf{p}}(\mathrm{j})$. When $\mathbf{p}$ does not find a RkNN in class j, $\mathbf{p}$ is considered unreachable from the entire class j and $N_{\mathbf{p}}(\mathrm{j})$ is set to ∞. Next, we compute $\mathrm{Mem}_{\mathbf{p}}(\mathrm{j})$. It indicates the overall membership of $\mathbf{p}$ to class j. $\mathrm{Mem}_{\mathbf{p}}(\mathrm{j})$ depends on the RkNN count from class j as well as $N_{\mathbf{p}}(\mathrm{j})$, the distance from the nearest RkNN of $\mathbf{p}$ from class j.

$$N_{\mathbf{p}}(\mathrm{j}) = \{min(M_{\mathbf{p}}(\mathrm{i})); i = 1, 2, \ldots, n, \mathbf{y}_{\mathrm{i}} = \mathrm{j}, R_{\mathbf{p}}(\mathrm{i}) = 1\} \tag{5.3}$$

For each class j, $\mathrm{j} = 1, 2, .., c$, class membership score of $\mathbf{p}$, $Mem_{\mathbf{p}}(\mathrm{j})$ is calculated.

$$Mem_{\mathbf{p}}(\mathrm{j}) = \frac{1}{N_{\mathbf{p}}(\mathrm{j})} \sum_{\substack{\mathrm{i} \\ \mathbf{y}_{\mathrm{i}}=\mathrm{j}}}^{n} R_{\mathbf{p}}(\mathrm{i}) \tag{5.4}$$

Remarks : A higher value of class-specific RkNN count and smaller distance between $\mathbf{p}$ and the nearest class-specific RkNN indicates higher confidence of $\mathbf{p}$ to that class. A zero RkNN count from a class results in zero confidence of the instance to that class. Note that $\mathrm{Mem}_{\mathbf{p}}(\mathrm{j})$ could be greater than 1. By RkNN principles, even for the same k value, the neighbor count of different points vary (depending on their configurations). In such a scenario, it is difficult to adopt the distribution of their distances (as the number of neighbors would vary widely). So, in $\text{Mem}_{\mathbf{p}}(\text{j})$, we have considered the nearest neighbor distance from class j only.

A toy example of $\mathbf{Mem_p(j)}$ calculation: Let us have two classes A and B. Let the test point be **p**. We have the information about **p**'s RkNN counts as well as it's respective nearest neighbor distances from class A and class B also. Let the RkNN count from class A and class B be 2 and 3 respectively. Let the nearest neighbor distances $N_{\mathbf{p}}(\text{A})$ and $N_{\mathbf{p}}(\text{B})$ be 0.5 and 1 respectively.

$$\text{Mem}_{\mathbf{p}}(\text{A}) = \frac{1}{0.5} \times 2 = 4$$

$$\text{Mem}_{\mathbf{p}}(\text{B}) = \frac{1}{1} \times 3 = 3$$

This indicates the importance of nearest neighbor distance in our scheme. Though the RkNN count from class B is higher than that of class A, **p**'s class-membership to A is greater than that of B by virtue of the smaller distance. Besides k-reverse nearest neighbor configuration, the nearest neighbor's proximity from a class plays a decisive factor in computing the class-memberships.

Step 3: In this step, we will classify **p** to any one of the known classes $1, 2, .., c$ or to the unknown class on the basis of class membership scores. Max_Mem(**p**) value 0 indicates a zero RkNN count from entire set of known (training) classes. It indicates remoteness of **p** from the training classes and **p** is classified to the unknown class. Max_Mem(**p**) > 0 signifies the presence of **p** within some known class space and **p** is assigned to the class with Max_Mem(**p**). Class_prediction(**p**) gives the final prediction for **p**, it can be the unknown class or any one of the known classes.

$$\text{Max\_Mem}(\mathbf{p}) = \max_{\text{j}} \text{Mem}_{\mathbf{p}}(\text{j}), j = 1, 2, \ldots, c \tag{5.5}$$

$$\text{Max\_Mem\_class}(\mathbf{p}) = \text{argmax}_j \ \text{Mem}_{\mathbf{p}}(j) \tag{5.6}$$

$$\text{Class\_prediction}(\mathbf{p}) = \begin{cases} \text{unknown class} & , \text{Max\_Mem}(\mathbf{p}) = 0 \\ \text{Max\_Mem\_class}(\mathbf{p}), \text{otherwise.} \end{cases} \tag{5.7}$$

General Remarks:

1. *Not in the neighborhood of any*: Our scheme classifies an instance to the unknown class only when the instance does not possess any RkNN in the known training space. In other words, it does not lie in the k^{th}-nearest neighborhood of any training instance.

2. *Do we need to search the training space for each test instance?* A training instance $\mathbf{x}_\mathrm{i} \in \mathrm{D}_\mathrm{tr}$ can be a RkNN of a test instance $\mathbf{p}$ only if $d(\mathbf{x}_\mathrm{i}, \mathbf{p})$ is less than the k^{th} nearest neighbor distance of $\mathbf{x}_\mathrm{i}$. Here, we assume that k^{th} nearest neighbor search of each $\mathbf{x}_\mathrm{i}$ is done in the training space only once and stored for computations in the later stages. We conduct a single k-nearest neighbor search of the entire training set D_tr and find the distance of the k^{th} nearest neighbor of each training point $\mathbf{x}_\mathrm{i}$, , i=1,2, ...,n. Next, we only need to find and compare $d(\mathbf{x}_i, \mathbf{p})$ with k^{th}-nearest neighbor distance of $\mathbf{x}_\mathrm{i}$. If the former is lesser, $\mathbf{x}_\mathrm{i}$ becomes a RkNN of $\mathbf{p}$ and vice versa. Hence, the RkNN lookup of the entire test instance set requires just one k-NN search of the training set (in context of itself).

5.6 Experimental Setup

In this section,we propose a setup to make a comprehensive assessment of the proposed and competing method's performance on classification of the known classes and detection of the unknown class. A brief outline on the four essentials, namely *Datasets*, *Comparing methods*, *Parameter Optimization* and *Evaluating Metrics* is presented in the following subsections in order.

5.6.1 Datasets

We have employed ten real-world multi-class datasets to evaluate the relative efficacies of the proposed and compared methods. Table 5.2 summarizes the basic statistics of their attributes. The MNIST dataset is obtained from https://pjreddie.com/projects/mnist-in-csv/ while the source of the remaining ones is the Keel Dataset Repository [1]. MNIST dataset has 784 features and we obtain a Reduced-MNIST version by extracting the top features whose eigenvalue value summation covers 90% feature variance. Essentially, we perform PCA to extract the features. Reduced MNIST dataset has 79 features. We present the results of both MNIST and Reduced MNIST datasets individually in this

Algorithm 6 Reverse-nearest neighborhood based classification

Input: Training set $\mathrm{D_{tr}}$ with c known classes, Test point $\mathbf{p}$ which may belong to one of the c known classes or the unknown class (c+1).

Output: Class prediction of $\mathbf{p}$.

```
 1: Search for RkNN of $\mathbf{p}$ in $\mathrm{D_{tr}}$
        $R_{\mathbf{p}}(\mathbf{i}) = \begin{cases} 1, & \text{if } \mathbf{x}_{\mathrm{i}} \text{ is a } RkNN \text{ of } \mathbf{p} \\ 0, & \text{otherwise} \end{cases}$
        $M_{\mathbf{p}}(\mathbf{i}) = \begin{cases} \text{distance}(\mathbf{p}, \mathbf{x}_{\mathrm{i}}), & \text{if } \mathbf{x}_i \text{ is a } RkNN \text{ of } \mathbf{p} \\ \infty, & \text{otherwise} \end{cases}$
 2: for each class $j$ from $1, 2, .., c$ do
 3:     Calculate $N_j(\mathbf{p}) = min(M(\mathbf{p}, \mathbf{x}_{\mathrm{i}}))$, $i = 1, 2, \ldots, n$,
            $\mathbf{y}_{\mathrm{i}} = j$, $\mathbf{x}_i$ is a RkNN of $\mathbf{p}$
 4:     Calculate $Mem_j(\mathbf{p}) = \frac{1}{N_j(\mathbf{p})} \sum_{\substack{i \\ \mathbf{y}_{\mathrm{i}}=j}}^{n} R_{\mathbf{p}}(\mathbf{x}_{\mathrm{i}})$
 5: end for
 6: $Max\_Mem(\mathbf{p}) = \max_j Mem_j(\mathbf{p})$
 7: $Max\_Mem\_class(\mathbf{p}) = argmax_j \ Mem_j(\mathbf{p})$,
 8: if $Max\_Mem(\mathbf{p})$=0 then
 9:     Classify $\mathbf{p}$ as unknown (c+1).
10: else
11:     Classify $\mathbf{p}$ to $Max\_Mem\_class(\mathbf{p})$ (known class).
12: end if
13: End
```

work. These datasets are obtained in closed form that is they do not possess any openness and the class information of all the instances are known. In order to accommodate them for the purpose of open set recognition, we have generated an open version of each dataset following the same protocol as done by [80]. The first step is to set the cardinalities of the known and unknown classes. For MNIST and Letter datasets, we have followed the recommended partition (by [80]) of 6 known, 1-4 unknown classes and 15 known, 1-11 unknown classes, respectively. For the remaining datasets, the following protocol is adopted.

Let the non-open or regular instance set be denoted by D. $\mathrm{D} = \{(\mathbf{x}_i, \mathrm{y_i})|, \mathbf{x}_i \in \mathrm{X}, \mathrm{y_i} \in \mathrm{C}\}$, $\mathrm{X} \subset \mathbb{R}^N$ and $\mathrm{C} = \{c_1, c_2, \ldots, c_n\}$. Hence the number of classes in the dataset is n. We randomly equi-partition D into a training set D_{tr} and D_{te}. $\mathrm{D}_{tr} \cup \mathrm{D}_{te} = \mathrm{D}$ and $\mathrm{D}_{tr} \cap \mathrm{D}_{te} = \phi$. We will generate open training-test tuple $(\mathrm{D}^o_{tr}, \mathrm{D}^o_{te})$ from D_{tr} and D_{te}. We will select the sets of known classes and unknown classes, C_k and C_u respectively from C. The instances belonging to C_k will appear in both D_{tr}^{o} and D_{te}^{o} whereas the instances belonging to the unknown class set, C_u will appear in D_{te}^{o} only. The cardinality of C_k, denoted by c_k is fixed to $\lfloor 0.5 \times n \rfloor$. The cardinality of C_u, c_u is varied from 1 to $\lceil 0.5 \times n \rceil$. Here, we describe a procedure for generating $(\mathrm{D}_{tr}^{o}, \mathrm{D}_{te}^{o})$ at one particular value of openness.

1. $\mathrm{C}_k = \{$A set of c_k classes from C$\}$.

   $\bar{C}_k = \mathrm{C} \ - \ \mathrm{C}_k$.

2. For a given c_u, $\mathrm{C}_u = \{$A set of c_u classes from $\bar{C}_k\}$.

3. $\mathrm{D}_{tr}^{o} \ = \{(\mathbf{x}_\mathrm{i}, \mathrm{y}_\mathrm{i}), |\mathbf{x}_\mathrm{i} \in \mathrm{D}_{tr}$ and $\mathrm{y}_\mathrm{i} \in \mathrm{C}_k\}$.
   The instances in D_{tr} which belong to C_k goes to the open training set.

   $\mathrm{D}_{te}^{k} \ = \{(\mathbf{x}_\mathrm{i}, \mathrm{y}_\mathrm{i}), |\mathbf{x}_\mathrm{i} \in \mathrm{D}_{te}$ and $\mathrm{y}_\mathrm{i} \in \mathrm{C}_k\}$.
   D_{te}^{k} is the collection of test instances which belong to the known class/es.

   $\mathrm{D}_{te}^{u} \ = \{(\mathbf{x}_\mathrm{i}, \mathrm{y}_\mathrm{i}), |\mathbf{x}_\mathrm{i} \in \mathrm{D}_{te}$ and $\mathrm{y}_\mathrm{i} \in \mathrm{C}_u\}$.
   D_{te}^{u} is the collection of test instances which belong to the unknown class/es. We re-label the instances in D_{te}^{u} to the unknown class (instead of their actual class labels).

   $\mathrm{D}_{te}^{o} = \mathrm{D}_{te}^{k} \cup \mathrm{D}_{te}^{u}$.
   D_{te}^{o}, the open test set consists of the test instances belonging to the known classes as well as the unknown class.

By varying c_u in step 2, we vary the openness in D_{te}^{o}. In our experiments, for each dataset, we repeat step 1 and step 2 five times to generate 25 folds of open data (at each level of openness). After generating the open set versions, we calculate the openness value of each such partition using the formula proposed in [81].

Openness $= 1 - \sqrt[2]{\frac{2 * |\text{Training classes}|}{|\text{Target classes}| + |\text{Test classes}|}}$. Target class consists of all the training and test classes as well as the leftover unknown classes that do not participate in training and testing. An example is illustrated below in the **Openness Calculation Example**.

Remarks: As said earlier, we have followed the openness generation protocol similar to the state-of-the-art methods. We may note that the number of opennesses (openness as defined in the previous paragraph) generated for a dataset depends on the the number of classes it originally has. Following this, *Vehicle*, a dataset with 4 classes has 2 openness values (0.244 and 0.293) while *Texture*, a 11 class dataset has six openness values in the range 0.233-0.326. The same openness calculation formula stated in Introduction is used for these computations.

Openness calculation example: Let us consider the *Dermatology* dataset which has 6 classes. The number of target classes for this dataset is always 6. Following the above-mentioned protocol, we have 3 known classes and we will have 1,2 or 3 unknown classes at each stage.

3 known classes, 1 unknown class: Number of training classes=3. Number of test classes (known+unknown)= 4. Number of target classes=6. Following the formula, openness =0.225.

3 known classes, 2 unknown class: Number of training classes=3. Number of test classes (known +unknown) = 5. Number of target classes=6. Following the formula, openness =0.261.

3 known classes, 3 unknown classes: Number of training classes=3. Number of test classes (known+unknown)= 6. Number of target classes=6. Following the formula, openness =0.293.

5.6.2 Parameter Optimization

Most open set learners, including ours involve parameters whose values have to be determined empirically. The optimized values of these parameters are determined via cross-validation on the training set. We carve out a cross-validation training set, T and validation set V from D^o_{tr} only. For open set classification, we introduce openness in V following the same protocol as described in the above section. If m is the number of classes in D^o_{tr}, we fix the known class and unknown class cardinalities at $\lfloor 0.5 \times m \rfloor$ and $\lceil 0.5 \times m \rceil$ respectively.

Let us illustrate this with an example. Let there be 6 classes and 100 instances in D^o_{tr}. We randomly partition the D^o_{tr} into cross-validation training set, T and validation set, V. Each of T and V has 50 instances. We randomly choose 3 classes as known classes and the remaining 3 classes fall into the unknown class. We remove the instances from unknown classes in the training set T. In the validation set, instances from the known classes as well unknown classes are present.

Table 5.2: Description of datasets. N, f and C denote the number of instances, features and total number of classes in order. c_k and c_u denote the cardinalities of the known and unknown classes respectively.

Datasets	**N**	**f**	**C**	c_k	c_u
Dermatology	358	34	6	3	1-3
Letter	20000	16	26	15	1-11
MNIST	70000	784	10	6	1-4
Reduced-MNIST	70000	79	10	6	1-4
Optdigits	5620	64	10	5	1-5
Penbased	10992	16	10	5	1-5
Segment	2310	19	7	3	1-4
Shuttle	58000	9	7	3	1-4
Texture	5500	40	11	5	1-6
Vehicle	846	18	4	2	1-2
Vowel	990	13	11	5	1-6

To optimize N parameters, we perform an N-dimensional grid search on the training set validation set tuple (T,V) and select the parameter value/s giving the best output on the validation set. *Accuracy* is used for evaluation of performance.

5.6.3 Comparing methods

Open set recognition and classification have been accomplished efficaciously by a number of works in the past few years. For comparative assessment of performance of our scheme, we have selected five methods which are briefly described next.

- *1-vs-set, [81]*: It is a baseline method in the field of open set recognition. The recommended version of "1-vs-all" is chosen for comparison.
- *WSVM, [80]*: This is possibly the best performing open set learner so far. But LETTER and MNIST datasets are run on the recommended values of $C = 2$, $\gamma = 2$,$\delta = 0.1$ and $C = 2$, $\gamma = 0.03125, \delta = 0.1$ respectively. For the remaining datasets, γ and C values are selected via two-dimensional grid search. As recommended in the paper, threshold probability, P is set to *0.5*openness.*
- *Multi-class probability of inclusion, PI-SVM [40]*: Probability of inclusion or into the class probability is the foundation of this work. '1-vs-rest' binary SVM with threshold probability, P value *0.5*openness* is considered for execution. Similar to [80], tuning of γ and C are required for this scheme. LETTER and MNIST datasets are run on the recommended values of $C = 2$, $\gamma = 2$,$\delta = 0.1$ and $C = 2$, $\gamma = 0.03125, \delta = 0.1$ respectively. For the remaining datasets, parameters are fixed

through grid search.

Nearest neighbor distance-ratio open set classifier by [58] has addressed open set recognition through a tweaked knn classifier. They proposed two slightly different schema which stand apart from each other in terms of performance. Since the interest of this work lies with classification through nearest neighborhood, we consider both versions for comparison.

- *Nearest neighbor distance-ratio open set classifier (OSNN-CV)* : An instance is classified as unknown when getting a class mismatch between its two nearest neighbors. No user defined parameter is involved.
- *Nearest neighbor distance-ratio open set classifier (OSNN-NDR)*: The distance between two nearest neighbors belonging to different classes are noted for a test instance. If the ratio of the distance (nearer to farther) is sufficiently large, the instance is classified as unknown. The ratio of the two distances (nearer/ farther) is computed and compared with a threshold, namely T. For unknown class classification, T threshold range suggested by the authors is between *0.5* and *1.* Through parameter optimization, a single value is selected from $0.5, 0.55, \ldots, 1$ for each dataset.
- The proposed method: The proposed scheme requires tuning of the neighborhood k. The value of k is chosen via cross-validation on the training set.

5.6.4 Evaluating indices

In this piece of work, we deal with learners which detect *unknown* class instances alongside the usual classification of instances into one of the known classes. *Accuracy*, *Average F_1 over known and unknown classes ($AKUF_1$)* and *Known class F_1* are employed to provide insight into known class classification as well as unknown class detection. Before going into the details, we describe notation.

The class of *known* classes (known or training classes taken together) is considered the positive class and the set of classes absent during training or the *unknown* class is dubbed as negative. Let the known classes set be, $\mathrm{K} = \{1, 2, \ldots, c\}$ and the unknown class label be $c + 1$. A *true positive* prediction denotes that the classifier prediction is correct and the actual class is any one among $1, 2, \ldots, c$. In a similar fashion, a *true negative* is a correct prediction and the actual class is $c + 1$, the unknown class. A *False*

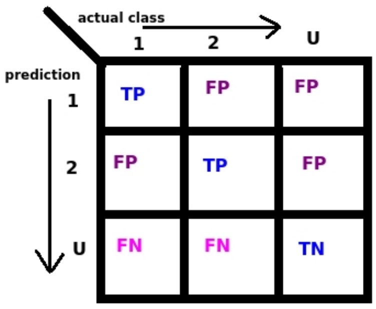


Figure 5.2: It depicts TP, TN, FP, FN for a toy scenario which has 2 known classes and an unknown class. Class 1 and class 2 constitute the set of known classes and U denotes the unknown class. The first two diagonal elements correspond to the correct predictions for class 1 and class 2 and belonging to the TP set. The 3^{rd} diagonal cell corresponds to the correct predictions for the unknown class U and hence is counted as TN. Remaining elements of row 1 and 2 correspond to the FPs or false predictions into the known classes. For example, cell(2,1) counts the cases where the actual class is 1 but the prediction has been class 2. For cell(2,U) the actual class of the instances is unknown class U but class 2 is predicted. Non-diagonal elements of row 3 correspond to the cases where prediction has been made into the unknown class U but actual class is a known class (1 or 2).

positive prediction is incorrect and the prediction is between 1 and c. There can be two cases of a false positive prediction — true class is the *unknown* class but the learner has misclassified into a *known* class. The other possible case is when the true class is some known class 1 (say) but the prediction has been made into some other known class 3 (say). *False negative* denotes that an instance from a *known* class has been incorrectly classified into the *unknown* class. The total and individual counts of *true positive, true negative, false positive* and *false negative* are represented as *TP, TN, FP* and *FN* respectively. An example is illustrated in Fig 5.2.

- *Accuracy*: For evaluating the classification performance of a learner, accuracy is the primary choice. It measures the fraction of correct predictions against the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.8}$$

  Intricate details like individual class performance of a learner cannot be deduced from accuracy alone. The next metric is employed to address the same.

- *Known class F_1*: This particular measure estimates the efficiency of the schemes in correct classification of the known class instances in a mixed bag of known and unknown instances.

- *Average F_1 over known and unknown classes ($AKUF_1$)*: In order to address the limitation of the above and provide a better glimpse of the class performances, AKUF$_1$ is computed. F_1 is measured for a single class where the possible classes can be more than one. F_1 calculates the harmonic mean of precision and recall for the concerned class. Below, the F_1 calculation for the positive class is demonstrated.

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F_1 &= \frac{2 \times Precision \times Recall}{Precision + Recall} \end{aligned} \tag{5.9}$$

  In the context of open set recognition, classes are broadly classified into *known* and *unknown* and the two are equally significant. F_1 is individually calculated on the *known* (positive) class as well as the *unknown* (negative) class. Mean of the above two are computed as the AKUF$_1$ and interpreted to evaluate the overall performance of the learners. A similar metric has been used by [58].

5.7 Results and Discussion

This section of the paper is devoted to the summarization and comparative analysis of the experimental results. Before proceeding to the discussion, we would like to clarify the figurative layout. The empirical results are obtained with different openness values where the range of openness varies across datasets depending on the number of classes. For LETTER and MNIST datasets, we have set the known class and unknown cardinalities according the experimental protocol of [80]. For a proper presentation, we have provided *three* graphical layouts for each dataset, one each for three evaluating metrics, namely *Accuracy*, *Average F_1 over known and unknown classes* (AKUF$_1$) and *Known class F_1*. Results on AKUF$_1$ are presented in Figure 5.4 to Figure 5.14). Figures 5.15-5.25 and Figures 5.26-5.36 show the *Accuracy* and *Known class F_1* plots respectively. Additionally, we have presented the summarized results in three tables 5.3, 5.4 and 5.5. Each table is dedicated to a metric and reports the number of best results obtained by each method on each dataset. A total of 50 scenarios or opennesses have arisen for the 10 datasets and the corresponding results are reported in the tables. In the following three paragraphs, we discuss the comparative performance of the methods on accuracy, AKUF_1, and Known class F_1 in order with reference to their corresponding plots and tables.

Accuracy is a primary choice when one has to evaluate a classifier. Table 5.3 records the number and percentage of best performance delivered by each of the comparing methods. Out of the 50 cases, the proposed method delivers best results on 39 scenarios (78%), followed by 6 (12%), 4 (8%) and 1 (2%) scenarios by WSVM, PI-SVM and 1-vs-Set respectively. On MNIST and Dermatology datasets, the proposed method achieves best result on 1 (25%) out of 4 opennesses and 1 (50%) out of 2 opennesses. On all other datasets, performance of the proposed method is better at more than 50% of the cases. For lower values of opennesses, the known class/es play a key role in determining accuracy value while the growing unknown class contributes more at higher opennesses. From the figures, it can be seen that the accuracy achieved by the proposed method is fairly constant across increasing openness. This indicates the robustness of the proposed scheme to variable opennesses. Figures 5.15 to 5.25 portray the graphical portrayal of accuracy performances delivered by the methods against increasing openness. These plots show the accuracy scores of the methods at different opennesses. Let us analyze Fig 5.15, which corresponds to the *Dermatology* dataset. At openness value 0.2257, performance of 1-vs-Set is best among the lot, for the remaining opennesses (openness values 0.2614 and 0.2929) the performance of proposed method is better than all the comparing methods. The betterment in performance by the proposed method is more at 0.2614 than at 0.2929.

Now, we analyze the relative capability of the proposed method to correctly predict the known class instances or *Known class* F_1. In practical scenario, this metric holds significance since its mimics the real world where we predict known things in a known and unknown world. Table 5.5 records the data of best outcomes on each dataset and its respective opennesses. Similar to the previous two measures, the proposed method gets the major share 76% (38 out of 50) best outcomes. Remaining 24% is shared by WSVM (8%, 4 out of 50), PI-SVM (10%, 5 out 50) and OSNN-CV (6%, 3 out of 5). Detailed known class F_1 values are available in Figures 5.26-5.36. Known class F_1 performance on *Dermatology* dataset is shown in Figure 5.26. At openness values 0.2257 and 0.2614, the proposed method performs best. PI-SVM scores best on the remaining openness (0.2929).

AKUF_1 performance is similar. Figures 5.4 to 5.14 present the AKUF_1 performance of the methods over increasing openness. These figures plot the actual outputs given by the proposed and compared methods. Let us analyze Fig. 5.4 (*Dermatology* dataset). It

Table 5.3: Performance on **Accuracy**. The table gives the summary of the best performances obtained by each method on each dataset.

Dataset (# of Opennesses)	Methods											
	Proposed Method		1-vs-Set		WSVM		PI-SVM		OSNN-CV		OSNN-NDR	
	# of Wins	Win%	# of Wins	Win%	# of Wins	Win%	#of Wins	Win%	# of Wins	Win%	#of Wins	Win%
Dermatology (3)	**2**	**66.67%**	1	33.33%	0	0	0	0	0	0	0	0
Letter (11)	**8**	**72.72%**	0	0	3	27.27%	0	0	0	0	0	0
MNIST (4)	1	25%	0	0	**2**	**50%**	1	25%	0	0	0	0
Optdigits (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Penbased (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Segment (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Shuttle (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Texture (6)	**5**	**83.33%**	0	0	1	16.67%	0	0	0	0	0	0
Vehicle (2)	**1**	**50%**	0	0	0	0	**1**	**50%**	0	0	0	0
Vowel (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Total	**39/50**	**78%**	1	2%	6/50	12%	4/50	8%	0	0	0	0

Table 5.4: Performance on **Average F_1 over known and unknown classes (AKUF_1)**. The table gives the summary of the best performances obtained by each method on each dataset.

Dataset (# of Opennesses)	Methods											
	Proposed Method		1-vs-Set		WSVM		PI-SVM		OSNN-CV		OSNN-NDR	
	# of Wins	Win%	# of Wins	Win%	# of Wins	Win%	#of Wins	Win%	# of Wins	Win%	#of Wins	Win%
Dermatology (3)	**3**	**100%**	0	0	0	0	0	0	0	0	0	0
Letter (11)	**9**	**81.81%**	0	0	2	18.18%	0	0	0	0	0	0
MNIST (4)	0	0%	0	0	**3**	**75%**	1	25%	0	0	0	0
Optdigits (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Penbased (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Segment (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Shuttle (4)	**2**	**50%**	0	0	1	25%	1	25%	0	0	0	0
Texture (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Vehicle (2)	**2**	**100%**	0	0	0	0	0	0	0	0	0	0
Vowel (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Total	**40/50**	**80%**	0	0	6/50	12%	4/50	8%	0	0	0	0

Table 5.5: Performance on **Known class** F_1. The table gives the summary of the best performances obtained by each method on each dataset.

Dataset (# of opennesses)	Methods											
	Proposed Method		1-vs-Set		WSVM		PI-SVM		OSNN-CV		OSNN-NDR	
	# of Wins	Win%	# of Wins	Win%	# of Wins	Win%	#of Wins	Win%	# of Wins	Win%	#of Wins	Win%
Dermatology (3)	**2**	**66.67**%	0	0	0	0	1	33.33%	0	0	0	0
Letter (11)	**7**	**63.63**%	0	0	2	11.11%	0	0	2	18.18%	0	0
MNIST (4)	1	25%	0	0	**2**	**50%**	0	0	1	25%	0	0
Optdigits (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Penbased (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Segment (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Shuttle (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Texture (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Vehicle (2)	0	0	0	0	0	0	**2**	**100%**	0	0	0	0
Vowel (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Total	**38/50**	**76%**	0	0	4/50	8%	5/50	10%	3/50	6%	0	0

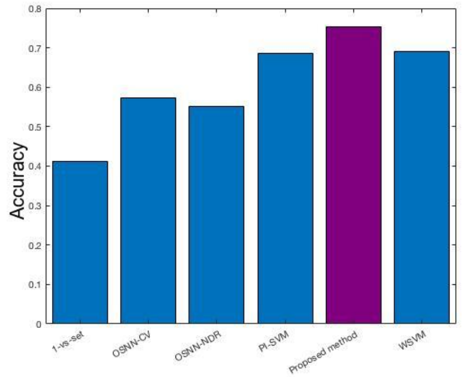
can be observed that the performance of the proposed method lies above all others at all three openness values. But the degree of improvement over the other methods is more pronounced at openness values 0.2257 and 0.2929. Similar analysis for all datasets can be be made by consulting the remaining figures. Table 5.4 shows the overall statistics of best AKUF_1 performance by the methods. The proposed method delivers the best performance on 50% or more cases for all but one dataset. Out of the 50 cases, the proposed method wins in 40 cases (80%) followed by 6 (12%) and 4 (8%) cases by WSVM and PI-SVM respectively. These figures indicate the capability of the proposed scheme in correctly predicting the known classes as well as the unknown class.

Comparative results presented in the above three paragraphs indicate the efficaciousness of the proposed scheme in both known and unknown aspects of open set learning. The proposed method maintains its superior performance on datasets with lesser number of classes (*Dermatology*, *Vehicle*) as well as on datasets with a larger number of classes (*LETTER*, *Vowel*, *Texture*). The intrinsic multi-class framework of the proposed scheme accounts for this robustness.

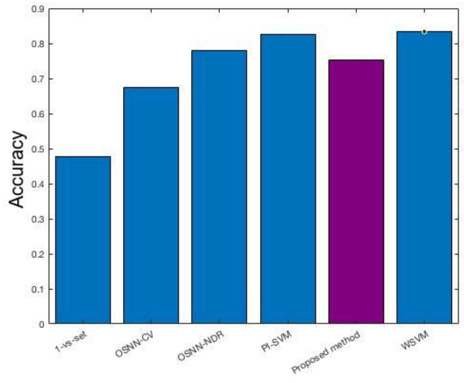
The performance of the proposed method on MNIST dataset is not as good as a couple of methods (namely WSVM and PI-SVM). Moreover, it also shows a deviation from it's own (proposed method's) performance on the remaining datasets. We investigated the loss of performance on MNIST dataset and our findings point to the high-dimensionality of this dataset. Our method is based on RkNN principles where distance and neighborhood relations are the only information that we cultivate for classification. Our method suffers from curse of dimensionality at 784 features and failed to perform as competently as on the remaining datasets. To validate our findings, we have generated outputs on a reduced version of MNIST dataset. The Reduced-MNIST version is obtained by extracting the top features which covers 90% feature variance. Reduced MNIST dataset has 79 features. Fig 5.7 shows the AKUF_1 performance of proposed and comparing methods on Reduced MNIST. It shows that the performance of the proposed method is better than that of all others. The results are also superior to that of the best performing methods (WSVM and PI-SVM) on regular MNIST (with all features) of 784 features (Refer to Figure 5.6 (for MNIST) and 5.7 (for Reduced MNIST)). Figures 5.18 and 5.29 show the accuracy and known class F_1 results of these experiments. The results are in congruence with AKUF_1 performance.

5.7.1 Reporting average (over all opennesses of a dataset) AKUF$_1$ results of five datasets

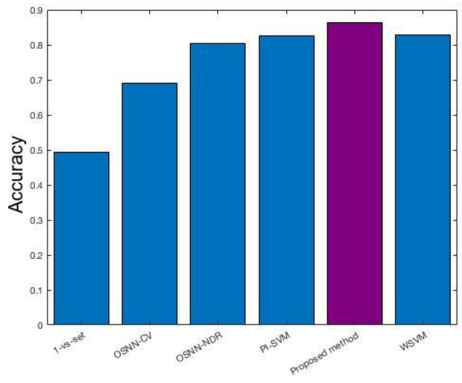
For five datasets (Dermatology, MNIST Reduced-MNIST, Optdigits and Segment), we calculate the average of AKUF$_1$ scores over the various openness values (of each dataset). In Figure 5.3, we plot the average AKUF$_1$ results of the proposed method and the five competing methods. The results indicate the certain superiority of the proposed method over all five comparing methods (including neighborhood based openset classifiers OSNN-CV and OSNN-NDR) in all datasets except MNIST.



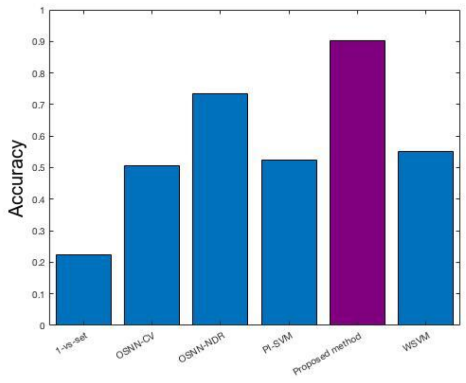
(a) Dermatology



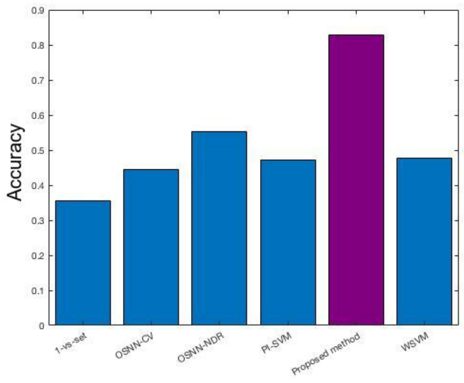
(b) MNIST



(c) Reduced-MNIST



(d) Optdigits



(e) Segment

Figure 5.3: The results indicate the certain superiority of the proposed method over all competing methods on four out of five datasets. On MNIST dataset, the proposed method suffers from the issue of high dimensionality of features. The enhanced performance of the proposed method on Reduced-MNIST (with reduced feature set) dataset affirms this fact.

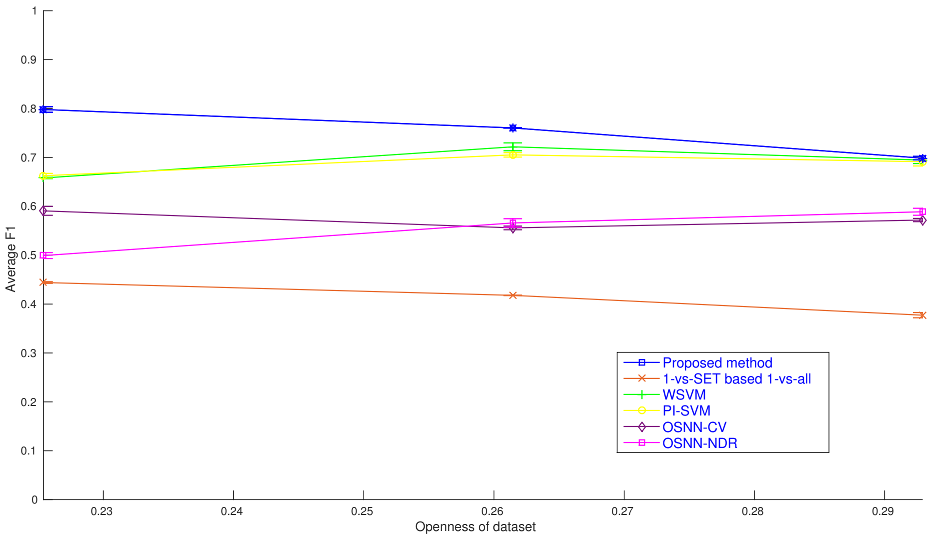


Figure 5.4: AKUF_1 results on *Dermatology* on *three* openness values.

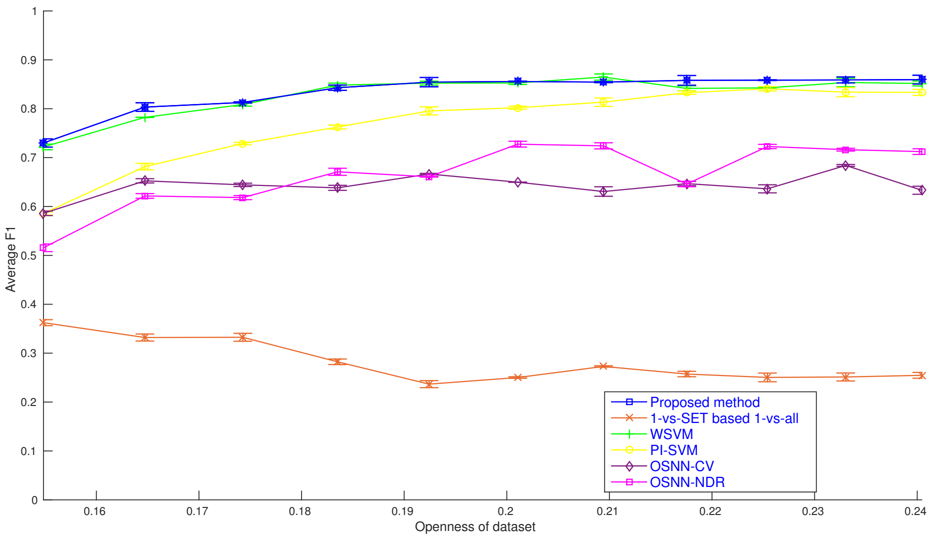


Figure 5.5: AKUF_1 results on *Letter* on *eleven* openness values.

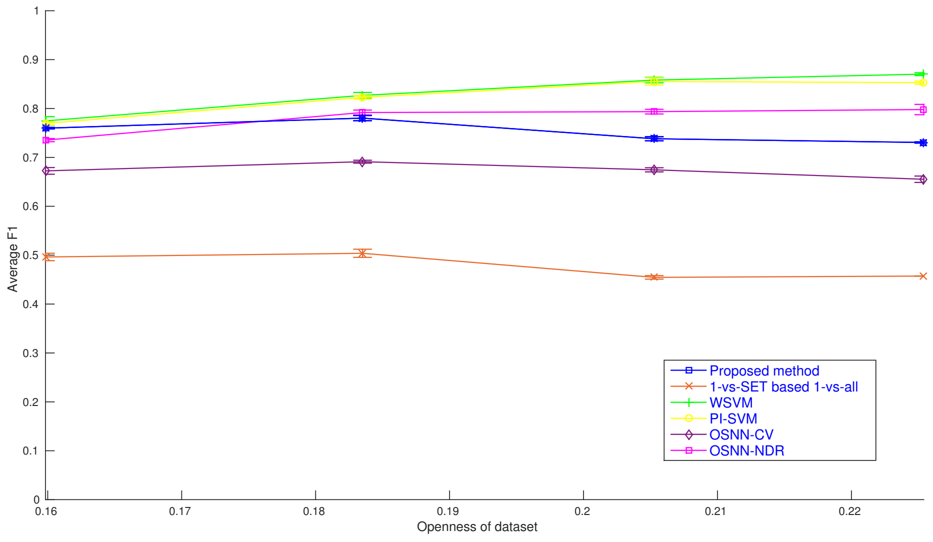


Figure 5.6: AKUF_1 results on *MNIST* on *four* openness values.

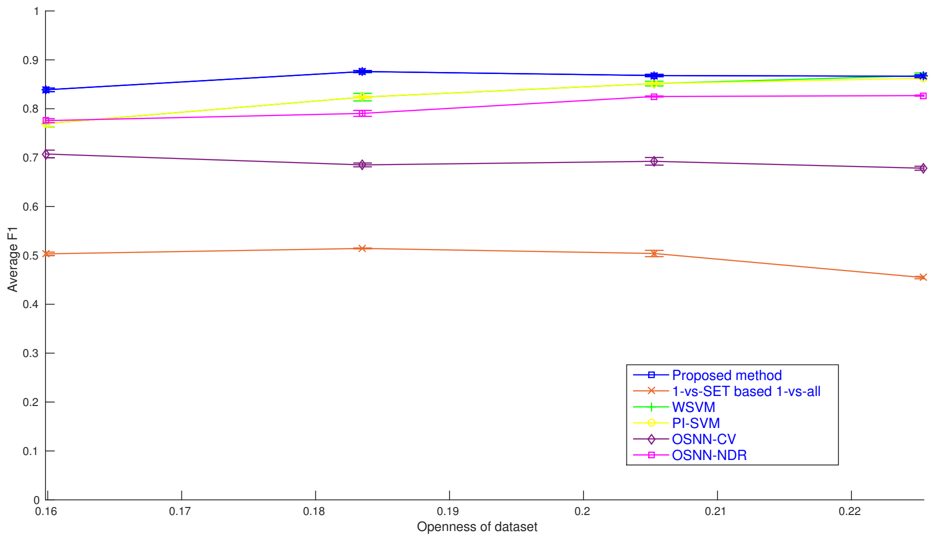


Figure 5.7: AKUF_1 results on *Reduced-MNIST* on *four* openness values.

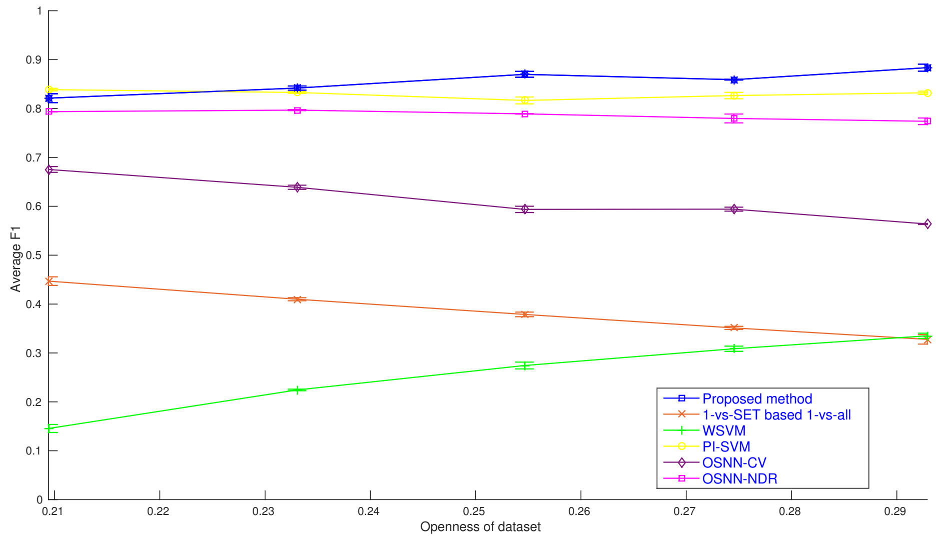


Figure 5.8: AKUF_1 results on *Optdigits* on *five* openness values.

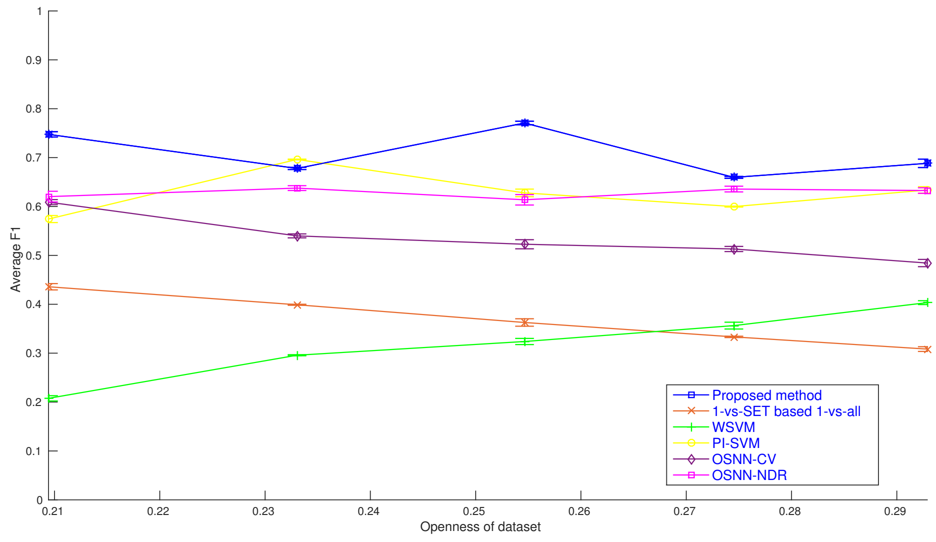


Figure 5.9: AKUF_1 results on *Penbased* on *five* openness values.

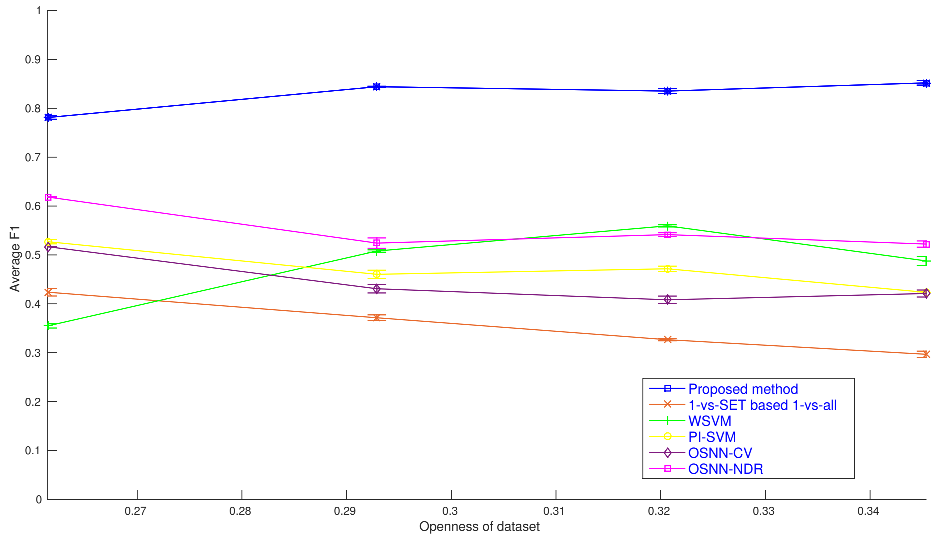


Figure 5.10: AKUF_1 results on *Segment* on *four* openness values.

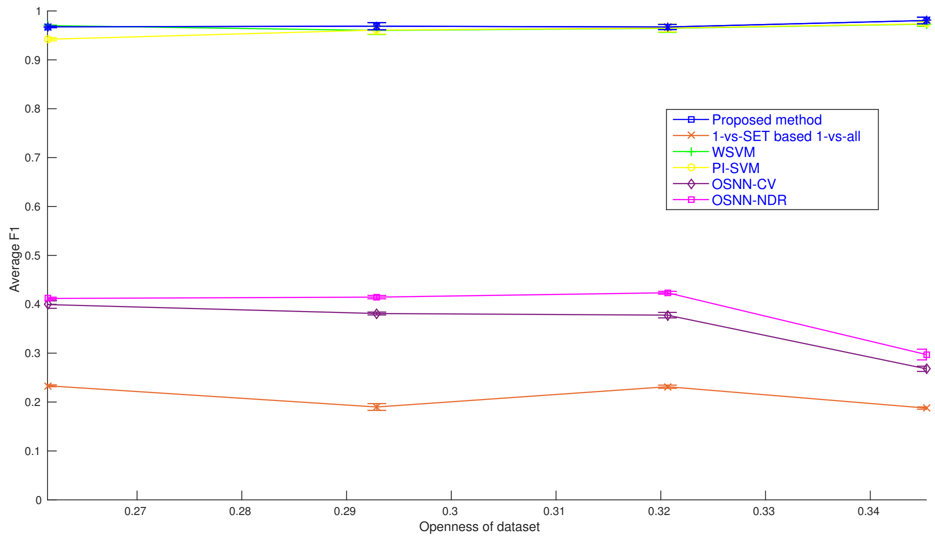


Figure 5.11: AKUF_1 results on *Shuttle* on *four* openness values.

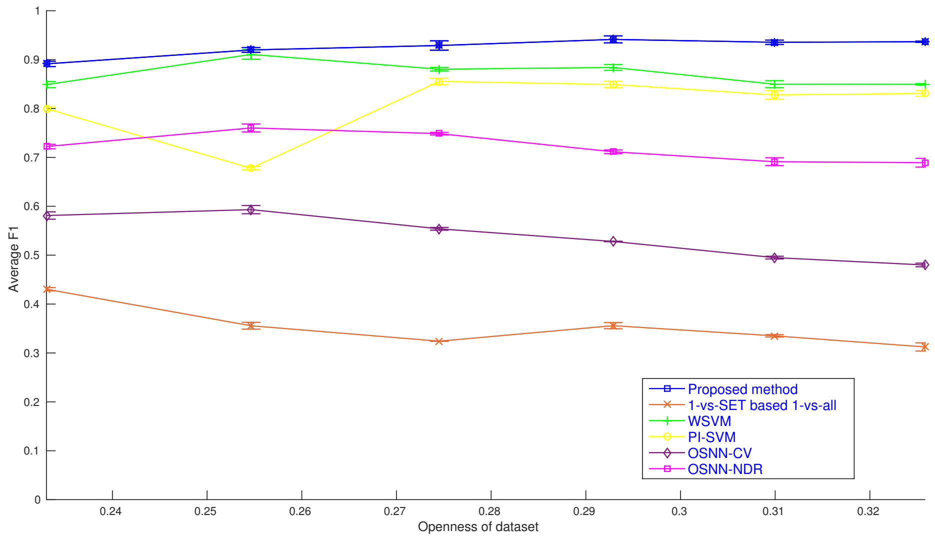


Figure 5.12: AKUF_1 results on *Texture* on *six* openness values.

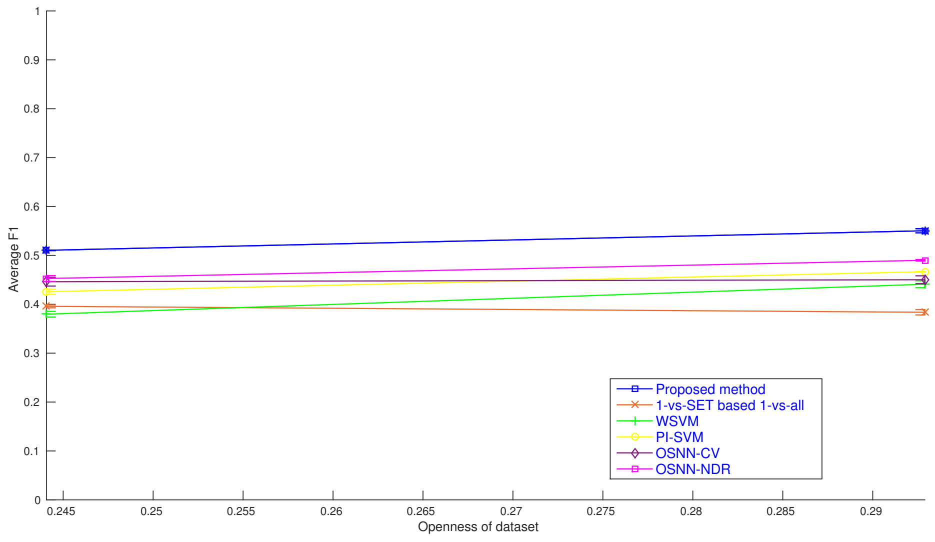


Figure 5.13: AKUF_1 results on *Vehicle* on *two* openness values.

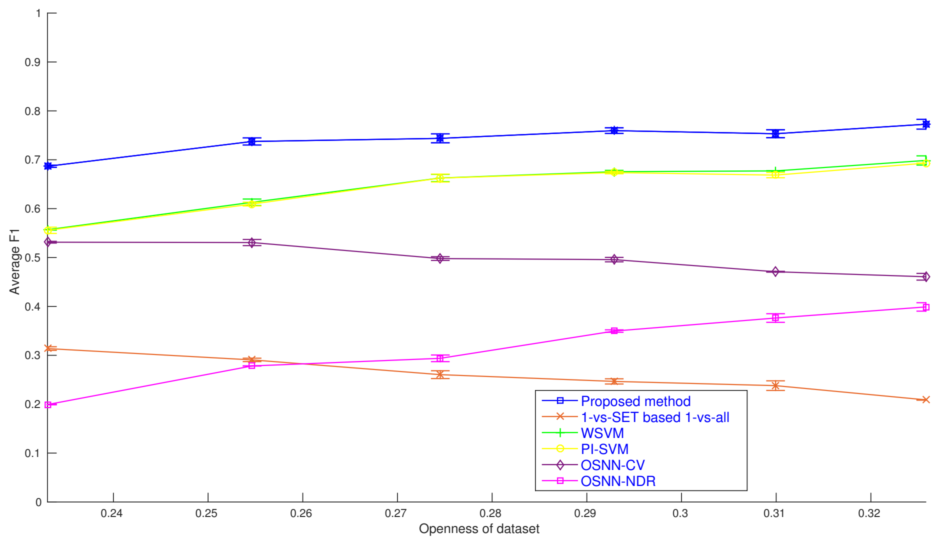


Figure 5.14: AKUF_1 results on *Vowel* on *six* openness values.

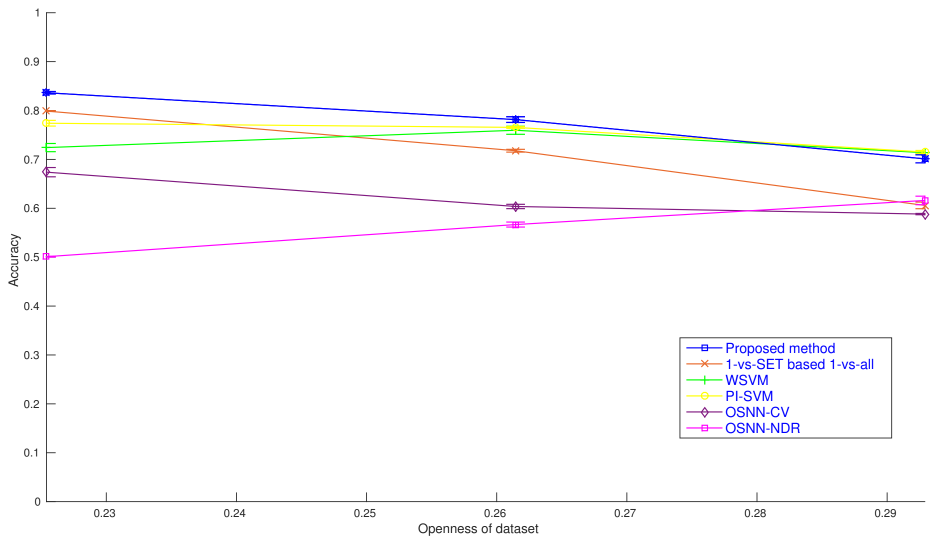


Figure 5.15: Accuracy results on *Dermatology* on *three* openness values.

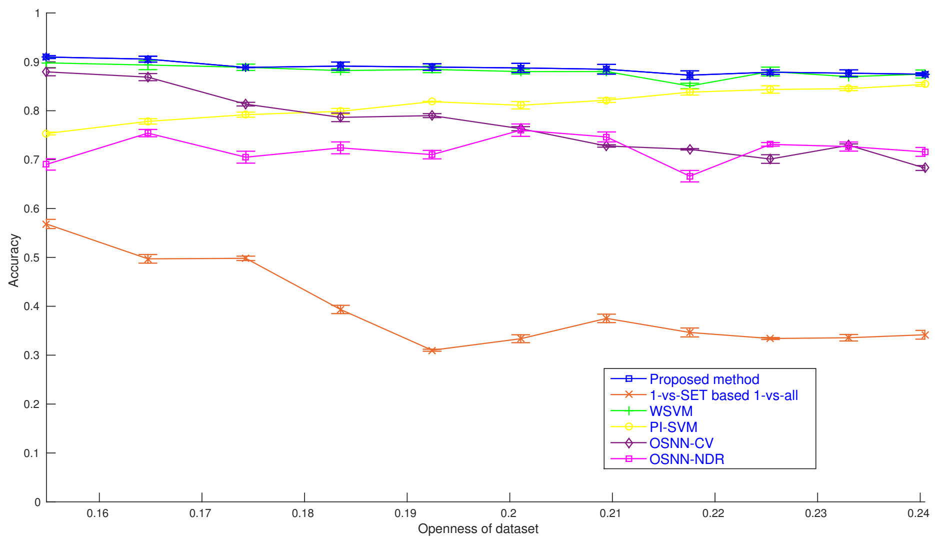


Figure 5.16: Accuracy results on *Letter* on *eleven* openness values.

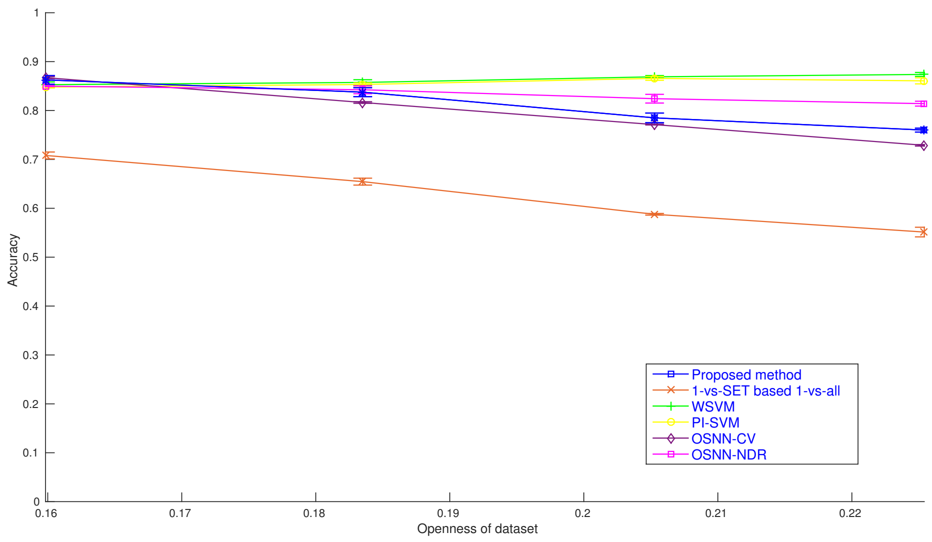


Figure 5.17: Accuracy results on *MNIST* on *four* openness values.

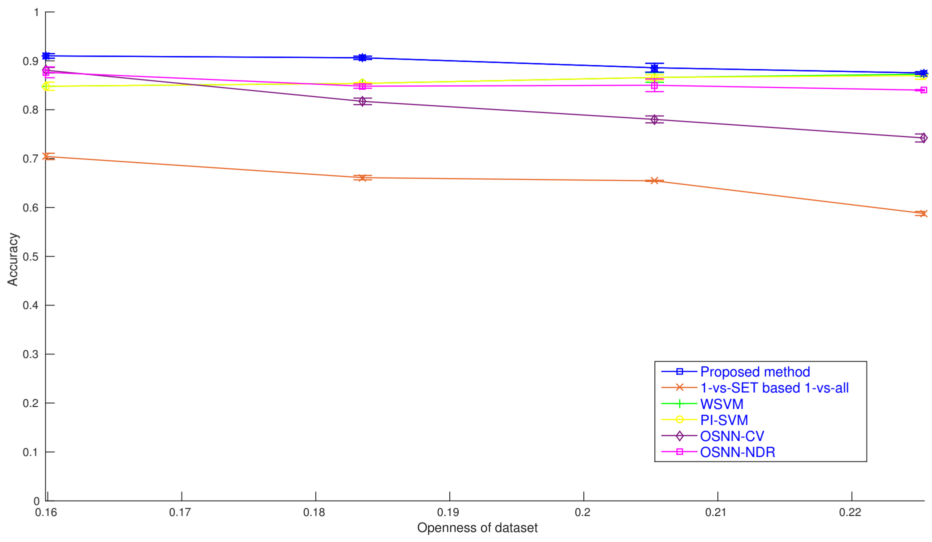


Figure 5.18: Accuracy results on *Reduced-MNIST* on *four* openness values.

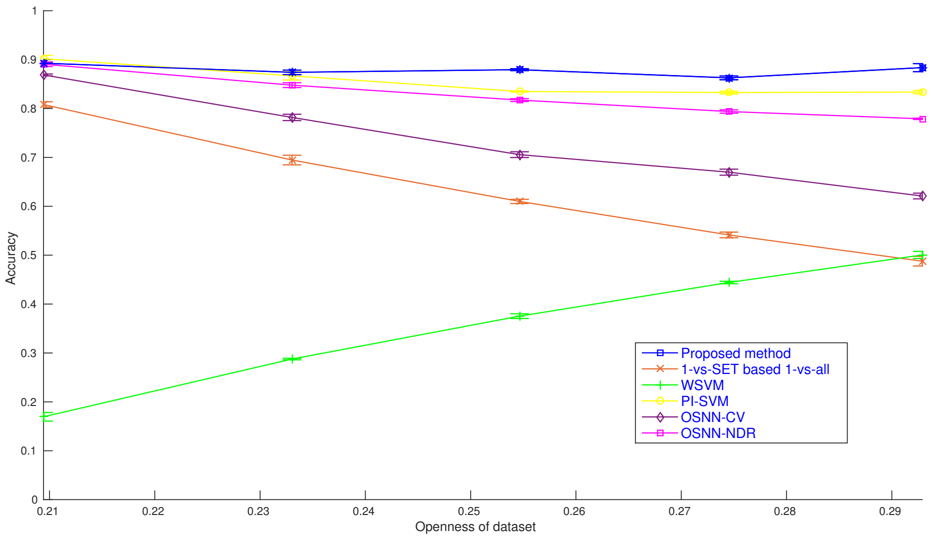


Figure 5.19: Accuracy results on *Optdigits* on *five* openness values.

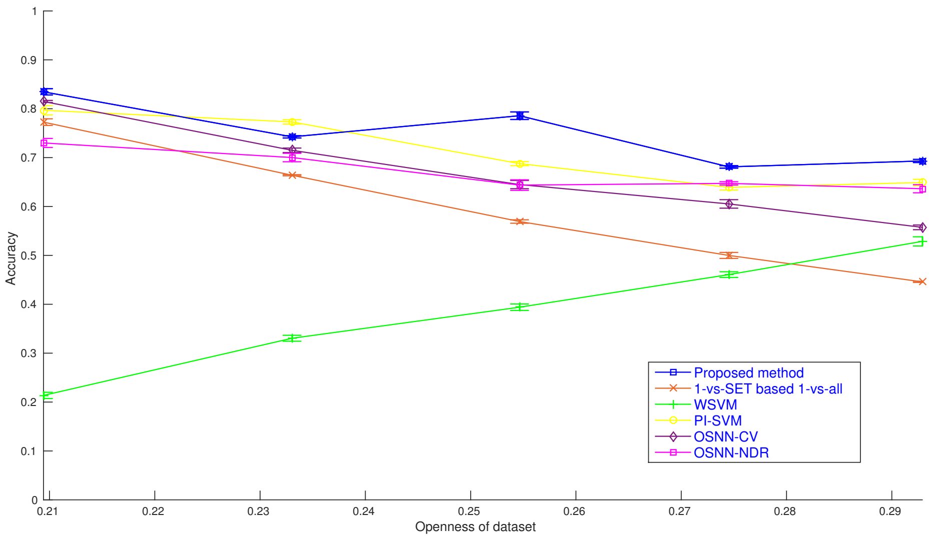


Figure 5.20: Accuracy results on *Penbased* on *five* openness values.

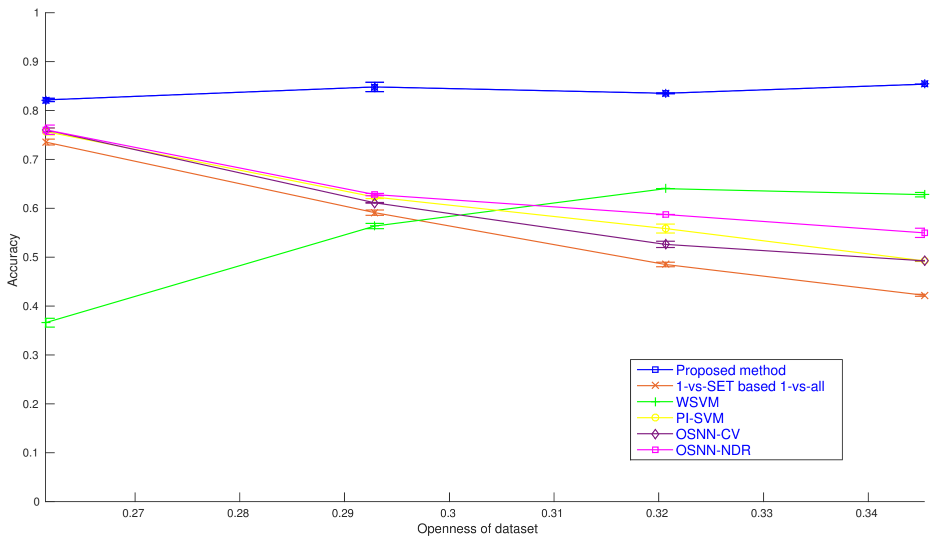


Figure 5.21: Accuracy results on *Segment* on *four* openness values.

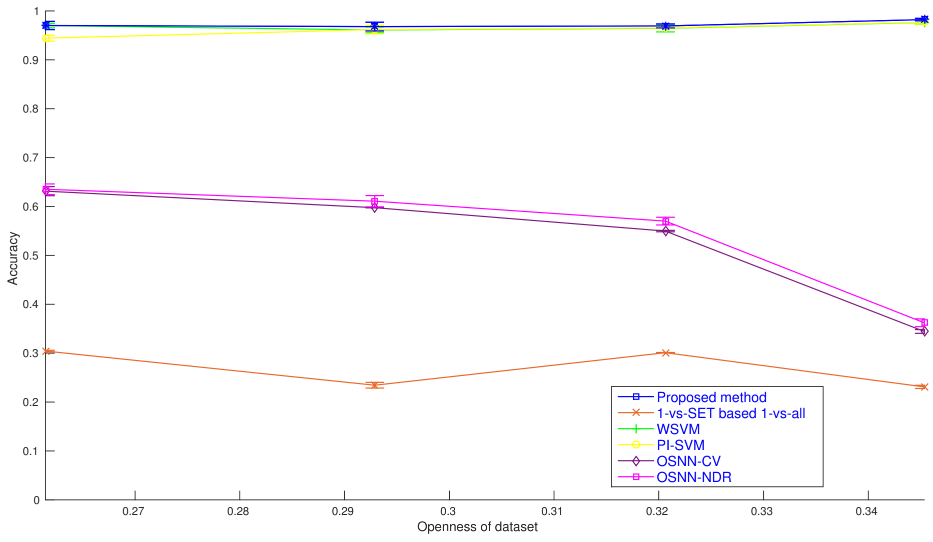


Figure 5.22: Accuracy results on *Shuttle* on *four* openness values.

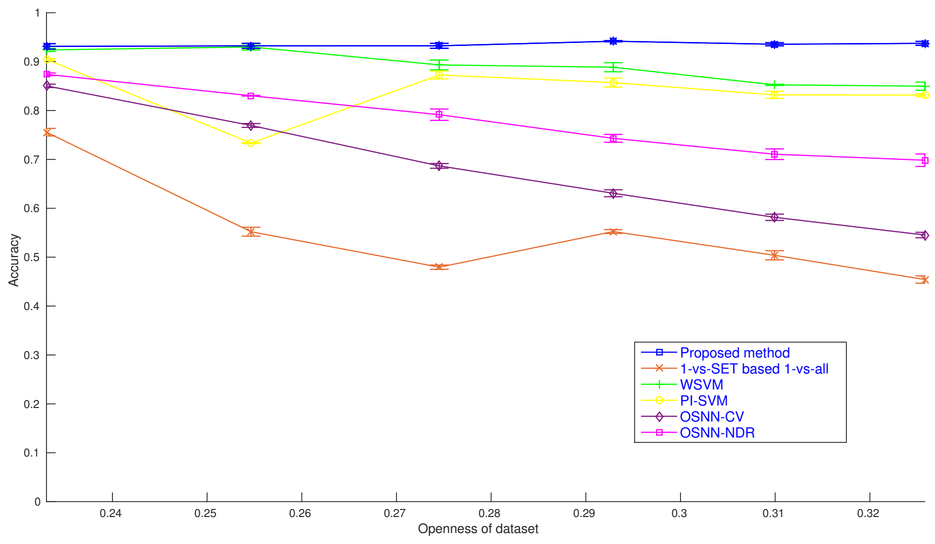


Figure 5.23: Accuracy results on *Texture* on *six* openness values.

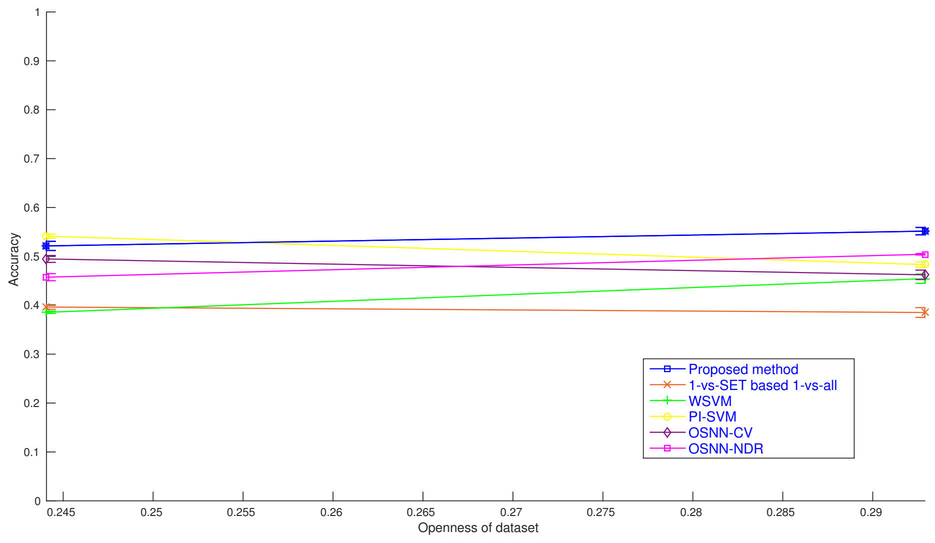


Figure 5.24: Accuracy results on *Vehicle* on *two* openness values.

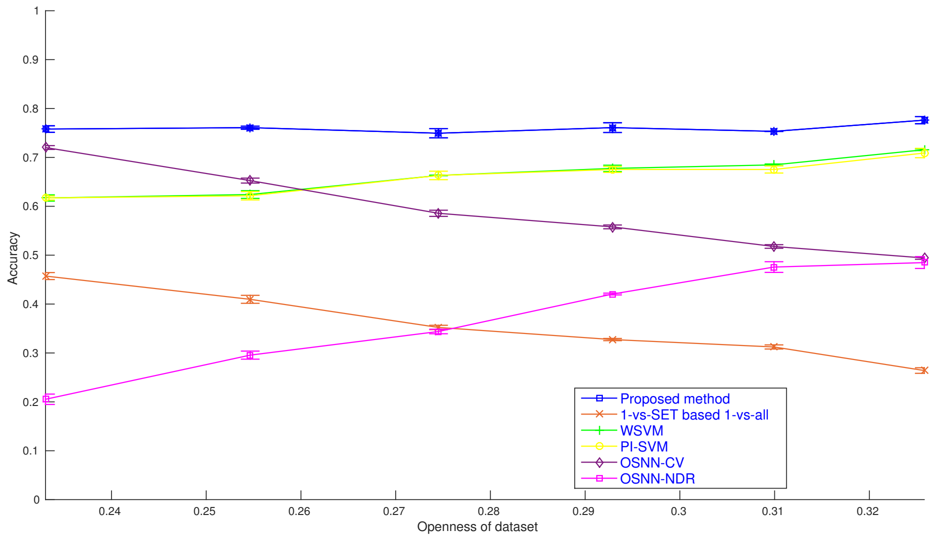


Figure 5.25: Accuracy results on *Vowel* on *six* openness values.

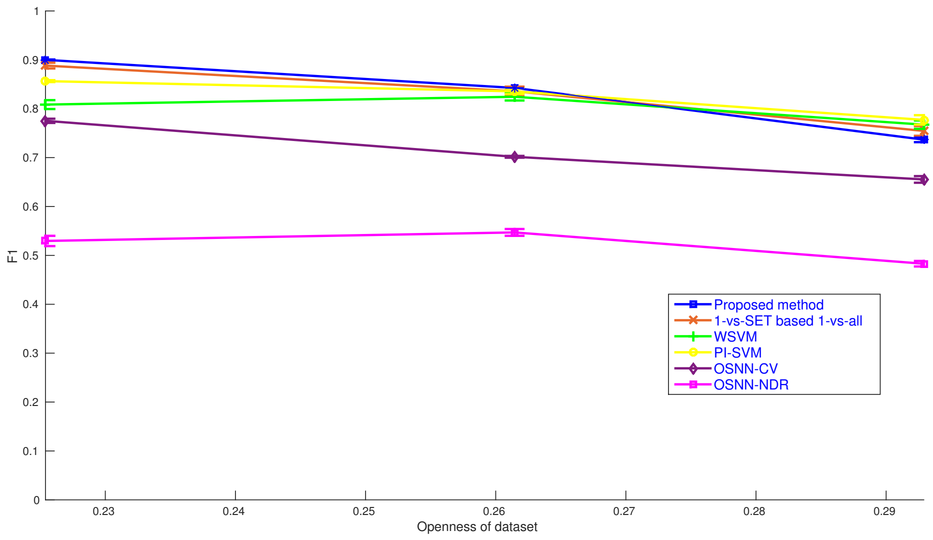


Figure 5.26: F_1 results on *Dermatology* on *three* openness values.

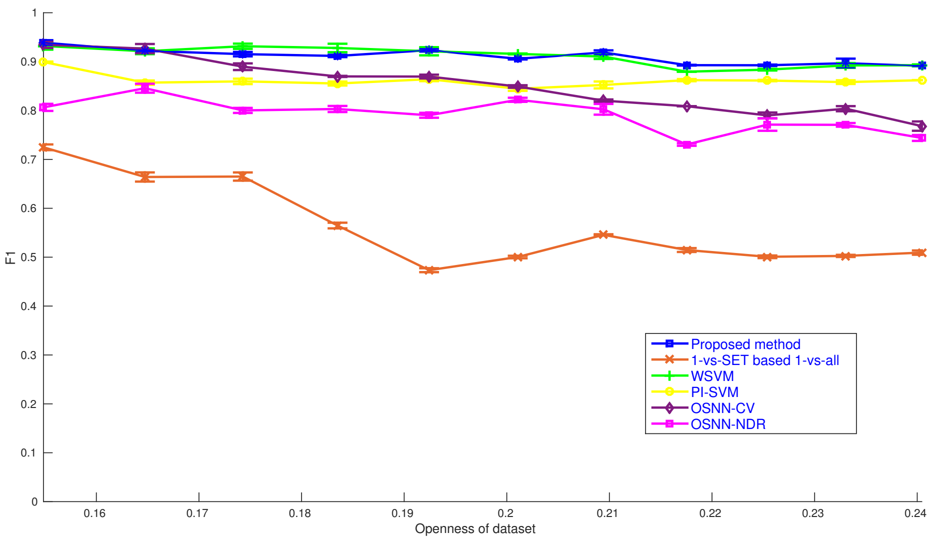


Figure 5.27: F_1 results on *Letter* on *eleven* openness values.

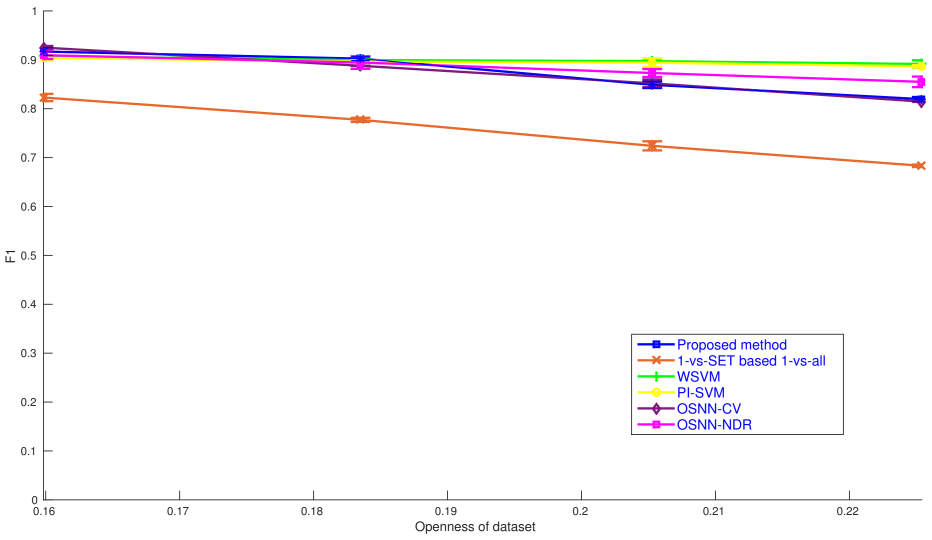


Figure 5.28: F_1 results on *MNIST* on *four* openness values.

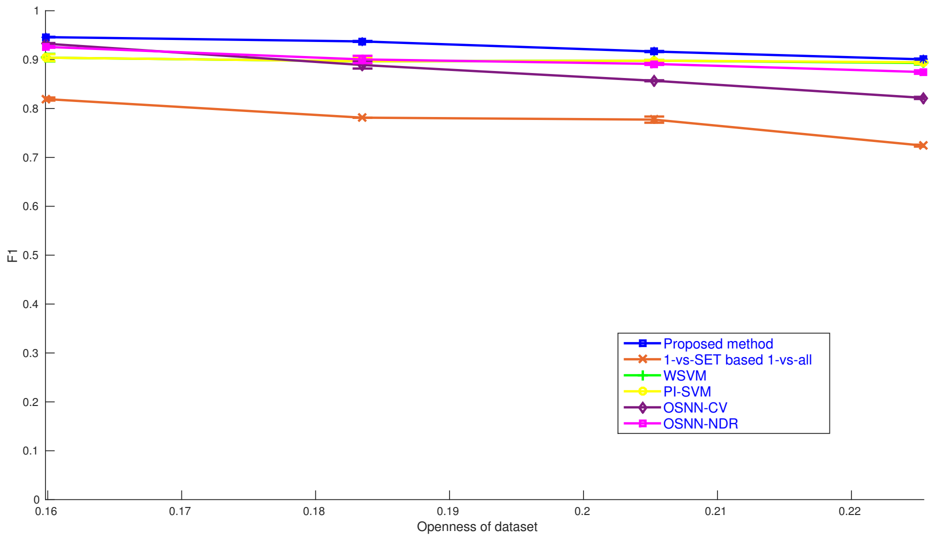


Figure 5.29: F_1 results on *Reduced-MNIST* on *four* openness values.

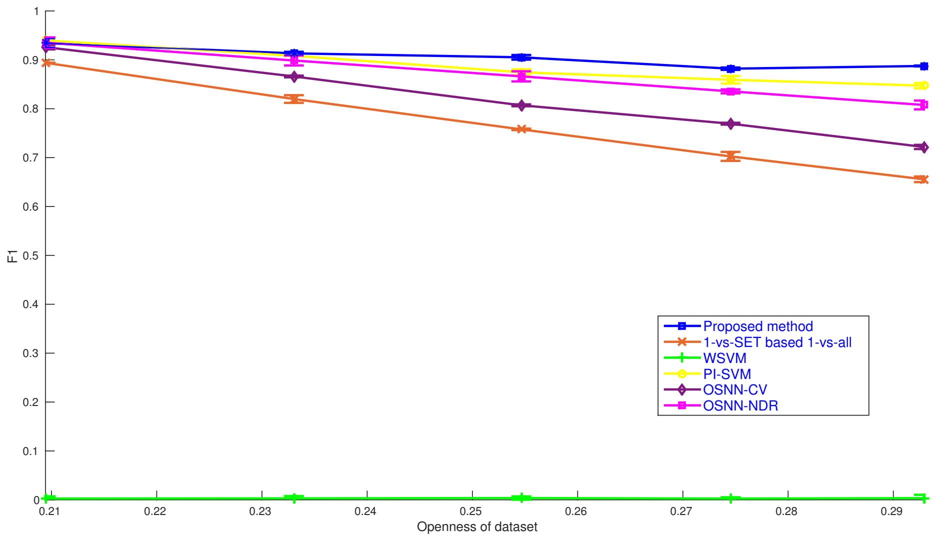


Figure 5.30: F_1 results on *Optdigits* on *five* openness values.

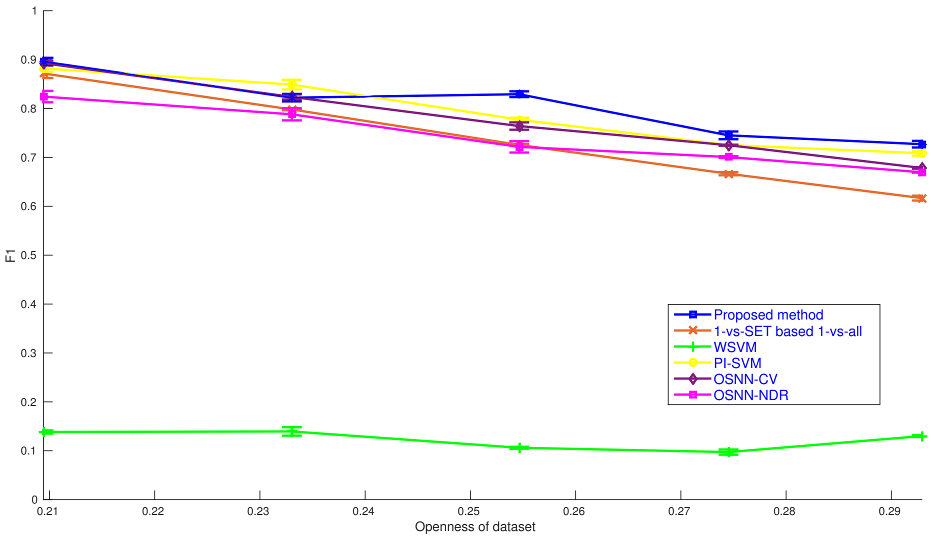


Figure 5.31: F_1 results on *Penbased* on *five* openness values.

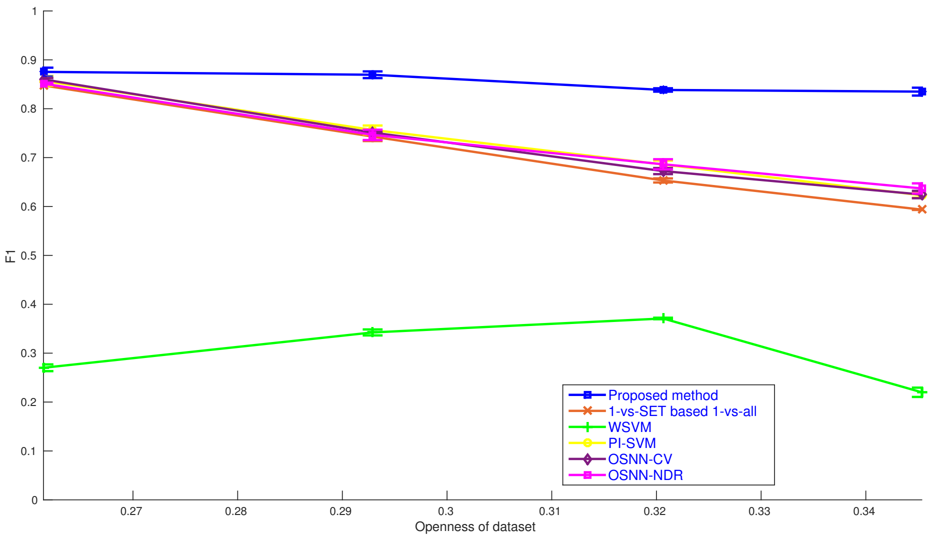


Figure 5.32: F_1 results on *Segment* on *four* openness values.

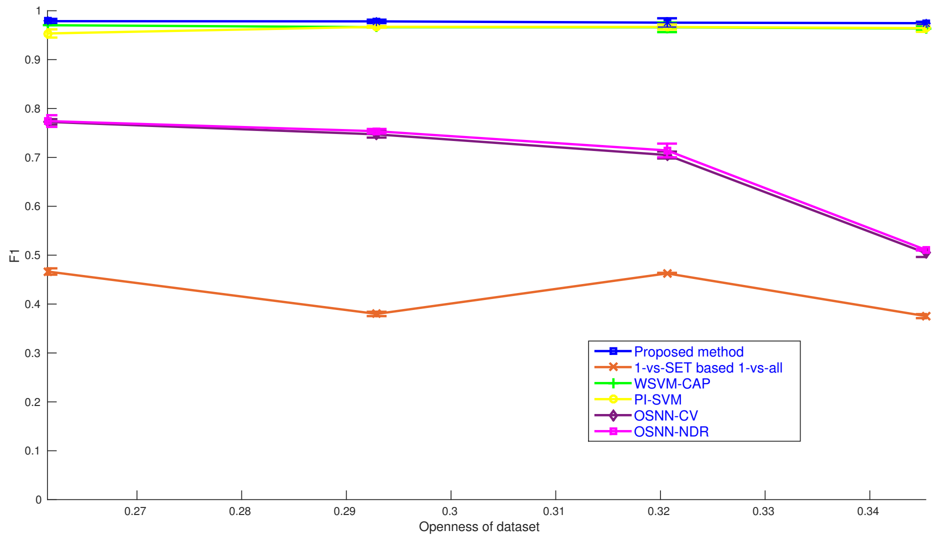


Figure 5.33: F_1 results on *Shuttle* on *four* openness values.

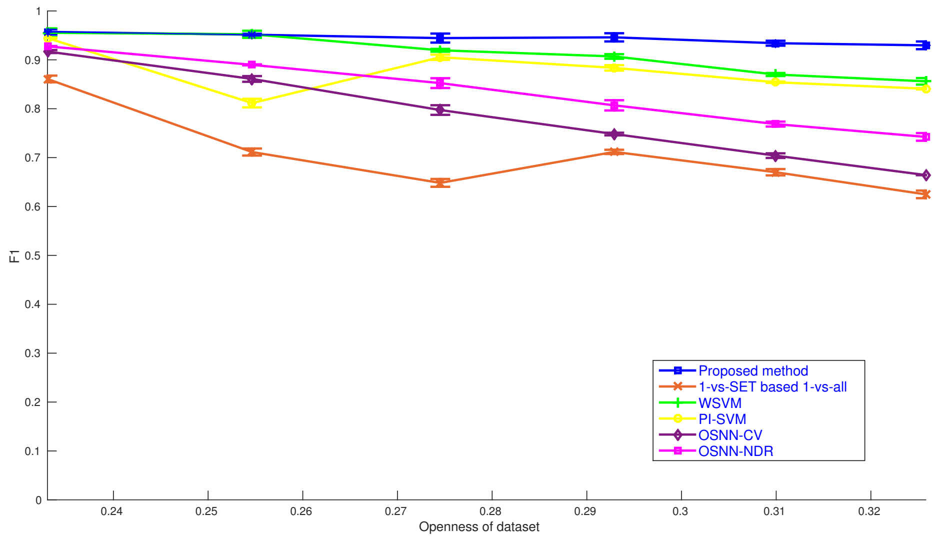


Figure 5.34: F_1 results on *Texture* on *six* openness values.

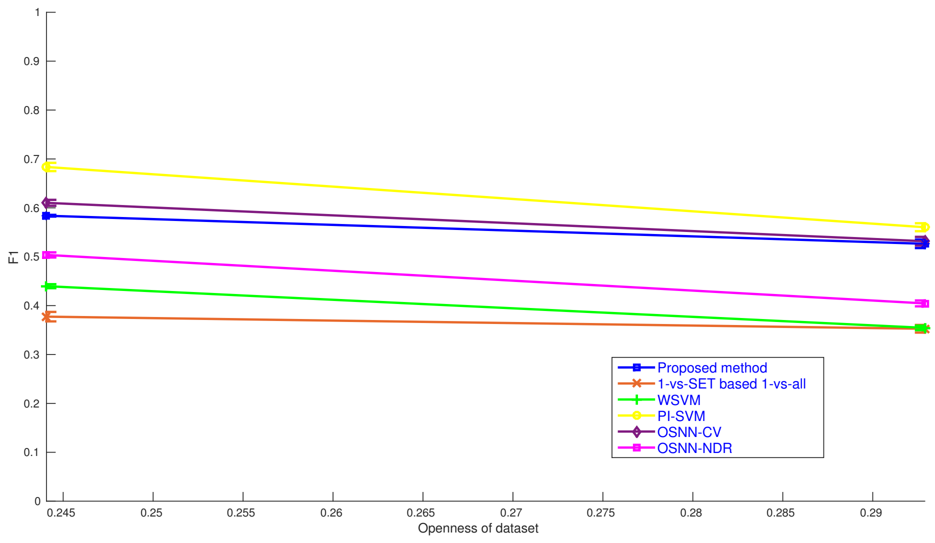


Figure 5.35: F_1 results on *Vehicle* on *two* openness values.

5.7.2 Limitations of the proposed scheme

The proposed work deals with RkNN in which distance and neighborhood relation is the only information that is interpreted. Like any other distance-based scheme, our method suffers from the curse of dimensionality at higher dimensions. The same phenomenon was observed for the original MNIST dataset with 784 features. To curb this problem, we suggest a reduction in feature dimension of a dataset with ≥ 100 features through feature extraction or selection before proceeding with the RkNN-based learning and classification. The improvement in performance on Reduced-MNIST dataset (Figures 6, 17, 28) over original MNIST dataset (Figures 5, 16, 27) indicates dimensionality can be an issue.

5.8 Experiment on parameter tuning

On *four* datasets, namely, *Dermatology, Vehicle, Segment and Vowel*, we have conducted a parameter tuning experiment. Neighborhood size k is the only tunable parameter of our scheme. From our detailed experimental study and analysis, we have seen that a k value in the range 2, 3, 4, 5, 6 works well for all the datasets that we have used. Accordingly, we have reported the accuracy results of the four mentioned datasets across these five k values. Figures 5.37-5.40 show the same. It is interesting to note that a

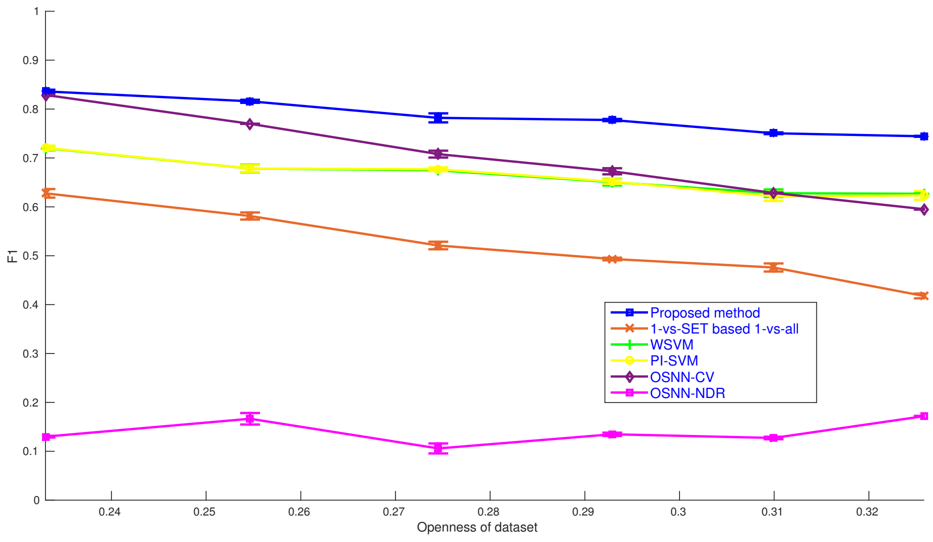


Figure 5.36: F_1 results on *Vowel* on *six* openness values.

single k value may not work well on a dataset. So, it is advisable to tune the k value across different opennesses of a single dataset. The detailed procedure for parameter optimization is given Section 5.6.2.

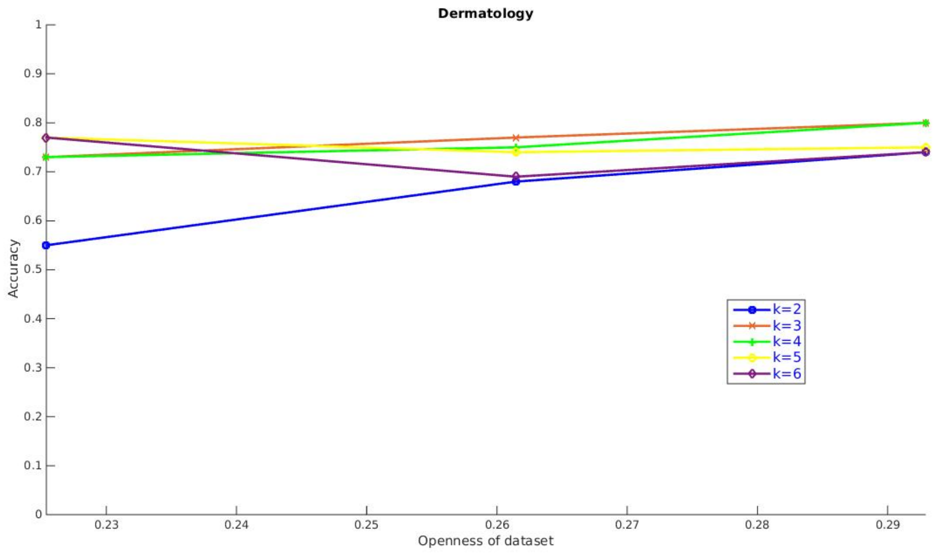


Figure 5.37: *Accuracy* results on *Dermatology* on *three* openness values and varying *k* values.

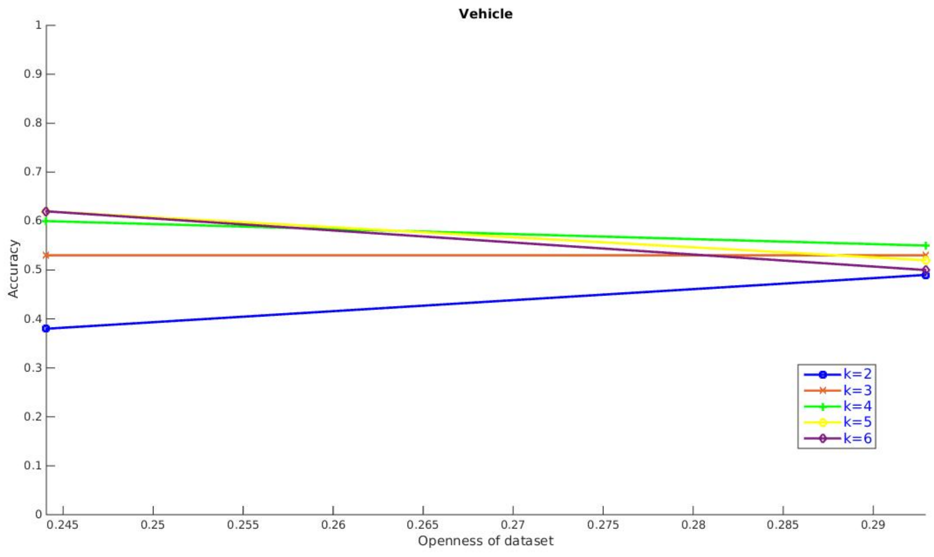


Figure 5.38: *Accuracy* results on *Vehicle* on *two* openness values and varying *k* values.

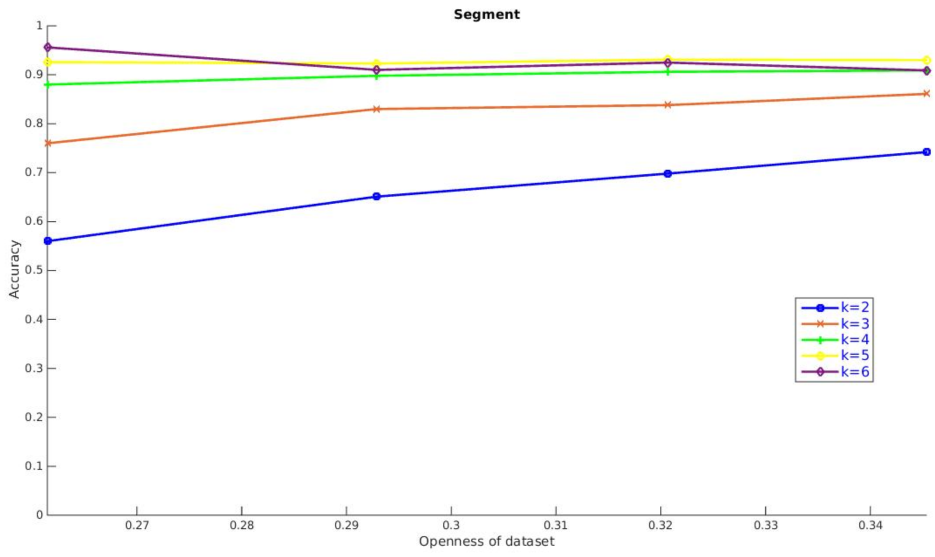


Figure 5.39: *Accuracy* results on *Segment* on *four* openness values and varying k values.

5.9 Summary

In this chapter, we have presented a novel reverse k-nearest neighbor based classifier. The elegance of this classifier lies with it's innate ability to address open set classification. RkNN based neighborhood identification does the task of unknown class detection besides the regular known class classification naturally. Choice of k or neighborhood size is dataset dependent and it is determined through cross-validation on the training set. Apart from that, no thresholding or parameters are involved to distinguish the known and unknown subspaces. A unique attribute of the proposed scheme is that it estimates and explores the sampling window implicitly. The RkNN process itself adaptively adjusts the class boundaries, depending on the local sparseness of the training data and this contributes to the simplicity and efficiency of the scheme. The proposed classifier also operates on an intrinsic multi-class framework. A comprehensive empirical study affirms the capability of the proposed scheme to deliver competent to superior performance against competing classifiers in an open set backdrop.

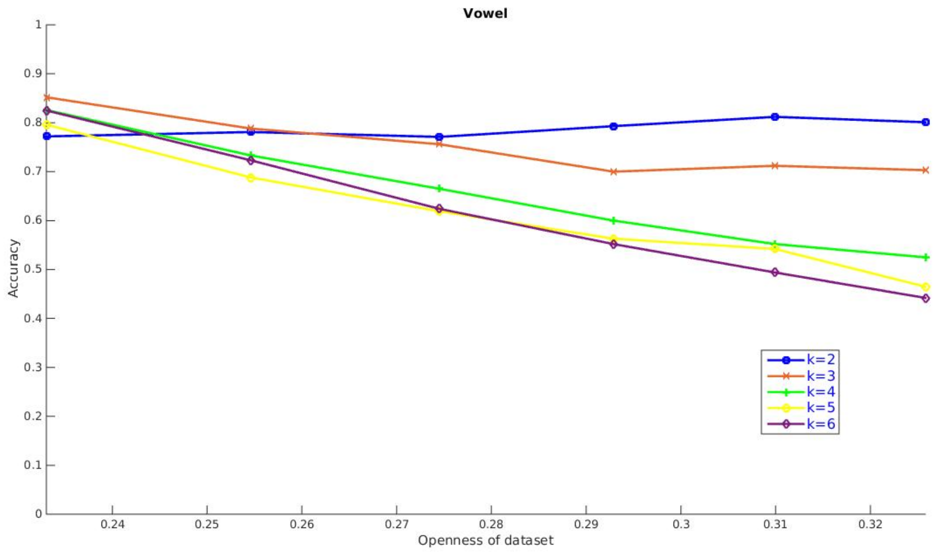


Figure 5.40: *Accuracy* results on *Vowel* on *six* openness values and varying k values.

Chapter 6

Conclusion and Scope of Further Research

The principal objective of this thesis is to find ways of meaningful learning of the real world data where the data or classification paradigm possess some form of deviations from the traditional scenario. In this thesis, we have addressed three particular classes of such deviations — i]. generic class imbalance of datasets, ii]. multi-label datasets and iii]. open set classification. Our primary concern has been to identify one or more key aspects of each of these classes of data or classification paradigm. After identifying the aspect/s, we work on towards them to generate an efficient classifier for solving the problem. We have reported a chapter-wise list of findings below.

- **Chapter 2**: In this chapter, we have addressed class imbalance, a classical form of data irregularity. Data preprocessing through minority class oversampling is a standard and fruitful technique for handling such data. But, there is a critical gap between oversampling to improve the classification results and non-encroaching oversampling. We have shown that the synthetic minority points estimated by the state-of-the-art methods do not preserve the boundaries of the Voronoi cells (constructed on the basis of original training points). They encroach upon the majority class cells.

    1. To solve this issue, we have proposed a novel direction of minority set estimation before generating the synthetic minority points. We estimate the support of the minority classes. We have proposed two solutions using i]. Minimum Spanning Tree (MST) and ii]. Relative Neighborhood Graph (RNG). In either

case, the synthetic minority points lie within the minority cells (of the Voronoi diagram) only.

2. Between the two solutions rendered by us, the second one (using RNG) has a layer of refinement over the first (using MST). In the first solution, the synthetic minority points lie within a small hypersphere around the original minority point and do not distribute across the entire minority cell. In the second solution, we solve this issue by implementing an adaptive protocol.
3. The empirical study indicates the efficaciousness of the proposed schemes. The second solution fares better than the first scheme and surpasses the performance of all the state-of-the-art minority oversamplers.

- **Chapter 3**: In this chapter, we focus our attention on multi-label datasets, whose abundance can be attributed to a number of real-world domains like medical, text and tag recommendation systems. Such datasets have a single feature (instance) set but their class memberships distribute across multiple labels. Consequently, the class partitions vary from one label to another. The single, original feature set may not be optimal for learning all the labels.

  1. In a multi-label dataset, the positive and negative class geometries of the labels are distinct. Initially, we learn the positive and negative class lattices of each label. Two graph based techniques i]. MST and ii]. RNG are used for determining the lattices.
  2. In spite of having the same instance set, the set and number of lattices vary from label to label. We derive a label specific feature set from the lattice sets of each label.
  3. The feature extraction technique that we have proposed is a supervised one and takes into account the class information, which is the distinguishing factor. This gives us label-specific features for the labels.
  4. The findings of our experiments indicate that the proposed schema of label-dedicated features have a generic multi-label learning capability and have fared better than most of the multi-label classifiers across different metrics.

- **Chapter 4**: This chapter is our second attempt on multi-label datasets and it focuses on the class-imbalance aspect of the multi-label datasets. In general, a number of labels of a multi-label dataset have very few positive instances leading to a high imbalance ratio. While performing the experiments of the previous chapter, we have noticed that scores on metrics like Hamming Loss may not provide us a clear picture. For proper learning, we have to work on and analyze the imbalance aspect of multi-label data.

1. First, we should note that the degree of class-imbalance of a multi-label dataset varies across labels. There is no single imbalance ratio of a multi-label dataset, instead we have one for each label.
2. The first solution that we have proposed solves the imbalance issue by oversampling the minority classes of each label. We generate the synthetic minority points from the density-adaptive reverse nearest neighborhood of the minority points. The original instances and the synthetic minority points are together used to learn the classifiers for each label.
3. In the second solution, we add an augmented misclassification cost for the minority class of each label. We set the misclassification cost according to the imbalance ratio of a label.
4. Both the solutions solve the problem of class-imbalance of multi-label datasets in an elegant as well as effective fashion. They surpass the performances of all the state-of-the art methods in multi-label learning.

- **Chapter 5**: This chapter deals with open set classification, where a classifier needs to correctly classify the known class instances as well as identify the unknown ones. It is an important class of learning because open set classification mimics the real-world learning where we grow and learn incrementally in an unknown world.

  1. The solution that we propose is based on Reverse nearest neighborhood principles, which is quite novel in the field of classification. It allows us to perform unknown class detection and known class classification simultaneously.
  2. The proposed solution is elegant and simple. Moreover, in our solution, we do not need to expose the classifier to the unknown class instances in the training phase at all. Concurring with the essence of open set classification, the classifier is modeled on the known training instances only.
  3. The empirical results have shown the potency of the proposed scheme to efficiently handle open set classification. However, we have also observed that the proposed technique suffers from the curse of dimensionality. Consequently, on datasets with over 100 features, we suggest a reduction in feature set cardinality via PCA or any standard feature extraction technique.

6.1 Future Work

In this thesis, we have addressed three specific types of deviations (in data or in classification protocol) and provided solutions to them. We have some specific plans for our future work. First of all, we will re-study these three particular types of deviations to find and unravel some more useful characteristics and work on them. Specifically, we would like to focus on known unknown and unknown unknown aspect of open set classification. In context of multi-label data, we would like to focus and address label correlation in multi-label datasets. From a perspective of real-world data, multi-label datasets with both nominal and numeric features is a relevant one and it would constitute our interest. The other end of our future work will be directed at learning some more types of irregularities in a classification framework which arises from a real-world scenario. Specifically, I would like to explore the domain of concept drift, where the data characteristics (and consequently the classifier model) varies over time.

Bibliography

[1] J. Alcalá-fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. a member of the old city publishing group. keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, 2011.

[2] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Towards open-set identity preserving face synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[3] S. Barua, M. M. Islam, X. Yao, and K. Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, Feb 2014.

[4] Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

[5] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[6] Lorenzo Bruzzone and Sebastiano B Serpico. Classification of imbalanced remote-sensing data by neural networks. *Pattern recognition letters*, 18(11):1323–1328, 1997.

[7] A. Bryant and K. Cios. Rnn-dbscan: A density-based clustering algorithm using reverse nearest neighbor density estimates. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1109–1121, 2018.

[8] J. Burez and D. Van den Poel. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3, Part 1):4626 – 4636, 2009.

[9] D. O. Cardoso, F. França, and J. Gama. A bounded neural network for open set recognition. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, July 2015.

[10] Douglas O. Cardoso, João Gama, and Felipe M. França. Weightless neural networks for open set recognition. *Mach. Learn.*, 106(9-10):1547–1567, October 2017.

[11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[12] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[13] Sheng Chen, Haibo He, and Edwardo A. Garcia. Ramoboost: Ranked minority oversampling in boosting. *Trans. Neur. Netw.*, 21(10):1624–1642, October 2010.

[14] Weiwei Cheng, Eyke Hüllermeier, and Krzysztof J Dembczynski. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 279–286, 2010.

[15] D. A. Cieslak, N. V. Chawla, and A. Striegel. Combating imbalance in network intrusion datasets. In *2006 IEEE International Conference on Granular Computing*, pages 732–737, May 2006.

[16] M. A. Córdova Neira, P. Ribeiro Mendes Júnior, A. Rocha, and R. Da Silva Torres. Data-fusion techniques for open-set recognition problems. *IEEE Access*, 6:21242–21265, 2018.

[17] Zachary Alan Daniels and Dimitris N Metaxas. Addressing imbalance in multi-label classification using structured hellinger forests. In *AAAI*, pages 1826–1832, 2017.

[18] Debashree Devi, Saroj kr. Biswas, and Biswajit Purkayastha. Redundancy-driven modified tomek-link based undersampling: A solution to class imbalance. *Pattern Recognition Letters*, 93(Supplement C):3 – 12, 2017. Pattern Recognition Techniques in Data Mining.

[19] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. Fast and efficient visual codebook construction for multi-label annotation using predictive clustering trees. *Pattern Recognition Letters*, 38:38 – 45, 2014.

[20] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 681–687, Cambridge, MA, USA, 2001. MIT Press.

[21] Seyda Ertekin, Jian Huang, and C. Lee Giles. Active learning for class imbalance problem. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 823–824, New York, NY, USA, 2007. ACM.

[22] Tom Fawcett and Foster Provost. Adaptive fraud detection. *Data Min. Knowl. Discov.*, 1(3):291–316, January 1997.

[23] Alberto Fernández, Sara del Río, Nitesh V Chawla, and Francisco Herrera. An insight into imbalanced big data classification: outcomes and challenges. *Complex & Intelligent Systems*, 3(2):105–120, 2017.

[24] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73(2):133–153, November 2008.

[25] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. 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)*, 42(4):463–484, July 2012.

[26] V. García, J. S. Sánchez, R. Martín-Félez, and R. A. Mollineda. Surrounding neighborhood-based smote for learning from imbalanced data sets. *Progress in Artificial Intelligence*, 1(4):347–362, Dec 2012.

[27] Eva Gibaja and Sebastián Ventura. Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(6):411–444, 2014.

[28] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. *In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30, 2004.

[29] Bastian Goldluecke and Daniel Cremers. Convex relaxation for multilabel problems with product label spaces. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[30] B. Gorte and N. Gorte-Kroupnova. Non-parametric classification algorithm with an unknown class. In *Proceedings of International Symposium on Computer Vision - ISCV*, pages 443–448, Nov 1995.

[31] U Grenander. *Abstract Inference.* Wiley, USA, 1981.

[32] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Proceedings of the 2005 International Conference on Advances in Intelligent Computing - Volume Part I*, ICIC'05, pages 878–887, Berlin, Heidelberg, 2005. Springer-Verlag.

[33] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, June 2008.

[34] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009.

[35] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[36] C. Huang, Y. Li, C. C. Loy, and X. Tang. Learning deep representation for imbalanced classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5375–5384, June 2016.

[37] J. Huang, G. Li, Q. Huang, and X. Wu. Joint feature selection and classification for multilabel learning. *IEEE Transactions on Cybernetics*, 48(3):876–889, March 2018.

[38] Kuan-Hao Huang and Hsuan-Tien Lin. Cost-sensitive label embedding for multi-label classification. *Mach. Learn.*, 106(9-10):1725–1746, October 2017.

[39] S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya, H. Al-Dossari, and S. Ahmad. An ensemble oversampling model for class imbalance problem in software defect prediction. *IEEE Access*, 6:24184–24195, 2018.

[40] Lalit P. Jain, Walter J. Scheirer, and Terrance E. Boult. Multi-class open set recognition using probability of inclusion. In *ECCV*, 2014.

[41] C. Ji, H. Hu, Y. Xu, Y. Li, and W. Qu. Efficient multi-dimensional spatial rknn query processing with mapreduce. In *2013 8th ChinaGrid Annual Conference*, pages 63–68, Aug 2013.

[42] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[43] Bikram Karmakar and Nikhil R. Pal. How to make a neural network say "don't know". *Information Sciences*, 430-431:444 – 466, 2018.

[44] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. *In: Proceedings of the ECML/PKDD-08 Workshop on Discovery Challenge*, 2008.

[45] X. Kong and P. S. Yu. Multi-label feature selection for graph classification. In *2010 IEEE International Conference on Data Mining*, pages 274–283, Dec 2010.

[46] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2):195–215, Feb 1998.

[47] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, AIME '01, pages 63–66, London, UK, UK, 2001. Springer-Verlag.

[48] Sauchi Stephen Lee. Noisy replication in skewed binary classification. *Comput. Stat. Data Anal.*, 34(2):165–191, August 2000.

[49] Feng Li, Duoqian Miao, and Witold Pedrycz. Granular multi-label feature selection based on mutual information. *Pattern Recogn.*, 67(C):410–423, July 2017.

[50] Tao Li and M. Ogihara. Toward intelligent music information retrieval. *Multimedia, IEEE Transactions on*, 8(3):564–574, June 2006.

[51] Ying Liu, Han Tong Loh, and Aixin Sun. Imbalanced text classification: A term weighting approach. *Expert Syst. Appl.*, 36(1):690–701, January 2009.

[52] D. O. Loftsgaarden and C. P. Quesenberry. A nonparametric estimate of a multivariate density function. *Ann. Math. Statist.*, 36(3), 06 1965.

[53] Jiaheng Lu, Ying Lu, and Gao Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD Conference*, 2011.

[54] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and SašO Deroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recogn.*, 45(9):3084–3104, September 2012.

[55] Musa A Mammadov, Alexander M Rubinov, and John Yearwood. The study of drug–reaction relationships using global optimization techniques. *Optimisation Methods and Software*, 22(1):99–126, 2007.

[56] Mostafa Mehdipour-Ghazi, Berrin A. Yanikoglu, and Erchan Aptoula. Open-set plant identification using an ensemble of deep convolutional neural networks. In *CLEF*, 2016.

[57] Luis J Mena and Jesus A Gonzalez. Machine learning for imbalanced datasets: Application in medical diagnostic. In *Flairs Conference*, pages 574–579, 2006.

[58] Pedro R. Mendes Júnior, Roberto M. de Souza, Rafael de O. Werneck, Bernardo V. Stein, Daniel V. Pazinato, Waldir R. de Almeida, Otávio A. B. Penatti, Ricardo da S. Torres, and Anderson Rocha. Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386, Mar 2017.

[59] D. Miller, L. Nicholson, F. Dayoub, and N. Sünderhauf. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3243–3249, May 2018.

[60] Stephen O Moepya, Sharat S Akhoury, and Fulufhelo V Nelwamondo. Applying cost-sensitive classification for financial fraud detection under high class-imbalance. In *2014 IEEE International Conference on Data Mining Workshop*, pages 183–192. IEEE, 2014.

[61] Jose M. Moyano, Eva L. Gibaja, Krzysztof J. Cios, and Sebastián Ventura. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. *Information Fusion*, 44:33 – 45, 2018.

[62] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2014.

[63] G. Nasierding, G. Tsoumakas, and A. Z. Kouzani. Clustering based multi-label classification for image annotation and retrieval. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 4514–4519, Oct 2009.

[64] Kang Ning, Hoong Kee Ng, Sriganesh Srihari, Hon Wai Leong, and Alexey I Nesvizhskii. Examination of the relationship between essential genes in ppi network and hub proteins in reverse nearest neighbor topology. *BMC bioinformatics*, 11:505, October 2010.

[65] Emanuel Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3):1065–1076, 09 1962.

[66] James Petterson and Tibério S. Caetano. Reverse multi-label learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1912–1920. Curran Associates, Inc., 2010.

[67] D. Pokrajac, A. Lazarevic, and L. J. Latecki. Incremental local outlier detection for data streams. In *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515, March 2007.

[68] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, pages 17–26, New York, NY, USA, 2007. ACM.

[69] M. Radovanović, A. Nanopoulos, and M. Ivanović. Reverse nearest neighbors in unsupervised distance-based outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1369–1382, May 2015.

[70] M Mostafizur Rahman and DN Davis. Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, 3(2):224, 2013.

[71] Ajita Rattani, Walter J Scheirer, and Arun Ross. Open set fingerprint spoof detection across novel fabrication materials. *IEEE Transactions on Information Forensics and Security*, 10(11):2447–2460, 2015.

[72] J. Read, L. Martino, and D. Luengo. Efficient monte carlo optimization for multi-label classifier chains. pages 3457–3461, 2013.

[73] Jesse Read. A pruned problem transformation method for multi-label classification. In *Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, volume 143150, page 41, 2008.

[74] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2011.

[75] Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoff Holmes. MEKA: A multi-label/multi-target extension to Weka. *Journal of Machine Learning Research*, 17(21):1–5, 2016.

[76] Oscar Reyes, Carlos Morell, and Sebastián Ventura. Effective lazy learning algorithm based on a data gravitation model for multi-label learning. *Information Sciences*, 340-341:159 – 174, 2016.

[77] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[78] E. M. Rudd, L. P. Jain, W. J. Scheirer, and T. E. Boult. The extreme value machine. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):762–768, March 2018.

[79] P. Sadhukhan. Learning minority class prior to minority oversampling. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[80] W. J. Scheirer, L. P. Jain, and T. E. Boult. Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324, Nov 2014.

[81] Walter Scheirer, Anderson Rocha, Archana Sapkota, and Terrance Boult. Toward open set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1757–1772, July 2013.

[82] Alex Shenfield and Shahin Rostami. Multi-objective evolution of artificial neural networks in multi-class medical diagnosis problems with class imbalance. In *2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8. IEEE, 2017.

[83] Newton Spolaôr, Everton Alvares Cherman, Maria Carolina Monard, and Huei Diana Lee. A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135 – 151, 2013. Proceedings of the XXXVIII Latin American Conference in Informatics (CLEI).

[84] Hongyu Su and Juho Rousu. Multilabel classification through random graph ensembles. *Machine Learning*, 99(2), May 2015.

[85] Kenneth J. Supowit. The relative neighborhood graph, with an application to minimum spanning trees. *J. ACM*, 30(3):428–448, July 1983.

[86] Yufei Tao, Man Lung Yiu, and N. Mamoulis. Reverse nearest neighbor search in metric spaces. *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1239–1252, Sept 2006.

[87] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, Inc., USA, 4th edition, 2008.

[88] G. Tsoumakas, I. Katakis, and I. Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, July 2011.

[89] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

[90] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[91] Grigorios Tsoumakas and Min-Ling Zhang. Learning from multi-label data. 2009.

[92] Sen Wang, Xiaojun Chang, Xue Li, Guodong Long, Lina Yao, and Quan Z Sheng. Diagnosis code assignment using sparsity-based disease correlation embedding. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3191–3202, 2016.

[93] Wei Wei, Jinjiu Li, Longbing Cao, Yuming Ou, and Jiahang Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4):449–475, July 2013.

[94] Gang Wu and Edward Y. Chang. Class-boundary alignment for imbalanced dataset learning. In *In ICML 2003 Workshop on Learning from Imbalanced Data Sets*, pages 49–56, 2003.

[95] Gang Wu and Edward Y. Chang. Kba: Kernel boundary alignment considering imbalanced data distribution. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):786–795, June 2005.

[96] Yu-Ping Wu and Hsuan-Tien Lin. Progressive random k-labelsets for cost-sensitive multi-label classification. *Mach. Learn.*, 106(5):671–694, May 2017.

[97] Jin Xiao, Ling Xie, Changzheng He, and Xiaoyi Jiang. Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications*, 39(3):3668 – 3675, 2012.

[98] Suping Xu, Xibei Yang, Hualong Yu, Dong-Jun Yu, Jingyu Yang, and Eric C.C. Tsang. Multi-label learning with label-specific feature reduction. *Know.-Based Syst.*, 104(C):52–61, July 2016.

[99] Chan-Yun Yang, Jr-Syu Yang, and Jian-Jun Wang. Margin calibration in svm class-imbalanced learning. *Neurocomput.*, 73(1-3):397–411, December 2009.

[100] Show-Jane Yen and Yue-Shi Lee. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3, Part 1):5718 – 5727, 2009.

[101] Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[102] Jia Zhang, Candong Li, Donglin Cao, Yaojin Lin, Songzhi Su, Liang Dai, and Shaozi Li. Multi-label learning with label-specific features by resolving label correlations. *Knowledge-Based Systems*, 159:148 – 157, 2018.

[103] Min-Ling Zhang, Yu-Kun Li, and Xu-Ying Liu. Towards class-imbalance aware multi-label learning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4041–4047. AAAI Press, 2015.

[104] Min-Ling Zhang, José M. Peña, and Victor Robles. Feature selection for multi-label naive bayes classification. *Inf. Sci.*, 179(19):3218–3229, September 2009.

[105] Min-Ling Zhang and Lei Wu. Lift: Multi-label learning with label-specific features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(1):107–120, Jan 2015.

[106] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, July 2007.

[107] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[108] M.L. Zhang and Z.H. Zhou. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18:1338–1351, 2006.

[109] Z. Zhang, W. Jiang, J. Qin, L. Zhang, F. Li, M. Zhang, and S. Yan. Jointly learning structured analysis discriminative dictionary and analysis multiclass classifier. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3798–3814, Aug 2018.

[110] Z. Zhang, F. Li, L. Jia, J. Qin, L. Zhang, and S. Yan. Robust adaptive embedded label propagation with weight learning for inductive classification. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), Aug 2018.

[111] Z. Zhang, F. Li, M. Zhao, L. Zhang, and S. Yan. Joint low-rank and sparse principal feature coding for enhanced robust representation and visual classification. *IEEE Transactions on Image Processing*, 25(6):2429–2443, June 2016.

[112] Zhao Zhang, Weiming Jiang, Zheng Zhang, Sheng Li, Guangcan Liu, and Jie Qin. Scalable block-diagonal locality-constrained projective dictionary learning. In *Pro-*

ceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'19, pages 4376–4382. AAAI Press, 2019.

[113] Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explor. Newsl.*, 6(1):80–89, June 2004.