2013

On Identifying Critical Nuggets Of Information During Classification Task

David Sathiaraj
Louisiana State University and Agricultural and Mechanical College, davids@srcc.lsu.edu

ON IDENTIFYING CRITICAL NUGGETS OF INFORMATION DURING CLASSIFICATION TASK

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
David Sathiaraj
B.E. in Mechanical (Production) Engineering, Osmania University, 1998
M.S. in Industrial Engineering, Louisiana State University, 2000
M.S. in Systems Science, Louisiana State University, 2001
May 2013

To Matthew, for reminding me continually that there can be no machine learning algorithm that can match the God-endowed learning abilities of a young toddler

Acknowledgments

Given the circumstances in the past year, I would like to acknowledge the most important source of strength and hope. The only thing as a family we could cling to was a personal faith in God. This helped sustain every day when any amount of personal advice or comforting words could not resolve the hopelessness of the circumstances.

Next, I would like to thank several computer science and engineering faculty - Dr. J. Chen, Dr. E. Triantaphyllou, Dr. S. Rai and Dr. B. Karki for helpful research discussions. I would also like to thank my PhD committee co-chairs, Dr. Clint Willson and Dr. Bijaya Karki and the rest of the committee for clear research perspectives and for reviewing and evaluating my work. I would like to thank all the LSU administrators (deans of graduate school, college of science and college of engineering) and academic staff who have provided valuable input, advice and support during this effort.

I would like to thank Dr. Kevin Robbins, the Chair of the Department of Geography and the Director of the NOAA Southern Regional Climate Center. As my supervisor at my full-time job, he has supported every aspect of this PhD effort and it has been steady, unwavering support.

I would like to thank my colleagues at the NOAA Southern Regional Climate Center at LSU and the NOAA-funded Southern Climate Impacts Planning Program at LSU. I would like to single out friends and colleagues - Hal Needham and Yixin Luo. Our whiteboards were used heavily and there was no end to stimulating research discussions.

I would like to thank friends who have been dedicated sources of encouragement and support - Stan Gavin, Julius Esunge, Kevin McKee, the Tabbs, Wade Smith, Philip and Hannah Ebenezer, Manoj and Shanti Manda, the Spencers, Jennie Seals, Alene Casemore, the Avants, Cary and Jane deBessonet and many more.

I would like to thank my Dad and my Mom (retired academics - one a PhD in Computer Science and the other a high-school teacher) for their exemplary, principled lives, for teaching me to stand up for truth and to take the "narrow road that leads to true life". I also acknowledge my family - siblings, in-laws and cousins - for their encouragement over the years. Matthew, our two year old, has been a source of immense joy and cheer (even during the rough times). I thank my wife, Josephine for being a "wall" of strength and courage and for inspiring hope, grace and perseverance.

Table of Contents

List of Tables

List of Figures

Abstract

In large databases, there may exist critical nuggets - small collections of records or instances that contain domain-specific important information. This information can be used for future decision making such as labeling of critical, unlabeled data records and improving classification results by reducing false positive and false negative errors. In recent years, data mining efforts have focussed on pattern and outlier detection methods. However, not much effort has been dedicated to finding critical nuggets within a data set. This work introduces the idea of critical nuggets, proposes an innovative domain-independent method to measure criticality, suggests a heuristic to reduce the search space for finding critical nuggets, and isolates and validates critical nuggets from some real world data sets. It seems that only a few subsets may qualify to be critical nuggets, underlying the importance of finding them. The proposed methodology can detect them. This work also identifies certain properties of critical nuggets and provides experimental validation of the properties. Critical nuggets were then applied to 2 important classification task related performance metrics - classification accuracy and misclassification costs. Experimental results helped validate that critical nuggets can assist in improving classification accuracies in real world data sets when compared with other standalone classification algorithms. The improvements in accuracy using the critical nuggets were statistically significant. Extensive studies were also undertaken on real world data sets that utilized critical nuggets to help minimize misclassification costs. In this case as well the critical nuggets based approach yielded statistically significant, lower misclassification costs than than standalone classification methods.

Chapter 1
Introduction

1.1 Data Mining

In the past 20 years, the rapid growth of the world wide web and spectacular advances in technology and communications have generated an enormous amount of data. During the same time period, the storage capacity for devices has grown exponentially and the cost of storage devices has dramatically reduced as well. Technological advances such as smart phones and tablet computers have provided users with a host of 'apps' (or applications) that enable users to generate volumes of data, transmit the data using mobile communication channels to remote servers and also retrieve and analyze the data when they wish to. A simple example is a 'pedometer app' that can record each step that you take and store the data either on your phone or on a remote server (or to use computing jargon - store it in the 'cloud'). The next time you go for a walk, you can retrieve the data back, analyze all previous workout details and use it to compare your current workout with the previous workouts. This is just a simple example of how our daily lives are intertwined with data generation, data storage and data access. Data is ubiquitous.

Since there is so much of data being collected, stored and accessed, how could this data be put to good use and can one gain a better understanding of the data sets being collected? This is where the field of data mining comes in. Data mining involves the analysis of large data sets (or data bases) resulting in the discovery of previously unknown relationships and patterns that may exist within the data sets [24]. Hence, data mining is also commonly referred to as 'knowledge discovery in databases'. An implicit assumption in data mining is that the person analyzing the data set is not looking for an obvious or known relationship or pattern, but instead looking for previously unknown characteristics from the data set. A key aspect in data mining is to use the data collected in gaining a stronger understanding of the data domain. The gained knowledge provides the user the ability to make smarter and efficient decisions. If one wishes to mine a data set, data mining essentially involves the following steps:

- Preprocessing and cleaning of the data set.
- Apply a data mining algorithm.
- Analyze and test results.

A good overview of the field of data mining and knowledge discovery, including steps in a data mining process is provided in [15].

1.1.1 Data Mining Algorithms

Depending on the task at hand, different algorithms can be used on large data sets. These algorithms use several techniques drawn from statistics, computer science, machine learning, operations research and information theory to construct data mining models. These models are then used to make predictions, detect outliers, derive patterns and discover knowledge about the underlying data domain. An introductory reading on data mining methodologies can be found in work such as [15], [7] and [16]. Data mining algorithms can be broadly categorized as follows:

Classification Tasks

These are used in data mining when the main goal is to predict if a data record belongs to one class or another. In this case, data sets have among other attributes, an attribute called the class attribute. During classification tasks, one uses existing knowledge of class attribute values to predict unknown class attribute values. Detecting spam among a group of email records can be categorized as a classification task. In this example, the predicted value would be if an email is spam or not. Other examples include detecting credit card fraud and diagnosis of a disease. Typically, a classification task involves 2 steps - a training phase and a testing phase. During the training phase, a group of data records (with known values of the class attribute) are used by the data mining algorithm to learn specific characteristics about the data set. As a result, a data mining model is derived. In the next step, data records from the test data set are provided to the data mining model. The model predicts values for the class attribute. The number of correct predictions is a barometer on the accuracy of the classification algorithm.

Clustering

Clustering algorithms are used to group data records such that each group has data records that have very similar characteristics. As an example, consider a credit history data set. If the task at hand is to lower interest rates for people with good credit history, its best to use a clustering algorithm that clusters people with good credit history into one group and clusters people with moderate and poor credit histories into two other groups. In a large credit card transactional data set, say with more than a billion data records, clustering may help a company to narrow their business focus on one group of customers.

Association Rules

Association Rules are used to identify rules that represent frequent occurrences of certain relationships/patterns between the different data records.

Simply put, association rules answer questions such as *if ‘X’ and ‘Y’ occur, does ‘Z’ also occur*? These algorithms have applicability in supermarkets where a store manager can place items together after studying customer purchase patterns that indicated that people tend to buy certain items together. As an example of association rule, if the finding (or rule) was that: *If Jambalaya Mix and Andouille Sausage are bought together, then LSU Fan Gear are also bought*, then the store manager can stock Louisiana cuisine items and LSU fan gear in the same aisle. The methodology of association rule is different from a classification task. One important difference is that classification involves a non-deterministic goal which is predicting an previously unknown class value whereas association rule derivation is a more deterministic task (as extensively covered in [18]).

Outlier Detection

In some applications and data sets, the task is to pick out outliers - data records that are very different from the other data records. Outlier detection algorithms can also be used in the preprocessing stage to remove noisy data records. Outliers may also indicate an interesting phenomenon occurring within the data set. Outlier detection based data mining approaches are outlined in work such as [29] and [5].

A detailed survey of various data mining approaches can be found in [9].

1.2 Critical Nuggets

Given the above background, this research work and dissertation focusses on classification tasks [1]. During a classification task, there may be certain subsets of data within a data set that are more interesting or more important than the rest of the data set. These subsets are critical nuggets of information. In recent times, detecting patterns and outliers has emerged as an important area of work in the field of data mining. It has several applications including detecting fraud in business transactional data [30], identifying network intrusions [30], isolating abnormal trends in time-series data [47] and picking out suspicious criminal activity [51]. A lot of work in data mining has been devoted to finding interesting patterns or rules in data sets ([34], [19] and [45]). Work such as [50] consider the identification of interestingness measures, but the work was catering towards pattern mining and association rules. This is different from identifying critical nuggets during classification tasks. As mentioned earlier, the differences between association rules and classification tasks have been outlined in [18]. In [29], research was carried out on the mining of outliers and the concept of distance-based outliers. The work proposed to identify records that were anomalous or different from the rest of the data set. A good definition of an outlier is that of [25], *an outlier is an observation that deviates so much from other observations as to arouse suspicions that it was caused by a different mechanism.* Distance-based measures as in [4], [44] and [30] have been used in algorithms to delineate outliers or abnormal records from normal records. However, not much work has focussed on finding critical nuggets of information that may be hidden in data sets. These nuggets of information may

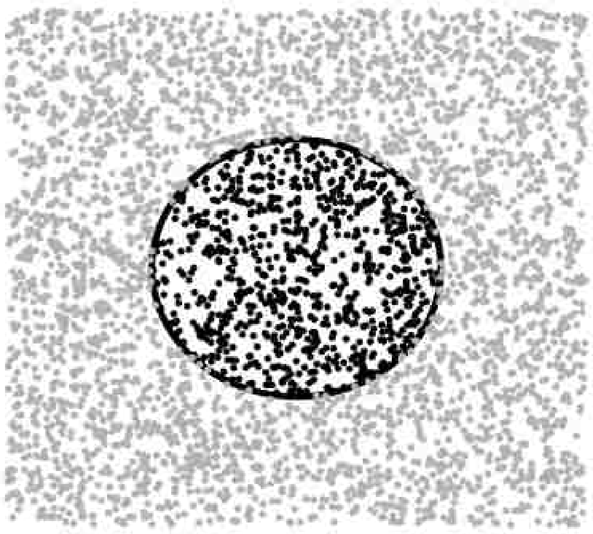
[1]Research work mentioned in chapters 1, 2, 3 and 4 has been accepted for publication in the IEEE Transactions of Knowledge and Data Engineering ©IEEE 2012. A detailed citation is *David Sathiaraj, Evangelos Triantaphyllou, "On Identifying Critical Nuggets Of Information During Classification Tasks," IEEE Transactions on Knowledge and Data Engineering, 23 May 2012. IEEE computer Society Digital Library. IEEE Computer Society, http://doi.ieeecomputersociety.org/10.1109/TKDE.2012.112.* It is also mentioned as [43]

not always be detected by pattern mining methods or by distance-based outlier detection methods as nuggets may not conform to a specific pattern and may not be outliers. Critical nuggets may not necessarily lie at a great distance from the rest of the data records.

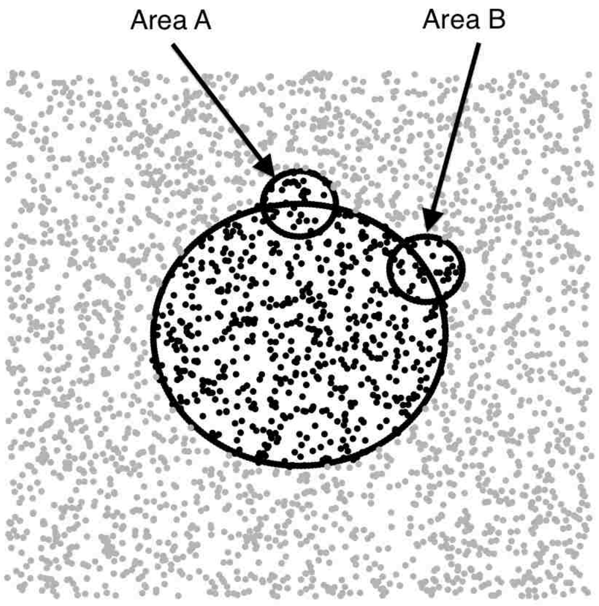
A simple visual example is outlined in Fig. 1.1, where the data set with protrusions around the circular region (Fig. 1.1(b)) might be considered more interesting than the simpler circular region (Fig. 1.1(a)). The protrusions serve as critical nuggets of information that are more interesting as these areas can be studied further for improved classification results. These protrusions also do not exhibit any outlier properties and hence outlier detection methods are not suitable for identifying them.

In real life, one such example is if one were asked to identify benign tumors that are very close to becoming malignant. Such data records, if they were to exist in a data set, would not 'deviate so much' from both benign and malignant observations, but instead would lie extremely close to the class boundary separating the benign and malignant classes. They may not necessarily 'deviate enough' to be captured by distance-based outlier detection methods.

In tight elections, the undecided voters are crucial in deciding the outcome. The problem of identifying the undecided voters and the attributes that can tilt them to the opposite side is valuable information. Another example is to predict cases from bank loan data that are very close to bankruptcy. In this setting, the important task is to identify cases before they become bankrupt. In many applications the problem is not of finding individual outliers, but instead, of finding critical nuggets (subsets of data) that provide valuable information which in turn can be used for improved classification results and a better understanding of false positive and false negative errors.



(a) An uninteresting circular region.



(b) Interesting protrusions around the circular region.

FIGURE 1.1. Grey dots indicate points of one class (−) while black dots indicate points of another class (+).

This research work considers the notion of identifying subsets of critical data instances in data sets. Critical nuggets of information can take the following form during classification tasks: data instances that lie very close to the class boundary and are sensitive to small changes in attribute values, such that these small changes result in the switching of classes. Such critical nuggets have an intrinsic worth that far out-weighs other subsets of the same data set. In classification tasks, consider a data set that conforms to a certain representation or a classification model. If one were to perturb a few data instances by making small changes to some of their attribute values, the original classification model representing the data set changes. Also, if one were to remove those data instances, the original model could change significantly. The magnitude of changes to the original model provides clues to the criticality of such data instances, as more critical data instances tend to impact the model more significantly than data instances that are comparatively non-critical. This idea is exploited in this research work to introduce the notion of critical nuggets, to define a metric for criticality and for the eventual mining of critical nuggets.[2]

1.3 Outline

This thesis provides in detail, the research work done in identifying critical nuggets in several 2-dimensional synthetic data sets and real world data sets. The work also outlines how the identified critical nuggets were used to improve the quality and accuracy of classification systems. This dissertation can be broadly divided into 3 parts:

- Motivation for critical nuggets and some algorithmic heuristics for deriving them.

[2]From this point on, the use of the terms 'critical nuggets' and 'critical sets' refer to the same concept.

- Improvement of classification accuracies using critical nuggets.

- Minimization of misclassification costs using critical nuggets.

In more detail, this research effort makes the following specific contributions:

- It defines a metric (CR_{score}, where CR stands for critical) to measure the criticality of subsets in a data set. A number of definitions of criticality have been explored. Though some of the definitions considered showed some initial promise, only one captured the nature of a critical nugget. Experiments were conducted to validate the most appropriate metric.

- Using the metric CR_{score}, experiments were conducted to identify the location and features of critical nuggets. This information was used to narrow the search space for finding critical nuggets.

- The CR_{score} provides for a rank ordering of nuggets. Only the sets that have a high CR_{score} are considered as critical nuggets. An algorithm is proposed to discover critical nuggets using the proposed CR_{score}. A simple test is provided that helps in resolving any conflicting scores revealed by the metric. The methodology was tested on some 2-dimensional geographical data sets to visually validate the proposed theories and then applied on some multi-dimensional real world data sets from the UCI machine learning repository [17]. On the real world data sets, the proposed methodology revealed that only a small number of nuggets, whose scores (using the proposed CR_{score}) were significantly higher than the rest of the subsets, need to be considered and can be characterized as true critical nuggets.

- Certain properties such as duality and the effect of the size of critical nuggets (sets) on the CR_{score} are explored. Some experimental validations for those properties are provided as well.
- Next, an experimental analysis is provided on how critical nuggets are used to improve the accuracy of classification tasks.
- As another application, critical nuggets are used to minimize misclassification costs. Some heuristic based approaches are provided for reducing misclassification costs. A detailed experimental analysis proved that critical nuggets can be used to minimize misclassification costs in classification systems.

This work is organized as follows. Chapter 2 provides a discussion on some related work from the literature. A motivation for finding critical nuggets is also provided. In Chapter 3, some ideas are explored to define a metric for criticality. After some validation experiments with some of the explored ideas, the criticality metric that best captured the notion of criticality was defined. This metric is used to calculate the criticality of a subset of data instances and is denoted as CR_{score}. In this chapter, algorithms have also been provided to help in calculating the metric and to reduce the search space by isolating approximate class boundaries in training data. In Chapter 4, a set of experiments are presented. The experiments were conducted on some real world data sets. For each of the data sets, critical nuggets were identified and properties such as the ones related to duality relations were validated. Experimental results on how critical nuggets can help in improving classification accuracy and minimizing classification costs have been provided. Chapter 5 provides the motivation for reducing misclassification costs and applying the knowledge of critical nuggets to that problem. Some methodologies are outlined to help minimize misclassification costs. Chapters 6 and 7 provide a detailed experimental and computational analysis using various data sets to highlight how critical nuggets can be applied to reduce misclassification costs. Chapter 8 provides concluding remarks and some ideas for future work.

Chapter 2
Motivation For Critical Nuggets

2.1 Related Work

Given the initial introduction to the problem in Chapter 1, can existing approaches in outlier detection help in finding critical nuggets? In recent data mining literature, there have been a number of research efforts directed towards outlier detection, including recent work such as [31]. Outlier detection has several applications including detecting fraud in business transactional data [30], identifying network intrusions [30], isolating abnormal trends in time-series data [47] and picking out suspicious criminal activity [51]. The concept of distance-based outliers was proposed in [29] to identify records that are different from the rest of the data set. Distance-based measures as in [3], [44] and [20] have been used in algorithms to delineate outliers or abnormal records from normal records. A lot of work in data mining has also been devoted to finding interesting patterns or association rules in data sets. . However, not much work has focussed on finding critical nuggets of information that may be hidden in data sets.

Critical nuggets in certain cases may involve outliers, but this may not always be true. In the example of the previous chapter, cells in tumors may not show anomalous behavior on an individual basis but collectively, such cells may contain critical pieces of information. In [8], the authors note that the performance of a distance-based outlier detection method *'greatly relies on a distance measure, defined between a pair of data instances, which can effectively distinguish between normal and anomalous instances. Defining distance measures between instances can be challenging when the data is complex.'* Moreover, critical nuggets that belong to a data set may not be at a great ‘distance’ from the other ‘normal’ points, and may end up being classified as ‘normal.’ For a comprehensive survey on outlier detection methods, please refer to the extensive survey in [8]. In the field of distance-based outlier detection, researchers have focussed on proposing algorithms that reduce the time complexity $O(n^2)$ of calculating distances [5], [4], [44] and [20].

Work has also been done on density-based outlier detection such as [6] where outliers are defined as objects that show anomalous trends with respect to their local neighborhoods and tend to lie in a less dense area with respect to a more dense local neighborhood. In [13], the concept of density-based detection is extended to cluster-based outlier detection where the approach does not only find single point outliers but instead clusters of outliers. However, cluster-based outliers may not necessarily lead to critical nuggets and may not carry information that is critically important to the data domain. Intuitively, cluster-based outlier methods may not necessarily lead to identifying critical areas such as the protrusions in Fig. 1.1(b) which do not lie at a great ‘distance’ from the rest of the points. With this differentiation between critical nuggets and outliers, can critical nuggets be found among data records near the boundary? One can utilize this intuition as a motivation in the derivation of critical nuggets. This is elaborated more in the ensuing section.

This work also uses critical nuggets to improve the classification accuracy of classification algorithms and to minimize misclassification costs. Classification accuracy improvements has been used as a benchmark to rate the performance of a newly proposed classification methodologies when compared with existing methodologies. Work such as [1] and [32], and recent work such as [35] have used classification accuracy as a benchmark to compare the predictive abilities of a classification methodology. Another performance metric that is frequently used to gauge the performance of a classification task is misclassification cost. The basic idea is that the predictive ability of a proposed classifier is able to reduce the number of wrong classifications/predictions. Misclassification cost reduction is used in work such as [48]. Examples of misclassification cost being used can be found in classification algorithms being applied in diverse fields such as environmental science/hydrology ([14]) and finance ([23]).

In order to evaluate the performance of a new classification methodology, it has become necessary to use statistical procedures to test if the improvements in a performance metric (such as classification accuracy or classification costs) are statistically significant. In the field of data mining and classification algorithms, work such as [10] and [11] have recommended the use of non-parametric statistical tests to rigorously compare the proposed methodology with existing classification methods. Such related work ([10] and [11]) have recommended non-parametric tests such as the Wilcoxon signed-ranks test [49] for comparing 2 classifiers (one of the reasons for choosing Wilcoxon is that it does not assume normal distributions). Work on reduction of misclassification costs such as [39] and [40] have ignored such statistical rigor whereas work such as [43], [48] and [35] have included non-parametric statistical tests to ensure that improvements in a proposed classifier's performance are statistically significant.

2.2 Motivation

The problem of finding critical nuggets brings up two important questions - how does one define criticality and where to search for such critical nuggets? Intuitively, one can look at criticality as the intrinsic worth of a subset of records. This worth is realized when the records are collectively removed from the data set or their attribute values undergo perturbation. This intuition is elaborated in the subsequent sections, during the derivation of a metric for measuring criticality. In order to answer the second question, as to where to search for such critical nuggets, one can utilize an intuition that is motivated by two commonly occurring scenarios in classification algorithms.

1. **Points near the boundary, in general, are critical:** The deciding factor for most classification algorithms is how accurately the algorithm classifies the points near the class boundaries (see also [45]). The points that are far from the class boundaries are the "slamdunk", easy cases, where the impact of misclassification is pretty minimal. However, the points near the class boundaries are more susceptible to misclassification. These points are critical in deciding the accuracy of any classification algorithm. The need for understanding this problem can be best explained by the real world example of a data set describing some type of cancer related cases. Most classification algorithms can easily classify a full-blown cancer case or a clearly cancer-free case. On the other hand, the border-line cases which may exhibit subtle symptoms of cancer are critical, as early detection can save a life. Hence, uncertain regions in and around the class boundaries can be crucial for identifying critical nuggets.

2. **Boundary features can be critical:** Secondly, as a corollary to the first scenario, there are certain regions along the boundary where the problem of classification becomes more difficult. As a simplistic example, consider a geographical data set that corresponds to a political boundary. Classifying records near sharply changing outlines (such as along a complex sea coast of a political boundary) is more difficult than straight edges of the boundary. For more complex data sets, there maybe certain inherent complex properties

that render the points near the boundary difficult to classify. Such regions have a higher potential for harboring critical nuggets.

In summary, using the first scenario, the search for critical nuggets is narrowed to a region near the boundary separating the classes. On the basis of the second scenario, where certain boundary features are more complex than others, the criticality metric (the CR_{score}) has been defined in such a way that it yields higher scores for sets of data records that lie near complex boundary features. In other words, the greater the complexity of a boundary feature, the higher the probability of misclassification. This results in higher scores being assigned for points near that complex boundary.

Chapter 3

Critical Nuggets and Problem Description

3.1 Formal Notation

Consider a training data set T_r comprised of m data instances, n attributes and two classes, denoted as '+' and '-' (these names are arbitrary). Consider also a sample neighborhood N to be a subset of T_r, comprised of d data instances (i.e., number of rows in N) of the data set T_r.

Besides the above notation, the following notation will also be used:

- C - any classification algorithm.
- T_r^+ - the subset of T_r comprised of only the '+' class.
- T_r^- - the subset of T_r comprised of only the '-' class.
- A - the set of attributes in T_r denoted as $\{A_1, A_2, A_3, ..., A_n\}$.
- D - the set of the data instances in T_r denoted as $\{D_1, D_2, D_3, ..., D_m\}$.
- d_j^+ - the number of instances in N that switch classes when attribute A_j is increased by δ_j.
- d_j^- - the number of instances in N that switch classes when attribute A_j is decreased by δ_j.
- $N[A_j]$ - column vector of size $d \times 1$, formed by choosing only attribute A_j from matrix (neighborhood) N.
- $N[A_1 : A_j]$ - matrix of size $d \times j$, formed by choosing attributes A_1 through A_j.

- $N_1.N_2$ - appending two matrices, column-wise (e.g., if N_1 and N_2 were each of size 2×3, the combined matrix would be of size 2×6).

- M_0 - the model obtained by using classification algorithm C on training set T_r.

- P_0 - the vector of predicted class values by model M_0 when applying M_0 on a neighborhood of instances, N.

- B^+ - the set of '+' points near the boundary separating the two classes, '+' and '-'.

- B^- - the set of '-' points near the boundary separating the two classes, '+' and '-'.

- $\overline{B^+}$ - the set of '+' points not near the boundary separating the two classes, '+' and '-'.

- $\overline{B^-}$ - the set of '-' points not near the boundary separating the two classes, '+' and '-'.

From the above definitions, it follows that: $|T_r^+| = |B^+| + |\overline{B^+}|$, $|T_r^-| = |B^-| + |\overline{B^-}|$ and $|T_r| = |T_r^+| + |T_r^-|$, where $|X|$ denotes the cardinality of set X.

3.2 Definition of Criticality

One can look at criticality as the intrinsic worth of a subset of records. This worth is realized when the records are collectively removed from the data set or their attribute values undergo perturbation. Initial steps in defining the critical metric (CR_{score}) relied on the effect of removing a neighborhood of data instances on a classification model. A classification model M_0 was initially derived by applying a classification algorithm C on the training data T_r. Then a neighborhood of data instances, N, was removed from T_r and a new classification model M_1 was obtained by applying C on $T_r - N$. The difference in predictions made by M_0 and M_1, divided by the number of data instances in N, was initially used as the criticality measure, CR_{score}. The greater the difference in predictions between M_0 and M_1, the higher the CR_{score} was and vice versa. Some 2-dimensional data sets were used to validate this approach. However, this metric could not isolate all the critical areas even though some of the critical areas were obvious during a simple visual inspection of the 2-dimensional validation data set. Hence, a different approach was considered and upon validation using the 2-dimensional data sets, this new approach in deriving the CR_{score} is outlined. More experiments (as described in Section 4) with some real world data sets further support the choice of this metric.

Consider a training data set T_r with m data instances, each instance having n attributes denoted as A_j ($j \epsilon \{1, 2, ..., n\}$). The underlying assumption is that all attributes are numeric and not categorical. From T_r, form a neighborhood N, by choosing a data instance D_i as a center and finding a group of points that belong to the same class as D_i and lying within a distance R from D_i. For simplicity, let us say that the neighborhood N is comprised of d data instances. The selection of parameters R and D_i used in forming a neighborhood N is further described in Section 3.8. First, a classification model M_0 is generated by applying a classification algorithm C to the training data set T_r. Using the classification model M_0, one can predict the class labels for the different data instances in question. For the d instances in neighborhood N, consider an attribute A_j. Also, for the d instances, the attribute A_j can be increased or decreased in magnitude. A parameter denoted by δ_j is used for this and δ_j varies for different attributes in neighborhood N. The calculation of this parameter is further explained in Section 3.5. After increasing A_j by an extent δ_j for just the d instances, the classification model M_0 for the new class labels for the d instances is queried. The average number of data instances that have switched classes in neighborhood N is computed and is denoted as w_j^+. If all the data instances in N switch classes, then one can infer that N is very sensitive to changes with respect to attribute A_j. The same test is applied on N by decreasing A_j by the same extent δ_j and find w_j^- by querying the classification model M_0 for the new class labels. For the attribute A_j, the average of w_j^+ and w_j^- is computed to get w_j. Repeating this process for all n attributes, the average of the w_j scores is computed as the CR_{score} for the neighborhood N.

Formally, the critical score is defined as

$$CR_{score} = \frac{\sum_{j=1}^{n}(w_j)}{n}, \tag{3.1}$$

where: $w_j = \frac{w_j^+ + w_j^-}{2}$, $w_j^+ = \frac{d_j^+}{d}$ and $w_j^- = \frac{d_j^-}{d}$.

3.3 Properties of the Critical Score

Based on the score developed above, the following properties are outlined:

- Each w_j value lies in the interval [0,1]. Each w_j value is calculated by averaging w_j^+ and w_j^- and $w_j^+ \epsilon [0,1]$ and $w_j^- \epsilon [0,1]$. Hence, $w_j \epsilon [0,1]$.

- $CR_{score} \epsilon [0,1]$, as there are n instances of w_j and CR_{score} is averaged over n.

- In calculating the critical score, the main idea is to find as many attributes that are sensitive to small changes (such as the increase and decrease of A_j by δ_j) that propel an entire subset from one class to another. The greater the number of attributes that are sensitive to such changes, the higher is the resulting CR_{score}.

- A neighborhood of data instances N_1 is said to be more critical than a neighborhood N_2, if and only if $CR_{score}(N_1) > CR_{score}(N_2)$.

3.4 Computing the CR_{score}

Using the description in Section 3.2 on how the CR_{score} is calculated, the algorithm $GetNuggetScore$ is developed and is outlined as Fig. 3.1.

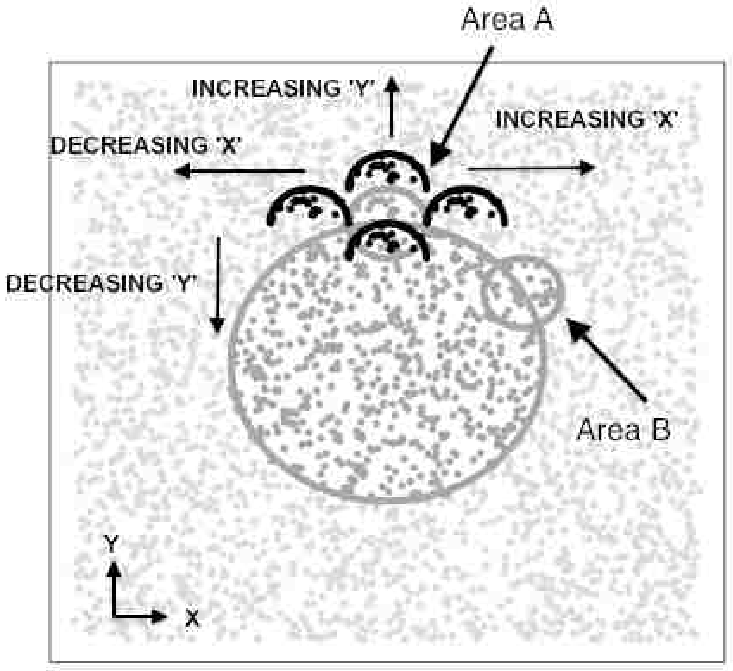
```
Require: $T_r$: the training set, $N$: a neighborhood of data instances and $R$: a distance
    parameter used in creating the neighborhood set, $N$
1:  $M_0$ = Model resulting from applying $C$ on training set, $T_r$
2:  $m$ = number of data instances in $T_r$
3:  $n$ = number of attributes in $T_r$
4:  $ScoresArray = \phi$
5:  for each $j$ in {1,2, ..., $n$} do
6:      $\delta_j$ = max($N[A_j]$) - min($N[A_j]$) {Finding the maximum and minimum values in
        vector $N[A_j]$}
7:      if $\delta_j$=0 then
8:          $\delta_j = R$
9:      end if
10:     $V = N[A_j] + \delta_j$ {Extract $A_j$, increment all values in $A_j$ by $\delta_j$}
11:     $N_1 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$ {Generate new matrix, keep previous columns
        and replace $A_j$ by $V$ instead}
12:     $P_0$ = Query $M_0$ to obtain new class labels for $N_1$
13:     $w_j^+$ = Average number of instances in $P_0$ that have switched classes
14:     $V = N[A_j]$ - $\delta_j$ {Decrement all values in column $A_j$ by $\delta_j$}
15:     $N_2 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$
16:     $P_0$ = Query $M_0$ to obtain new class labels for $N_2$
17:     $w_j^-$ = Average number of instances in $P_0$ that have switched classes
18:     $w_j = (w_j^+ + w_j^-)/2$
19:     Append $w_j$ to $ScoresArray$
20: end for
21: $CR_{score} = \frac{\sum(ScoresArray)}{n}$
22: return $CR_{score}$
```

FIGURE 3.1: The $GetNuggetScore$ algorithm.

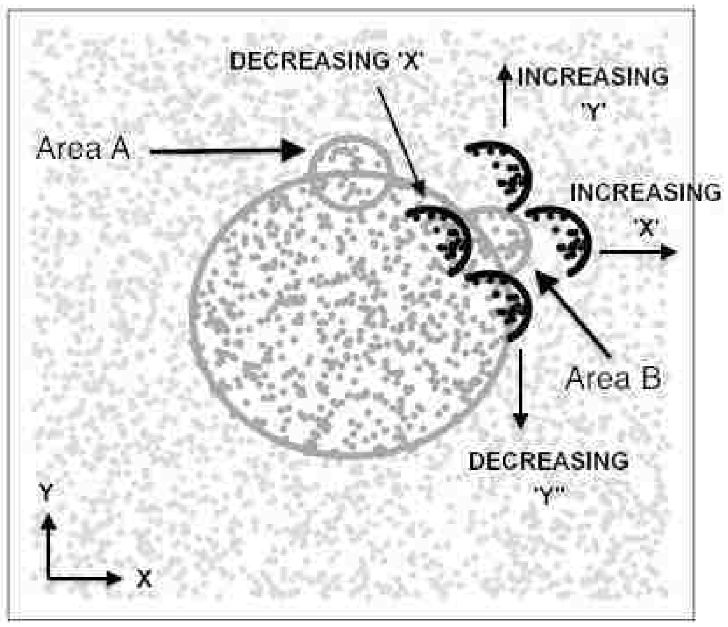
The computational complexity of the algorithm is derived as follows. Deriving the model M_0 is dependent on the complexity of the chosen classification algorithm (C). The complexity of the classification algorithm is denoted as $t(C)$. Each attribute A_j is analyzed by checking if increasing or decreasing the values of the attributes by an extent δ_j, switches the class label. Hence, for each attribute, the model M_0 is queried twice. There are d data instances in N and thus for each attribute there are $2 \times d$ queries. Since there are n attributes, the complexity of the for-loop in Fig. 3.1 is $O(dn)$. When $d \ll n$, the complexity of the for-loop becomes $\approx O(n)$. The total complexity of the algorithm is $O(t(C) + dn)$ ($\approx O(t(C) + n)$ when $d \ll n$).

3.5 Choosing δ_j

In the algorithm *GetNuggetScore*, the parameter δ_j is used. This parameter is a measure of how much the attribute values should increase or decrease. For each attribute A_j, an appropriate value of δ_j is calculated by computing the range of values of A_j in neighborhood N. This is to ensure that all the attribute values A_j get a chance to switch class labels. Fig. 3.2 is used to analyze a closely related issue. The data set in Fig. 3.2 has two attributes - let us say attributes 'X' and 'Y' and two subsets of interest denoted as area 'A' and area 'B.' Considering Fig. 3.2(a), increasing and decreasing the attribute values of 'X' by δ_j results in a majority of area A to be inside the opposite class. However, increasing and decreasing attribute values along 'Y', results in the switching of class values only when increasing 'Y' values. So, for this example, only along 3 out of 4 directions (increasing and decreasing of attributes), can there be a switch of classes. In this case, in 3 directions, all points in the neighborhood ended up shifting to the opposite class. Hence, using the defined CR_{score}, the score for area A would be $3/4 = 0.750$. Performing a similar analysis on area B, reveals (Fig. 3.2(b)), that in only 2 out of 4 directions (increasing attributes 'X' and 'Y') result in switching of class labels for all points in the neighborhood. Decreasing the attribute Y by δ_j results in only a portion of the points moving to the opposite class, ultimately, reducing the CR_{score}. Assuming that only half the points in the neighborhood switch classes when decreasing the attribute Y, one would have a CR_{score} of (1+1+ 0.5)/4=0.625.



(a) An example to illustrate shifting of points in neighborhood 'Area A'.



(b) An example to illustrate shifting of points in neighborhood 'Area B'.

FIGURE 3.2: Shifting of data instances.

The above analysis reveals that two subsets (i.e., areas A and B in Fig. 3.2), which in reality should be considered of equal importance when the entire subset is considered may end up with different CR_{score} values. In other words, although area A plays the same role as area B, the two end up with considerably different CR_{score} values (e.g., 0.750 and 0.625 respectively).

The cause of this phenomenon is the relative orientation of the axes system and the two areas (subsets) in general. This was indicated in the analytical steps described earlier in deriving those two values. This can be easily remedied as follows. We rotate one attribute (say axis 'X' in Fig. 3.2) with respect to another attribute (axis 'Y' in Fig. 3.2). We consider a sequence of rotations by some angle θ until a complete rotation of $360°$ is achieved. The summation of weights, w_j^+ and w_j^- (equation (1)) are recorded at each step. At the end of these iterations, the maximum value of the sum of weights, w_j^+ and w_j^- is returned and recorded in memory for each attribute being considered. The final CR_{score} value for a given subset is the average of all the previously recorded w_j^+ and w_j^- values, after all the attributes have been considered. When this approach is used on the example depicted in Fig. 3.2, then both areas A and B are assigned similar CR_{score} values (i.e., the value of 0.750).

The above remedied approach is incorporated as a modification to the $GetNuggetScore$ algorithm. The modified algorithm is called the $GetNuggetScoreRevised$ algorithm. The algorithm's steps are provided in Fig. 3.3. The modified algorithm includes a call to the $RotationTest$ heuristic, outlined as Fig. 3.4, which attempts to resolve the above discussed problem of two similar areas receiving different or non-representative scores. In the $GetNuggetScoreRevised$ algorithm one tests for non-representative results for each attribute (lines 18-32). Recall from Fig. 3.2(b), that a non-representative result occurred during the following scenario. When the points in the neighborhood were shifted along one direction, the class labels switched for all the points in the neighborhood. However, when the points were shifted in the opposite direction, it resulted in only a partial switching of class labels. A threshold parameter is used to test what percentage of instances in a neighborhood ended up switching class labels when a certain attribute value is increased or decreased. If all the instances end up switching class labels during either increasing or decreasing an attribute's values, then the algorithm $RotationTest$ need not be invoked since the non-representative scores result only during partial switching of class labels. If none of the instances end up switching class labels along both directions, there is no necessity to invoke the $RotationTest$ either. However, if all the instances end up switching class labels along one direction and a partial switching of labels occurs in the opposite direction, then one can use the threshold parameter to decide whether to invoke the $RotationTest$ or not.

If one wishes to minimize the number of calls to this test, then the threshold parameter can be set as high as 1, in order to short circuit the test and reduce the computation time. However, if the need is to ensure that all critical nuggets are mined out without any conflict, then one can lower the threshold to a value between 0.5 and 1. Setting a threshold of less than 0.5 is not necessary, as that would mean the attribute in question is not switching labels when shifted in both directions. If the condition in line 30 is satisfied (an XOR boolean operation is used), then the $RotationTest$ is invoked. The key idea in the $RotationTest$ is summarized as follows: for each attribute A_j, rotate the values corresponding to A_j by an angle θ with respect to another attribute A_k ($j \neq$k). For each of the different angles considered and the different attributes A_k, the sum of weights, w_j^+ and w_j^- is computed and recorded. After all the angles have been considered, the maximum value among the recorded sum of weights $w_j^+ + w_j^-$ is chosen. If the $RotationTest$ is

```
Require: $T_r$: the training set, $N$: a neighborhood of data instances and $R$: a distance
    parameter used in creating the neighborhood set, $N$
1: $M_0$ = Model resulting from training $C$ using training set, $T_r$
2: $m$ = number of data instances in $T_r$
3: $n$ = number of attributes in $T_r$
4: $ScoresArray = \phi$
5: for each $j$ in {1,2, ..., $n$} do
6:     $\delta_j$ = max($N[A_j]$) - min($N[A_j]$)
7:     if $\delta_j$=0 then
8:         $\delta_j = R$
9:     end if
10:    $V = N[A_j] + \delta_j$ {Extract $A_j$, increment all values in $A_j$ by $\delta_j$}
11:    $N_1 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$ {Generate new matrix, keep previous columns
       and replace $A_j$ by $V$ instead}
12:    $P_0$ = Query $M_0$ to obtain new class labels for $N_1$
13:    $w_j^+$ = Average number of instances in $P_0$ that have switched classes
14:    $V = N[A_j]$ - $\delta_j$ {Decrement all values in column $A_j$ by $\delta_j$}
15:    $N_2 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$
16:    $P_0$ = Query $M_0$ to obtain new class labels for $N_2$
17:    $w_j^-$ = Average number of instances in $P_0$ that have switched classes
18:    threshold=1
19:    if $w_j^+ \geq threshold$ then
20:        $up\_counter = True$
21:    else
22:        $up\_counter = False$
23:    end if
24:    if $w_j^- \geq threshold$ then
25:        $down\_counter = True$
26:    else
27:        $down\_counter = False$
28:    end if
29:    $sum\_score = w_j^+ + w_j^-$
30:    if $(up\_counter \oplus down\_counter) = True$ then
31:        $sum\_score$=RotationTest($M_0$,$N$,$A_j$,$R$)
32:    end if
33:    $w_j = (sum\_score)/2$
34:    Append $w_j$ to $ScoresArray$
35: end for
36: $CR_{score} = \frac{\sum(ScoresArray)}{n}$
37: return $CR_{score}$
```

FIGURE 3.3: The *GetNuggetScoreRevised* algorithm using the *RotationTest*.

invoked for each and every attribute, then the combined complexity of algorithms $GetNuggetScoreRevised$ and $RotationTest$ will be $O(n^2)$ (since the number of angles considered is a constant). However, during the experiments with some real world data sets, the data in some of the data sets was such that the complexity of finding a CR_{score} for each neighborhood was far lower than the worst-case theoretical complexity of $O(n^2)$. The methodology for the RotationTest is outlined in Fig. 3.4.

3.6 Searching Near The Class Boundary

Using the methodology for finding the CR_{score}, our goal is to find critical nuggets in T_r. The brute-force method would be to exhaustively examine all possible subsets, calculate their CR_{score} values and choose the critical nuggets based on the ordering of the CR_{score} values. However, for a large data set, this would be computationally cumbersome due to the combinatorial explosion of the problem. The question then becomes: How can one computationally mine for such small-sized critical nuggets in large data sets?

Since the brute-force method of investigating all possible combinations, would be computationally hard, one can look at candidate sets that have a high likelihood of being critical nuggets. A possible area that can be investigated is near the class boundary that separates the classes of the training set. The basis for this is that points near the boundary are more susceptible to switching of classes. When certain attribute values of boundary points [1] are perturbed, the chances of a boundary point switching to the opposite class are higher than a point deep in the interior.

In order to validate the idea that boundary points have higher potential of having high CR_{score} values, an experiment was conducted. The 2-dimensional ran-

[1]From this point on, the use of the terms 'boundary points' and 'points near the boundary' refer to the data instances that lie on or very close to the class boundary separating the two classes

```
Require: $M_0$: Model, $N$: a neighborhood of data instances, $A_j$: an attribute and $R$: a
    distance parameter used in creating the neighborhood set, $N$
 1: $Array = \phi$
 2: for each $\theta$ in {10,20, ..., 360} do
 3:    $TempArray = \phi$
 4:    $\delta_j$ = max($N[A_j]$) - min($N[A_j]$)
 5:    if $\delta_j$=0 then
 6:       $\delta_j = R$
 7:    end if
 8:    for each $k$ in {1,2, ..., $m$} and $k$ != $j$ do
 9:       $\delta_x = \delta_j * cos((\pi/180) \times \theta)$
10:       $\delta_y = \delta_j * sin((\pi/180) \times \theta)$
11:       $V_j = N[A_j] + \delta_x$
12:       $V_k = N[A_k] + \delta_y$
13:       if $j < k$ then
14:          $N_1 = N[1 : A_{j-1}].V_j.N[A_{j+1} : A_{k-1}].V_k.N[A_{k+1} : A_n]$
15:       else
16:          $N_1 = N[1 : A_{k-1}].V_k.N[A_{k+1} : A_{j-1}].V_j.N[A_{j+1} : A_n]$
17:       end if
18:       $P_0$ = Query $M_0$ to obtain new class labels for $N_1$
19:       $w_j^+$ = Average number of instances in $P_0$ that have switched classes
20:       $V_j = N[A_j]$ - $\delta_x$
21:       $V_k = N[A_k]$ - $\delta_y$
22:       if $j < k$ then
23:          $N_2 = N[1 : A_{j-1}].V_j.N[A_{j+1} : A_{k-1}].V_k.N[A_{k+1} : A_n]$
24:       else
25:          $N_2 = N[1 : A_{k-1}].V_k.N[A_{k+1} : A_{j-1}].V_j.N[A_{j+1} : A_n]$
26:       end if
27:       $P_0$ = Query $M_0$ to obtain new class labels for $N_2$
28:       $w_j^-$ = Average number of instances in $P_0$ that have switched classes
29:       Append $(w_j^+ + w_j^-)$ to $TempArray$
30:    end for
31:    Append $max(TempArray)$ to $Array$ {i.e., find the max score among the $k$ at-
       tributes}
32: end for
33: return $max(Array)$
```

FIGURE 3.4: The *RotationTest* method.

domly generated synthetic data set depicted in Fig. 1.1(b) was used for this experiment. Fig. 3.5 is the result of the experiment. For each point of class '+', the $GetNuggetScoreRevised$ algorithm was used to find a list of scores. The scores were then plotted as a heat map with higher scores being marked as darker grey dots and lower scores being marked as lighter grey dots. The heat map is represented in Fig. 3.5. It can be observed that there are darker shades along the boundary as compared to the interior. This indicates that the potential of finding critical nuggets is higher along the boundary. Hence, one can focus the search along the boundary as compared to the interior. This would greatly reduce the search space as well. Similar experiments were conducted with the two dimensional sets and similar conclusions were reached.

In order to find an approximate boundary set from the training data, a boundary detection algorithm is proposed. The algorithm is tested using the 2-dimensional randomly generated synthetic data set depicted in Fig. 1.1(b). In [36], the authors proposed a boundary detection algorithm to speed up classifications by Support Vector Machines (SVMs). Though our boundary detection algorithm is similar in spirit to the one in [36], our methodology is simpler as our goal in finding an approximate boundary is merely to reduce the search space in finding the critical nuggets. The proposed algorithm uses Euclidean distances to rank the distances between points. The algorithm, $FindBoundary$, is outlined in Fig. 3.6. The algorithm works in two phases since this study focusses on two-class classification problems. In each phase, a boundary set is isolated for each class in the data set. So for isolating the boundary points for the '+' class, the algorithm works by calculating distances to all points in T_r^- from each point in T_r^+. For every point in T_r^+, 5 closest T_r^- points are chosen and the average of the 5 distances is calculated (S_i). The score S_i, computed for each of the T_r^+ points, is sorted and stored in

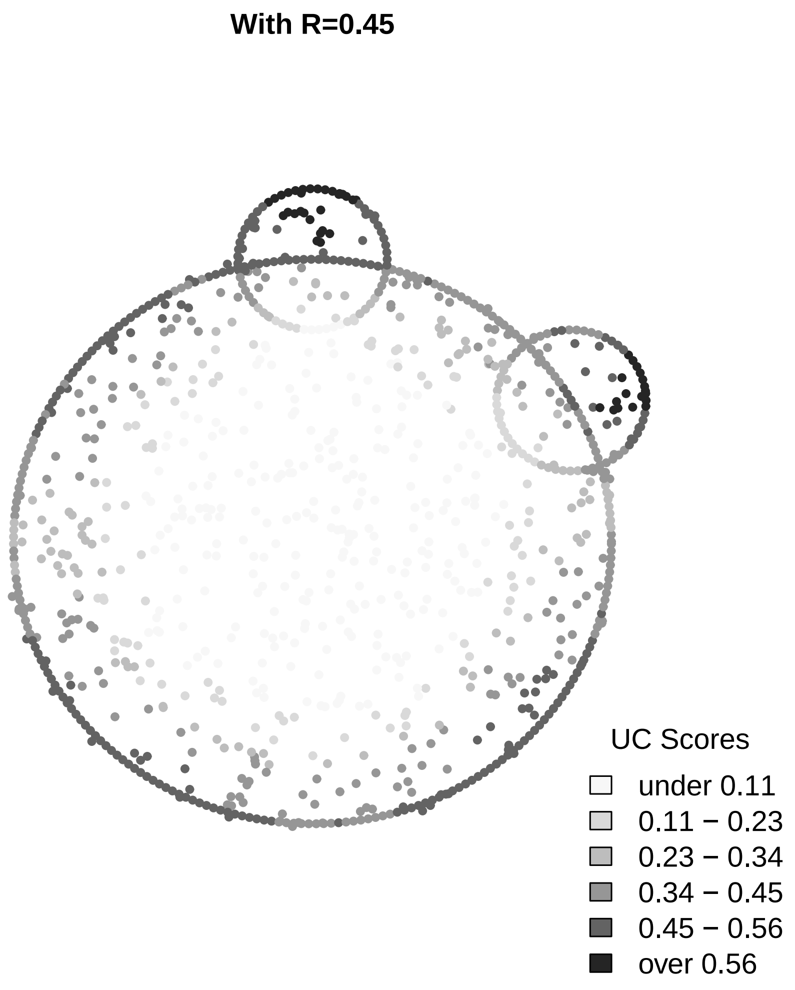


FIGURE 3.5: Boundary Points Having Higher CR_{score} values.

```
Require: $T_r^+$ and $T_r^-$: The training sets for classes '+' and '-'
 1: Initialize array $S = \phi$
 2: for each point $i$ in $T_r^+$ do
 3:     Calculate distance to all points in $T_r^-$
 4:     Find the nearest 5 $T_r^-$ points to $i$
 5:     Using the above top 5 distances, compute an average to get a score $S_i$
 6:     Store each $S_i$ score in list $S$
 7: end for
 8: Sort all the scores in the list $S$
 9: Plot a graph of sorted scores in S.
10: Find cut-off index $i$ where the slope of the graph starts increasing drastically. {Points
    near the boundary will have smaller scores and as one moves away from the boundary,
    the scores start increasing drastically.}
11: Points corresponding to scores $S_1$, $S_2$, ..., $S_i$ in list $S$ form the boundary set $B^+$.
12: return Boundary Set $B^+$.
13: Repeat algorithm with $T_r^-$ in the role of $T_r^+$ to get Boundary Set $B^-$.
```

FIGURE 3.6: The *FindBoundary* algorithm.

a list S. To find the boundary set, one needs a cut-off point that would provide the required subset. Using a visualization technique, a graph of the scores in S is plotted. The points that lie closest to the class boundary will have the smallest average distances. A cut-off point is chosen where the slope of the graph increases sharply, indicating a threshold, beyond which, includes interior points with higher S_i scores. This procedure is then carried out again for isolating boundary points for the '-' class.

3.6.1 Complexity of the *FindBoundary* algorithm

The complexity of the *for-loop* in the *FindBoundary* algorithm is: $O(|T_r^+| \times |T_r^-|) = O(m^2)$. Sorting takes $O(mlogm)$. Thus the total complexity is $O(m^2) + O(mlogm) = O(m^2)$.

3.7 The *FindCriticalNuggets* Algorithm

The *FindCriticalNuggets* algorithm works in two phases where in each phase, it identifies critical nuggets for each one of the two classes. Using the reduced boundary set for each class, the data instances in the boundary set are considered one at a time. Each data instance in the boundary set is considered as a center for a neighborhood. A neighborhood is formed by finding all points that belong to the same class and lie within a distance R from the center point. One class at a time is considered since the goal is to find critical nuggets that belong to one class but switch to the other class when their attribute values are perturbed (a total of two classes is assumed). If there are $|B^+|$ data instances in the boundary set which belong to the same class (say '+'), one can form $|B^+|$ neighborhoods by considering each instance in B^+ as a center. For each of the $|B^+|$ neighborhoods, the CR_{score} is computed. The scores are then ranked and the higher scores are used to identify the critical nuggets in T_r^+. In the second phase, the other class (say '-') is considered. Hence $|B^-|$ neighborhoods are then considered to compute the CR_{score} values which in turn are sorted and ranked to identify critical nuggets in T_r^-. The algorithm is outlined in Fig. 3.7.

```
Require: $T_r$: the training set and $R$: the distance parameter to form the neighborhood
    set $N$
 1: $ScoresArray = \phi$
 2: Split $T_r$ into $T_r^+$ and $T_r^-$
 3: $B^+ = FindBoundary(T_r^+)$
 4: for each $p_0$ in $B^+$ do
 5:     $N = \{x \mid x \in B^+ \wedge |x - p_o| \leq R\}$ {Finding same class points, within a distance
        $R$ from $p_o$}
 6:     $CR_{score} = GetNuggetScoreRevised(T_r, N, R)$
 7:     Append $CR_{score}$ to $ScoresArray$
 8: end for
 9: Sort (descending) and rank scores in $ScoresArray$
10: Plot sorted scores in $ScoresArray$ as a histogram and use the histogram to find
    index $k$ that separates the highest $k$ scores from the rest of the scores.
11: Use $k$ to find top $k$ Critical Nuggets for class '+'.
12: Re-initialize ScoresArray and repeat steps 2-11 with $B^-$ in the role of $B^+$.
```

FIGURE 3.7: The $FindCriticalNuggets$ Algorithm.

3.7.1 Overall Complexity

In summary, the process of finding critical nuggets first involves the identification of an approximate boundary set, and next considering a neighborhood around each boundary point and finding its CR_{score}. Identifying an approximate boundary involves complexity of $O(m^2)$, where m is the number of data instances in T_r. Using the boundary set and the CR_{score} values different neighborhoods are investigated. There are $|B|$ neighborhoods ($|B| \ll m$) and for each neighborhood the worst complexity (including the rotation test) is $O(dn^2)$ where d is the size of a typical neighborhood. This yields a total complexity for the entire process of identifying critical nuggets of $O(m^2+t(C)+|B|dn^2)$ which can be further simplified to $O(m^2+t(C)+n^2)$ (since $|B| \ll m$ and $d \ll n$).

3.8 Choosing R

In the *FindCriticalNuggets* algorithm, the distance parameter R is introduced to define a typical neighborhood. Choosing R is an important decision in identifying critical nuggets. Choosing a too small R value may yield single element critical nuggets (sets) while choosing a too large value of R will yield large sized neighborhoods that may not be sensitive to small changes in their attribute values. Also choosing a large value of R can increase the value of d, ultimately increasing the complexity of the algorithm. So for the experimental study, a range of R values are considered. The range of R values for our experiments depended on the data set. The general rule used during the study was to vary R in the following range $[0, x]$ - where x was a value that caused the maximum size of the neighborhood to not exceed 20% of the size of the data set. The main intuition for setting this was to limit the size of the neighborhoods, as large neighborhoods include points that are located away from the class boundary and hence are less sensitive to small changes in attribute values. Larger neighborhoods also have lower CR_{score} values and are not useful in mining of critical nuggets.

To find the critical nuggets among neighborhoods formed by different values of R, the following analysis was conducted. Using different values of R, the $FindCriticalNuggets$ algorithm was used to find the CR_{score} values for each neighborhood (each neighborhood was formed using every element in the boundary set as a center). For each value of R, the top k neighborhoods are identified based on their scores (The top k subset of neighborhoods were identified by first sorting the scores in descending order. Visualization methods such as plotting the sorted scores as a histogram can be used to identify an appropriate cut-off value of k. k is chosen by selecting a appropriate point in the histogram which delineates the small group of high scores from the majority of small scores). Therefore, if there are r values of R, one would have $k \times r$ neighborhoods. Some of the $k \times r$ neighborhoods could have been formed around the same center. Hence, among the $k \times r$ neighborhoods, the scores are ranked based on unique centers. The top k scores among the unique centers and their associated R value are used to identify the top k critical nuggets.

3.9 Duality

In the $FindCriticalNuggets$ algorithm, recall that the search for critical nuggets was on a class-by-class basis on two-class problems. Hence, for each class label, one obtains a set of critical nuggets. Since critical nuggets were identified for each of the class labels, a study was conducted to see if there were any relationships between the two sets of critical nuggets. A property that was investigated was duality - relationships between critical nuggets of the two classes. In Fig. 3.8, some critical nuggets for two classes are illustrated using darker shades of grey. The experiments mainly checked to see if the following scenarios occurred:

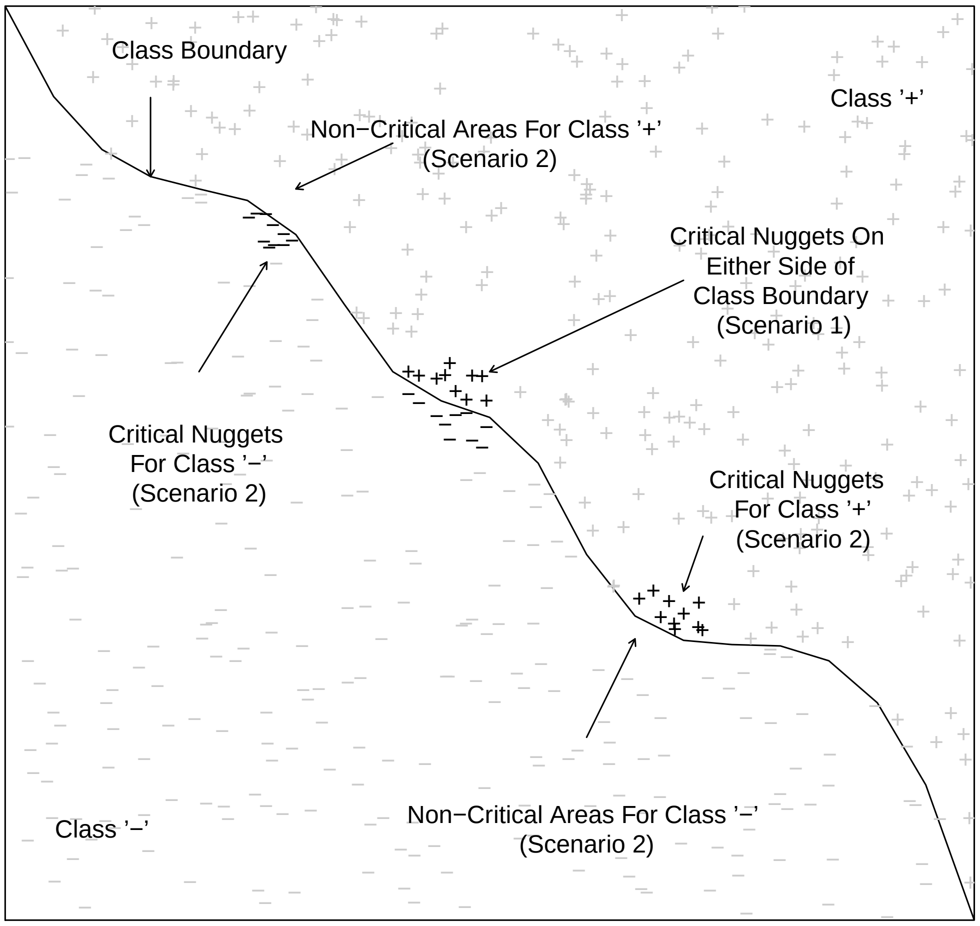

Class Boundary
Non−Critical Areas For Class '+' (Scenario 2)
Class '+'
Critical Nuggets On Either Side of Class Boundary (Scenario 1)
Critical Nuggets For Class '−' (Scenario 2)
Critical Nuggets For Class '+' (Scenario 2)
Class '−'
Non−Critical Areas For Class '−' (Scenario 2)

FIGURE 3.8: Illustration For Duality.

1. Scenario 1 - Do critical nuggets belonging to different classes lie in 'proximity' to one another? In other words, a check was done to see if the critical nuggets of different class labels lie in 'proximity' but on opposite sides of the class boundary (see Fig. 3.8 for Scenario 1).

2. Scenario 2 - Are there any sets (neighborhoods) that are non-critical for one class label, but lie in 'proximity' to a critical nugget belonging to another class and vice versa (see also Fig. 3.8 for Scenario 2)?

Both scenarios have good potential as they help in broadly dividing the data set into three regions. These regions are summarized as follows (assuming two classes '+' and '-'):

1. Region 1 - This is comprised of critical nuggets of one class that lie in close proximity to critical nuggets of the opposite class (i.e., subsets of the data set that have high CR_{score} values and lie in close proximity, but on opposite sides of the class boundary) and vice versa. This is depicted as Scenario 1 in Fig. 3.8.

2. Region 2 - Critical nuggets of one class that lie in close proximity to neighborhoods of the opposite class, neighborhoods that are not critical nuggets (and vice versa). This is depicted as Scenario 2 in Fig. 3.8.

3. Region 3 - Neighborhoods (or subsets of the training data) that lie in the interior (not near the class boundary) of either class (characterized by very low CR_{score} values).

The empirical results for this study indicate that if there are a group of nuggets from one class in close proximity with each other, then it is likely to have a corresponding group of nuggets for the other class on the other side of the boundary.

In other words, a group of nuggets for one class indicates a region of the data set of potential high interest for both classes. Decomposing a data set into the three regions described as above, has the potential to offer useful insights of the data. Such insights may assist the analyst to better understand the phenomenon or systems related to the data. Clearly, this is domain dependent.

3.10 Improving Classification Accuracy

Classification algorithms are usually judged based on the accuracy of their predictions. If the predictions include a minimum number of false positives and false negatives, the accuracy of an algorithm is rated as high. During the experimental stage with various data sets, tests were conducted to see if critical nuggets could help improve the classification accuracy. The identified nuggets were used in deriving additional small scale classification models. For each class, an additional classification model is built/trained by first deriving a new data set which is a subset of the original training data set. The new data set was derived by relabeling a subset of the original data records into two new classes as follows:

- Data records that belong to the top k (for finding k, see lines 9-11 in Fig. 3.7) critical nuggets (of say, the '+' class) become a part of one class.
- Data records that are near the top k '+' class critical nuggets but NOT belonging to the set of '+' class critical nuggets are labelled as another class (this may include instances that belong to both '+' and '-' classes).

An additional model is built similarly using the critical nuggets from the other class (say, '-'). This is illustrated in Fig. 3.9. In this figure, the right region indicates critical nuggets belonging to the '+' class (labeled as dark shaded '+' symbols) and surrounded by some neighboring points belonging to both '+' and '-' classes.

The region on the top left illustrates critical nuggets belonging to the '-' class (labeled as dark shaded '-' symbols) and surrounded by both '+' and '-' neighbors. Using this newly derived data set, a classification model was setup. Since very few subsets qualify as critical nuggets, the newly re-labelled data set has a skewed distribution of classes. There are very few data instances that belong to the first class of critical nuggets. At the same time, there is a disproportionately large number of non-critical neighbors. This skewed distribution can be remedied using a cost sensitive classifier [12]. One can model a cost sensitive learner that assigns higher costs for misclassifying the class that is less represented when compared to misclassifying objects of the more represented class. In our case, it is more important to identify a critical nugget correctly. So the cost classification model is derived by assigning a higher cost for not identifying a critical nugget correctly. Using the newly trained classification models built around critical nuggets, one can use it in tandem with the original classification model to predict the class labels of data records. According to the experiments described next, it turns out that this method of post-processing of classified data records using the information gained from the critical nuggets helps in improving the classification accuracy in data sets. In summary, the steps in improving classification accuracy are outlined as follows:

1. Using a standard classification algorithm C, derive a classification model M_0.
2. For the first class (such as the '+' class in Fig. 3.9), build a data set comprising of two new classes:
    - One class is comprised of only the top k critical nuggets (e.g., dark shaded '+' points within the right circle in Fig. 3.9). Label this class as 'I'.

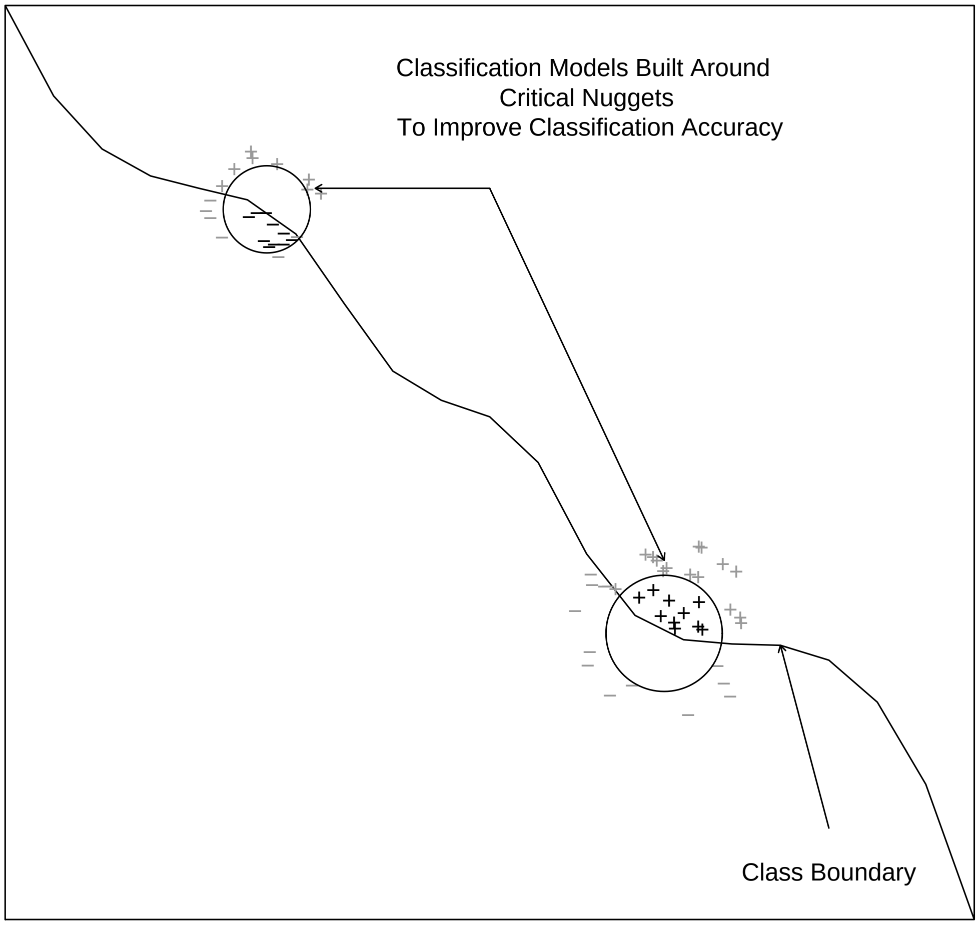

Classification Models Built Around
Critical Nuggets
To Improve Classification Accuracy
Class Boundary

FIGURE 3.9: Post-processing using Critical Nuggets For Improving Accuracy.

- The other class is comprised of non-critical records and it is derived by using the following principle: for each member in the set of critical nuggets (class ‘I’ in the previous step), choose the closest neighbors that are not part of class ‘I’ (e.g., points just outside the right circle in Fig. 3.9). Label these data instances as class ‘O’.

3. Using a nearest neighbor classifier, build a cost-sensitive classification model, assigning higher costs for misclassifying a record belonging to critical nuggets. The derived classification model is denoted as $M^{+}_{nuggets}$.

4. Repeat steps 2-3 for the second class (such as the ‘-’ class in Fig. 3.9) and the derived model is denoted as $M^{-}_{nuggets}$.

For a given test data instance or a new unlabeled data instance, the derived critical nuggets models ($M^{+}_{nuggets}$ and $M^{-}_{nuggets}$) are used along with the standard classification model M_0. We assign a class label guided by testing against the following set of rules:

- If $M^{+}_{nuggets}$ assigns a class label of ‘I’ and $M^{-}_{nuggets}$ assigns a class label of ‘O’, then the data instance is assigned a label of ‘+’.

- If $M^{-}_{nuggets}$ assigns a class label of ‘I’ and $M^{+}_{nuggets}$ assigns a class label of ‘O’, then the data instance is assigned a label of ‘-’.

- If $M^{+}_{nuggets}$ and $M^{+}_{nuggets}$ assign a class label of ‘O’, then the data instance is assigned a class label based on the class assigned by model M_0 obtained from the standard classification algorithm.

- If $M^{+}_{nuggets}$ and $M^{-}_{nuggets}$ assign a class label of ‘I’, then the data instance is assigned a class label based on the class assigned by M_0.

Chapter 4

Computational Analysis On Extraction Of Critical Nuggets And Accuracy Improvements

Some experiments were conducted on eleven multi-dimensional real world data sets from the UCI machine learning repository [17] and two 2-dimensional geographical synthetic data sets. The software was written in R [42] and Python [46] and utilized the data mining library Weka [22]. Associated software packages such as [27] were utilized as well. The algorithm $FindCriticalNuggets$ was applied to each of the data sets. The next two sections provide detailed summaries of the experimental analysis on one of the 2-dimensional synthetic data sets and two of the real world data sets. These summaries include details such as some results of the $FindCriticalNuggets$ algorithm, validation of properties such as duality, and the improvements in classification accuracy using critical nuggets. A similar analysis has been conducted for ten other real world data sets, but for the sake of space, detailed explanations have been provided only for two of the real world data sets. However, the classification accuracy improvements for all eleven real world data sets and two 2-dimensional synthetic data sets have been provided in the last subsection.

All the data sets used in the experiments were normalized (values for each attribute in a data set lie within $[0, 1]$) using normalization routines available in the Weka library. Euclidean distance measures were used in computing distances. For all the data sets, data instances were normalized and ten runs of 10-fold cross validations were performed. The following classification algorithms from Weka were used for this study: J48 (Weka's software implementation of the C4.5 [41] algorithm), IBk (Weka's implementation of K-nearest neighbor classifier [2], LMT

(Weka's implementation of Logistic Model Trees [32]) and NaiveBayes [28]. Default options in Weka were used for the algorithms J48, LMT and NaiveBayes and for the IBk algorithm, the nearest-neighbor parameter of K was set to 5. Table 4.1 provides a description of the data sets used during the experimental study.

TABLE 4.1. Description Of Data Sets Used.

Data Set	Number of Instances	Number of Attributes	Class (Distribution)
Synthetic Geographical (Georgia, USA)	10,387	2	+ (2,649), - (7,738)
Synthetic Geographical (Idaho, USA)	10,233	2	+ (2,963), - (7,270)
Wisconsin Breast Cancer (WDBC)	569	30	Benign (357), Malignant (212)
SPECT Heart	267	42	Normal (212), Abnormal (55)
Spambase	4,601	57	Spam(1,813), Not Spam(2,788)
German Credit Data	1,000	24	Good(700), Bad(300)
Pima Indian Diabetes	768	8	Positive(268), Negative(500)
Sonar	208	60	R(97), S(111)
Ionosphere	351	34	good(225), bad(126)
Cardiotocography2 [a]	1,950	22	Normal(1655), Suspect(295)
Liver Disorders	345	6	A(145), B(200)
Parkinsons [33]	195	22	H(48),P(147)
Glass2 [b]	163	9	Y(87), N(76)

[a]A variant of the Cardiotocography Data Set of UCI Repository, formed by considering data records belonging to only 2 out of 3 class attributes and ignoring the third class attribute called 'Pathologic'.

[b]A variant of the Glass Identification Data Set of UCI Repository, formed by combining data records belonging to class attributes 1 and 3 and renaming the class attribute as class 'Y', combining records having class attributes 2 and 4 and renaming as class 'N' and excluding records belonging to class attributes 5, 6 and 7. This variant has been used in prior work such as [32].

For the experiments, decisions had to be made with regard to the choice of a cost-sensitive classifier to handle the skewed data distribution (when building small classification models around critical nuggets) and the assignment of costs for the cost-sensitive classifier. For the improvement of accuracies using critical nuggets, two cost-sensitive algorithms (CostSensitiveClassifier [52] and MetaCost [12]) from Weka were considered to tackle the imbalanced data distribution. The algorithms, MetaCost and CostSensitiveClassifier, use a base classifier to instantiate the training process (the base classifier can be any standard classification algorithm). For the base classifier in MetaCost, the four classification methods used in the study (J48, LMT, IBk and NaiveBayes) were tested against the various data sets to see which classification method yielded the best accuracy. IBk, as a base classifier, consistently performed better than the others since the small models created around the critical nuggets were suited for neighborhood-based classification techniques. Hence IBk was chosen as the base classifier for MetaCost. MetaCost in combination with IBk consistently yielded better accuracies for different data sets when compared to the CostSensitiveClassifier method. So MetaCost in combination with IBk was chosen as the cost-sensitive classifier for the experiments. Cost ratios for cost-sensitive classifiers are ideally provided by domain experts. However, in this case, three different cost ratios 2:1, 5:1 and 10:1 were initially considered. The ratio of 5:1 (a cost of 5 was allocated towards misclassification of a critical nugget as compared to a cost of 1 allocated towards misclassification of a non-critical data instance) provided higher improvements in accuracy as compared to the ratio of 2:1. Increasing it to 10:1 did not provide any significant improvement when compared to accuracy improvements with the 5:1 ratio. The main goal among all the data sets was to correctly classify records that belong to critical nuggets. By assigning a higher cost (5) as a penalty for misclassification of critical nugget (as compared to equal cost penalties) was sufficient to meet the goal of improving classification accuracies for different data sets. Hence 5:1 was used as a cost ratio for the study.

4.1 Analyzing the Synthetic Geographical Data Set

This training data set is a synthetic data set that conforms to the political boundary of the State of Georgia, USA. This is a large data set that contains 10,387 observations comprising of 2 classes: '+' (2,649 instances) and '-' (7,738 instances). In other words, points inside the State of Georgia are defined as positive points, while the ones outside the political boundary are defined as negative points. A boundary set was approximated to 500 instances of class '+' and 500 instances of class '-'. This 2-dimensional data set has an advantage as it provides visual validation to the results. Figure 4.1 outlines the original data set.

Fig. 4.2 provides an approximation of the boundary set for the state of Georgia. Fig. 4.3 depicts the results of the $FindCriticalNuggets$ algorithm on the data set. Fig. 4.3 is similar to Fig. 4.2, except that some of the critical nuggets for both classes have been superimposed. Notice that in Fig. 4.3 the black dots indicate '+' data instances that have been identified as members of positive critical nuggets with a high CR_{score}. Also notice that these black dots line up along areas that have visually interesting features such as sharp bends and curves. Similarly, the dark grey dots indicate '-' data instances that have been identified as members of negative critical nuggets with a high CR_{score}.

Duality in this data set can be explained through the visual features. Areas 1, 2, 3 and 4 in Fig. 4.3 are visually interesting features. One can observe that critical nuggets for both classes have lined up near these visually interesting features. Also observe that critical nuggets for one class (black dots) tend to line up near the critical nuggets of the other class (dark grey dots).

Histograms of CR_{score} values for two different R values for each of the two classes are depicted as Fig. 4.4. Among the 500 different neighborhoods investi-

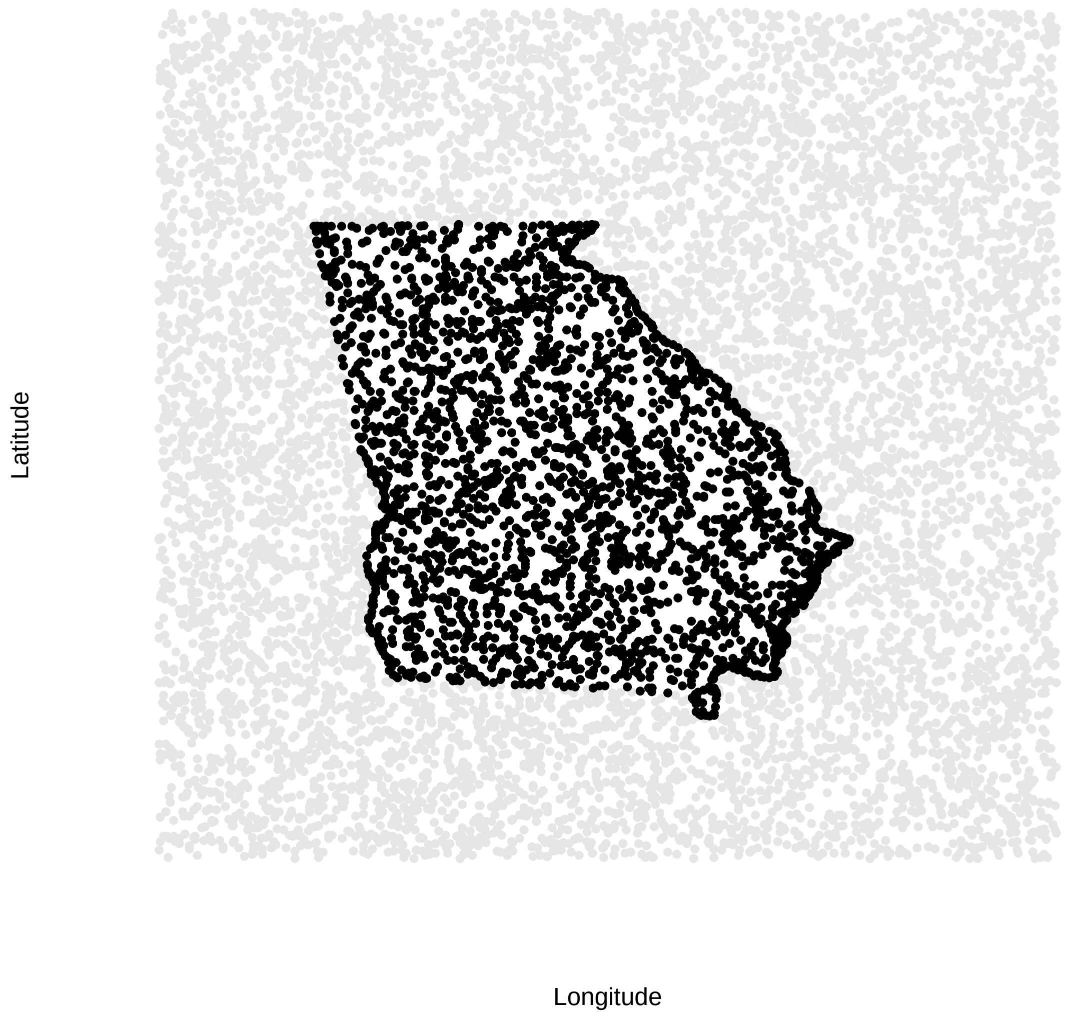


FIGURE 4.1: The Geographical Data set representing the state of Georgia, USA (black dots indicate points belonging to class '+' and Grey dots indicate points with class '-'.

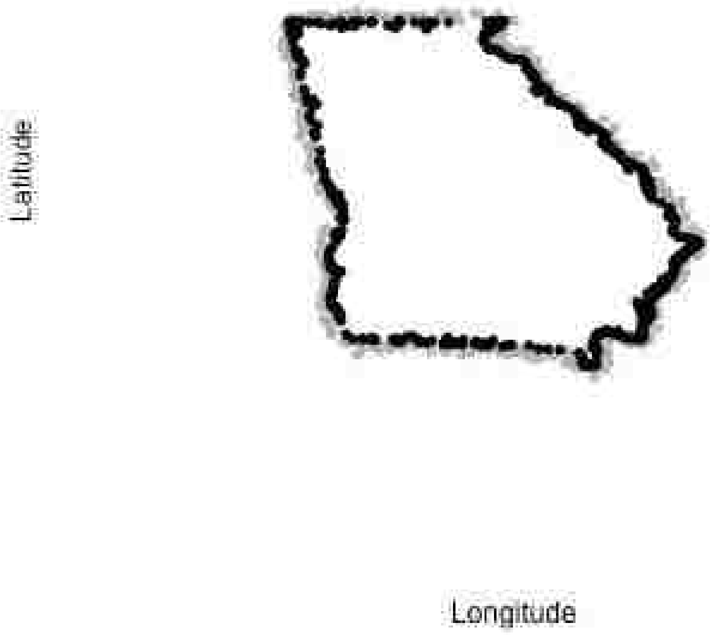


FIGURE 4.2: Boundary Approximation - Black dots indicate points belonging to class ‘+’ and Grey dots indicate points with class ‘-’.

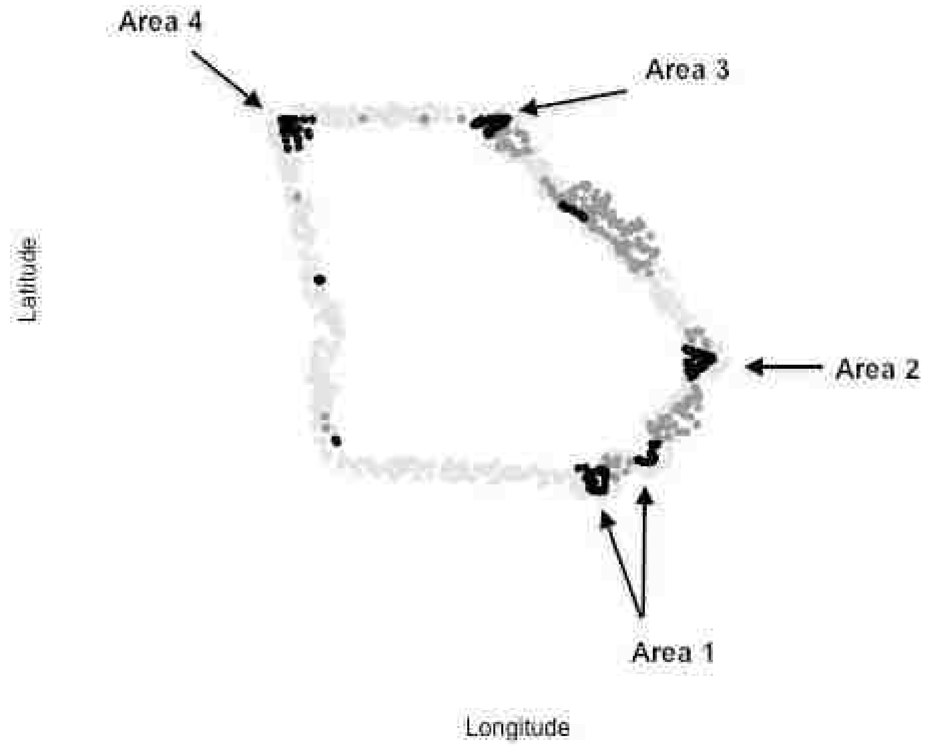


FIGURE 4.3: Critical Nuggets: Black dots indicate '+' points that belong to positive critical nuggets. Dark grey dots indicate '-' points that belong to negative critical nuggets. Light grey dots indicate '+' and '-' points that did NOT emerge as critical nuggets.

gated, it can be observed that there are indeed very few sets with high CR_{score} values, say greater than 0.75. This indicates the potential value of finding the critical nuggets in large data sets.

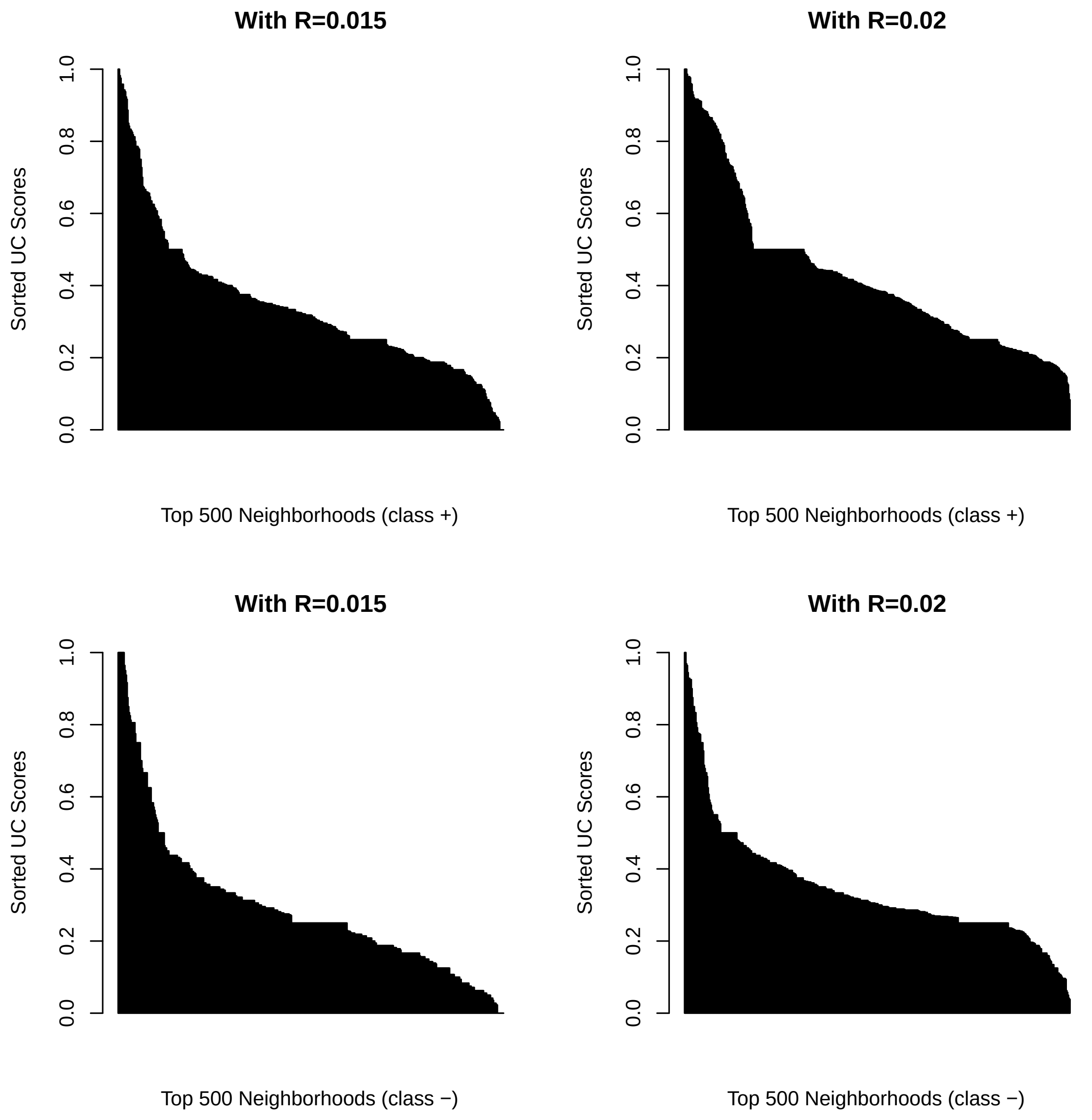


FIGURE 4.4: The Geographical Data set - Histograms of sorted scores for two different R values.

4.2 Analyzing the Wisconsin Breast Cancer (WDBC) Data Set

This data set has 569 data instances (357 Benign and 212 Malignant), 32 attributes (30 attributes when the record locator and class labels are skipped) and two types of class labels (Benign and Malignant). Using the $FindBoundary$ algorithm, an approximate boundary set comprising of 150 Benign and 150 Malignant data instances was selected. The main task was to apply the $FindCriticalNuggets$ algorithm to identify critical nuggets. The standard normalization function available in the Weka library was used to normalize this data set. For different values of R and for a given class, the $FindCriticalNuggets$ algorithm was run. For this analysis, five different R values were used: $\{0.20, 0.25, 0.30, 0.35, 0.40\}$. Increasing the range of R values beyond 0.40 increased the maximum size of the neighborhood to exceed the set limit of 100 neighbors (see Section 3.8). The neighborhoods that had the top 20 scores were identified for each value of R. Among the 100 ($= 20 \times 5$) neighborhoods, the top 20 neighborhoods were selected based on their scores and these neighborhoods were the critical nuggets.

The above steps of finding the top 20 critical nuggets was done for the other class as well. The top ranked critical nuggets for Benign and Malignant classes have been tabulated in Tables 4.2 and 4.3, respectively. A critical nugget is represented by the record numbers in the data set (e.g., Data instance #147 in the Breast Cancer data set is denoted as 147 in Table 4.2). Also, in each neighborhood represented in the table, the first instance is the point (center) around which a neighborhood was formed. This table includes for each critical nugget, the closest opposite class neighbors to the center, the most sensitive attribute and the R value that yielded the maximum score for that center.

Histograms of the CR_{score} values for two different values of R are outlined as Fig. 4.5. The histograms reveal that only a very small number of neighborhoods qualify as critical nuggets among the 150 different neighborhoods surveyed for each value of R. This underscores the potential importance of finding such sets.

The **duality** properties of critical nuggets can be studied by using Tables 4.2 and 4.3. The data instances that are surrounded by dark and transparent boxes are some of the examples to explain the duality properties of critical nuggets. As an example, note that the data instance 147 in Table 4.2 under the first column, "Critical Nuggets for Benign" also happens to be a neighbor of some of the critical nuggets of the malignant class in Table 4.3. Similarly, record 394 in Table 4.3 (surrounded by the transparent box) is a neighbor of some of the critical nuggets in the benign class (represented in Table 4.2). For this data set one finds that critical nuggets belonging to one class tend to be 'close' to critical nuggets belonging to the opposite class.

Above all, the methodology for finding critical nuggets has isolated benign records, whose attribute values when slightly perturbed, end up switching to the malignant class. This is valuable information as it can help identify benign records that are susceptible to switching over to the malignant class.

4.3 Analyzing the SPECTF Heart Data Set

This data set has 267 (212 Normal and 55 Abnormal) data instances with two class labels - Normal and Abnormal. The data set is defined on 44 attributes. A boundary set was generated comprising of 150 normal points and 30 abnormal points. The range of the R values used for the normal and abnormal classes was 0.50, 0.55, 0.60 and 0.65 (as per Section 3.8). **Histograms** of CR_{score} values for both normal and abnormal classes for different values of R are provided as Fig-

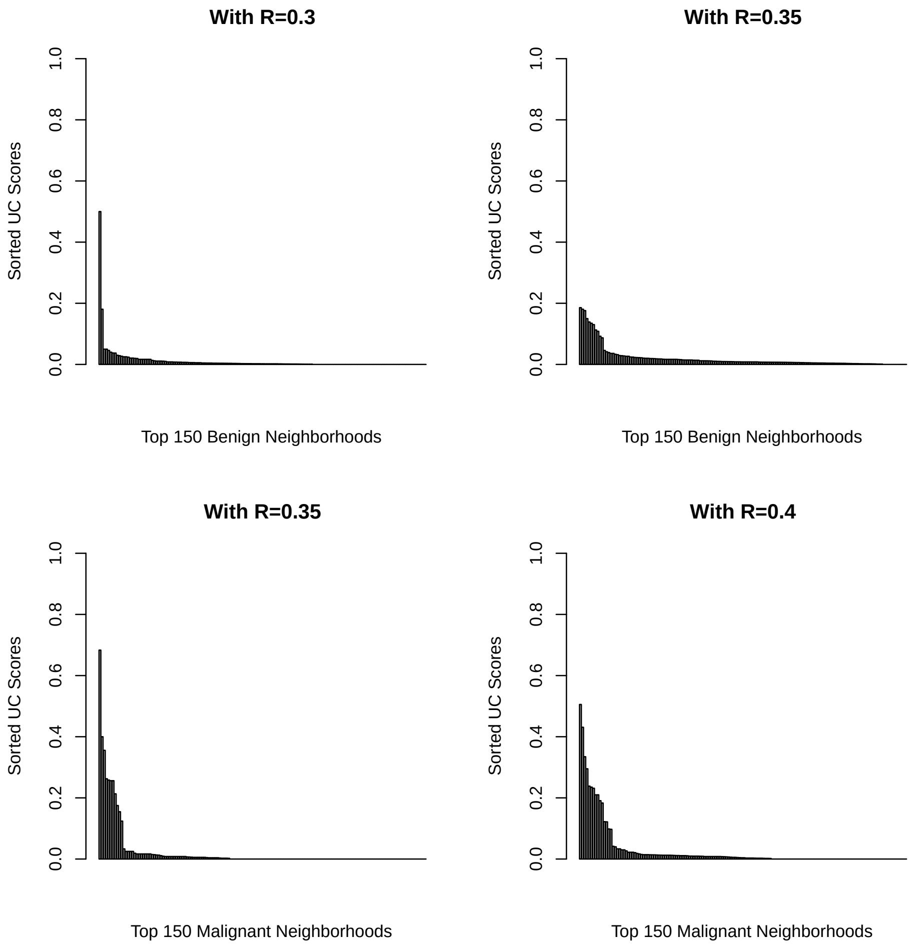


FIGURE 4.5: The WDBC Data set - Histograms of sorted scores for two different R values.

TABLE 4.2. Top results for the Benign Class in the WDBC Data Set.

Critical Nuggets for the Benign Class	Malignant Neighbors	CR_{score}	R	Attribute With Max Sensitivity
147	374, 402, 410, 422, $\boxed{447}$, 452, 453, 465, 486, 557	1.00	0.20	Mean Radius
288, 147, 239, 247, 265, 279, 353	374, $\boxed{394}$, $\boxed{402}$, 410, $\boxed{423}$, $\boxed{447}$, 452, 453, 527, 557	0.19	0.35	Worst Concave Points
239, 7, 147, 247, 288, 336	391, 394, 402, 422, 423, 435, 452, $\boxed{453}$, 557, 563	0.18	0.35	Worst Concave Points
247, 147, 184, 224, 279, 288	391, 398, 410, 419, 422, 453, 465, 486, $\boxed{557}$, 563	0.18	0.30	Worst Texture
224, 66, 97, 98, 147, 247, 321	$\boxed{374}$, 398, 410, 419, 422, $\boxed{447}$, 486, 538, $\boxed{557}$, 563	0.18	0.35	Worst Concave Points
111, 56, 66, 147, 184, 208, 211, 247, 284	384, 410, $\boxed{439}$, 445, $\boxed{453}$, 465, 486, 511, 540, 557	0.14	0.35	Worst Concave Points

TABLE 4.3. Top results for the Malignant Class in the WDBC Data Set.

Critical Nuggets for the Malignant Class	Benign Neighbors	CR_{score}	R	Attribute With Max Sensitivity
$\boxed{394}$	7, 30, 41, 153, 201, 269, 292, 299, 337	1.00	0.20	Mean Radius
431	7, 28, 97, 98, 112, 147, 205, 224, 236, 239	1.00	0.20	Mean Radius
437	7, 42, 97, 116, 202, 236, 251, 272, 299	1.00	0.20	Mean Radius
494	30, 67, 79, 153, 265, 275, 292, 297, 337	1.00	0.20	Mean Radius
503, 437	4, 33, 57, 61, 94, 157, 202, 232, 258	0.53	0.55	SE Texture
447, 368, 374, 394, 402, 423, 431, 437, 452, 494, 527, $\boxed{557}$	7, 32, 97, 108, 147, 203, 205, 265, 303	0.34	0.40	Mean Texture
549, 368, 394, $\boxed{453}$, 454, 466, 494	62, 79, 147, 153, 207, 233, 239, 264, 284	0.30	0.55	Mean Texture

ure 4.6. A total of 150 neighborhoods were investigated for the normal class and only a few neighborhoods had high CR_{score} values. For the abnormal class, only 30 neighborhoods were investigated as the data set has a very small number of data instances for this class and hence the boundary set is small as well. Hence, the histograms for the abnormal class do not show as much variation in CR_{score} values as the histograms for the normal class. An analysis similar to the one for the Wisconsin Breast Cancer Data was done to study **duality** and the results have been outlined in Tables 4.4 and 4.5. For the purposes of illustrating dual properties of critical nuggets, some of the data instances have been marked with boxes. This information could be highly useful in identifying patients that are at risk of becoming classified as ‘abnormal’.

TABLE 4.4. Top results for the Normal Class in the SPECTF Data Set.

Critical Nuggets for Class “Normal”	Neighbors in Class “Abnormal”	CR_{score}	R	Attribute With Max Sensitivity
131	215, $\boxed{216}$, 221, 225, 226, 227, 229, 230, 239, 241, 244, $\boxed{246}$, 247, 248, 259, $\boxed{262}$, 263, 264, 265, 267	1.00	0.50	F1R
35, 77	213, 214, 216, 220, 225, 227, 230, 233, 235, 241, 244, 248, 251, $\boxed{252}$, 253, 255, 258, 259, 263, 267	0.49	0.50	F1R
86, 18	213, 221, 224, 225, 226, 227, 230, 234, 235, 244, 248, 250, 252, 253, 255, 256, 259, 262, 263, 267	0.49	0.65	F1R
201, 16, 18, 68	215, 221, 226, 238, 240, 243, 244, 245, 248, 250, 252, 253, 255, 256, 258, 260, 261, 262, 263, 266	0.25	0.70	F8S
1, 18, 29, 68, 109	215, 221, 224, 226, 228, 234, 238, 240, 242, 243, 245, 249, 250, 252, 255, 256, 260, 262, 263, 266	0.21	0.65	F5S

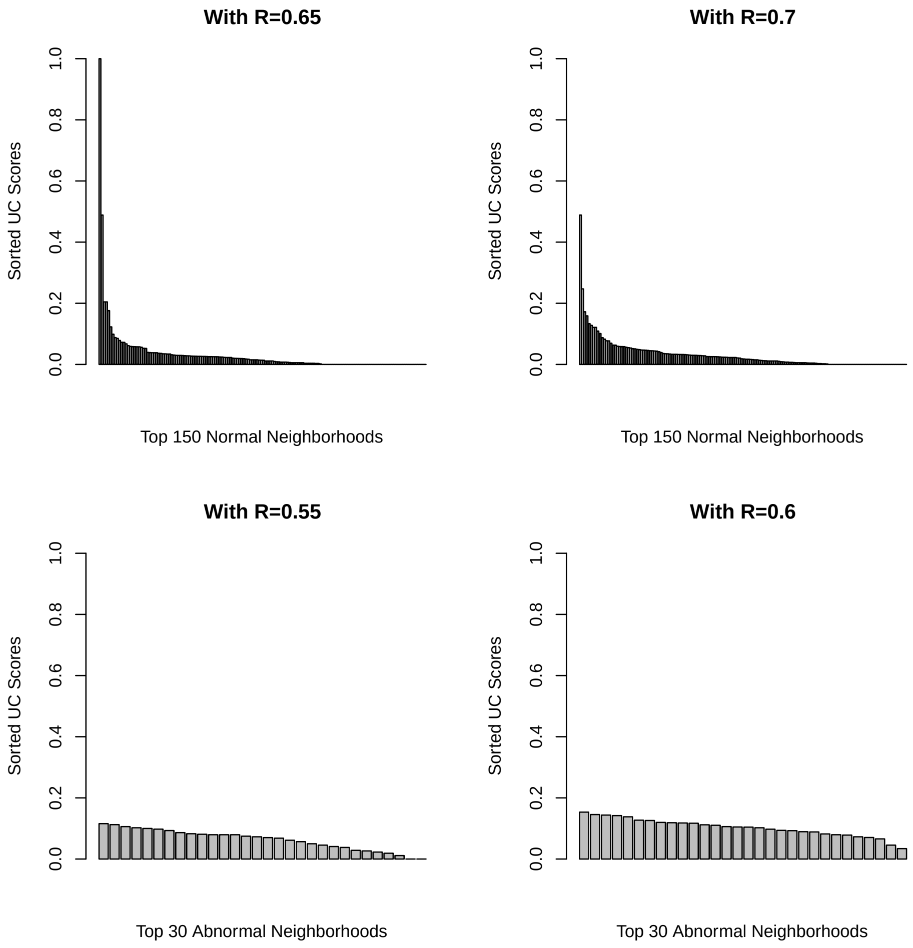


FIGURE 4.6: The SPECT Data Set - Histograms of sorted scores for two different R values.

TABLE 4.5. Top results for the Abnormal Class in the SPECTF Data Set.

Critical Nuggets for "Abnormal" Class	Neighbors in Class "Normal"	CR_{score}	R	Attribute With Max Sensitivity
$\boxed{223}$, 214, 216, 227, 228, 229, 232, 236, 237, 240, 244, 246, 252, 258, 262, 263, 267	2, 46, 69, 71, 95, 105, 135, 142, 149, 150, 157, 161, 168, 169, 173, 186, 190, 191, 197, 207	0.20	0.70	F2S
$\boxed{262}$, 216, 221, 223, 226, 229, 232, 236, 237, 238, 240, 242, 244, 245, 247, 250, 252, 255, 263, 266, 267	1, 2, 4, 6, 29, 69, 71, 72, 109, 142, 150, 155, 157, 161, 173, 175, 193, 197, 209, 210	0.18	0.70	F2S
$\boxed{252}$, 213, 214, 215, 216, 220, 221, 223, 226, 228, 232, 236, 237, 238, 240, 243, 244, 245, 248, 250, 253, 255, 256, 258, 261, 262, 263, 264, 266, 267	2, 7, 29, 44, 46, 57, 68, 70, 71, 109, 147, 156, 161, 163, 164, 175, 186, 191, 210, 211	0.17	0.70	F2S
$\boxed{246}$, 215, 216, 221, 223, 225, 226, 227, 229, 232, 237, 239, 240, 244, 247, 248, 253, 263, 264, 265, 267	2, 42, 69, 72, 82, 89, 95, 105, 115, 135, 142, 149, 150, 151, 168, 186, 190, 195, 197, 207	0.16	0.70	F2S
$\boxed{216}$, 214, 217, 219, 220, 221, 223, 225, 226, 227, 228, 229, 232, 233, 235, 237, 238, 239, 240, 241, 244, 246, 248, 250, 252, 253, 255, 258, 259, 261, 262, 263, 265, 266, 267	2, 7, 15, 20, 44, 46, 57, 70, 71, 77, 82, 89, 105, 142, 150, 156, 168, 186, 190, 207	0.16	0.70	F20S

4.4 Classification Accuracy Improvements Using Critical Nuggets

In Table 4.6, a summary of the classification accuracy improvements are provided. Eleven real world data sets from the UCI Repository [17] and two synthetic data sets were used. The column labelled as a_0 provides the accuracy of the chosen classification algorithm on a given data set, without the knowledge of critical nuggets. The next column labelled as a_1 indicates the improvements in accuracy of classification results as a result of knowledge gained through identification of critical nuggets.

In order to understand the values in the last column of Table 4.6 (labelled "Relative Improvement"), consider any row of the table. Suppose we consider the results for the SPECT Heart data set using 'J48' as the classification method. Without the use of the nuggets that accuracy was equal to 73.22%. With the use of critical nuggets the accuracy increased to 93.93%. The increase in accuracy is equal to $20.71\% = (93.93\% - 73.22\%)$. However, the maximum possible improvement would be equal to $100 - 73.22 = 26.78\%$. The previous increase of 20.71% represents $77.33\%(= \frac{20.71}{26.78})$ of the maximum possible improvement. The rest of the results in Table 4.6 have been computed in a similar manner. The exceptionally high values in the last column of Table 4.6 indicate the high potential critical nuggets may offer in improving classification accuracy.

Using the results of Table 4.6, the Wilcoxon [49] test was used to test the statistical significance of accuracy improvements using critical nuggets as compared to using a standard classification algorithm (without the knowledge of critical nuggets). For the comparison of two classifiers, the Wilcoxon Test has been recommended over other non-parametric and parametric tests by prior studies such as [10]. These results are summarized in Table 4.7. Notice that at 99% confidence level, the accuracy improvements using critical nuggets are statistically significant (p-values being less than 0.01) when compared to results obtained by using only the standard classification algorithms (J48, LMT, NaiveBayes and IBk).

TABLE 4.6. Improvements in Accuracy Using Critical Nuggets.

Data set	Classifier	a_0 (Accuracy before Critical Nuggets) (%)	a_1 (Accuracy With Critical Nuggets) (%)	Accuracy Increase $(a_1 - a_0)$ (%)	Relative Improvement $(\frac{a_1-a_0}{100-a_0})$ (%)
Synthetic Geographical (Georgia, USA)	J48	98.43	98.61	0.18	11.46
	LMT	99.49	99.91	0.42	10.19
	NaiveBayes	82.78	83.73	0.95	5.52
	IBk	98.29	98.51	0.22	12.87
Synthetic Geographical (Idaho, USA)	J48	98.83	98.93	0.10	8.55
	LMT	98.99	99.10	0.11	10.89
	NaiveBayes	80.02	81.33	1.31	6.56
	IBk	98.14	98.32	0.18	9.68
Wisconsin Breast Cancer (WDBC)	J48	94.22	98.05	3.83	66.26
	LMT	97.47	99.12	1.65	65.22
	NaiveBayes	93.34	96.00	2.66	39.94
	IBk	95.36	97.68	2.32	50.00
SPECT Heart	J48	73.22	93.93	20.71	77.33
	LMT	79.03	93.67	14.64	69.81
	NaiveBayes	68.13	89.36	21.23	66.61
	IBk	69.89	89.06	19.17	63.67
Spambase	J48	92.67	96.28	3.61	49.25
	LMT	93.33	96.60	3.27	49.03
	NaiveBayes	79.62	89.38	9.76	47.89
	IBk	90.85	95.41	4.56	49.84
German Credit	J48	73.13	79.76	6.63	24.67
	LMT	76.96	81.26	4.30	18.66
	NaiveBayes	75.42	82.50	7.08	28.80
	IBk	67.19	73.18	6.00	18.26
Pima Indian Diabetes	J48	75.00	84.69	9.70	38.78
	LMT	76.80	87.29	10.49	45.22
	NaiveBayes	75.74	86.44	10.70	44.11
	IBk	70.22	84.09	13.87	46.57
Sonar	J48	73.61	89.71	16.10	61.00
	LMT	77.07	90.05	12.98	56.61
	NaiveBayes	67.74	89.38	21.64	67.08
	IBk	86.44	95.43	8.99	66.30
Ionosphere	J48	89.68	98.40	8.72	84.50
	LMT	91.96	98.03	6.07	75.50
	NaiveBayes	82.65	94.73	12.08	69.63
	IBk	86.63	91.62	4.99	37.32
Cardiotocography2	J48	93.73	94.50	0.77	12.28
	LMT	94.66	95.61	0.95	17.79
	NaiveBayes	86.39	89.43	3.04	22.34
	IBk	92.72	93.68	0.96	13.19
Liver Disorders	J48	65.68	85.19	19.51	56.85
	LMT	69.71	85.22	15.51	51.20
	NaiveBayes	55.94	77.36	21.42	48.62
	IBk	63.01	82.14	19.13	51.72
Parkinsons [33]	J48	83.79	95.90	12.11	74.71
	LMT	84.56	97.08	12.52	81.09
	NaiveBayes	69.38	94.36	24.98	81.58
	IBk	95.79	98.67	2.88	68.41
Glass2	J48	79.14	96.32	17.18	82.36
	LMT	77.48	96.07	18.59	82.55
	NaiveBayes	61.96	94.36	32.40	85.17
	IBk	77.42	96.75	19.33	85.61

TABLE 4.7. Significance of Improvements

Comparison	Positive Ranks	Negative Ranks	p-value
J48 vs. J48+Critical Nuggets	0	91	< 0.01
LMT vs. LMT+Critical Nuggets	0	91	< 0.01
NaiveBayes vs. NaiveBayes+Critical Nuggets	0	91	< 0.01
IBk vs. IBk+Critical Nuggets	0	91	< 0.01

Chapter 5
Minimizing Misclassification Costs Using Critical Nuggets

Consider the illustration in Figure 5.1. The illustration depicts a common, generic decision making scenario when predicting class labels using critical nuggets. The positive and negative critical nuggets models are represented as circles in the illustration. This is a two-class (classes '+' and '-') data set, the classes being separated by a class boundary (represented as a jagged line in the illustration). The regions representing false positive and false negative are indicated as well. However, if the positive and negative critical nuggets models intersect, then data records that lie within the common area of intersection can be difficult to classify. In the previous chapter, during the improvement of classification accuracies using critical nuggets, class labels that were within this uncertain region were classified using the base classifier. In the illustration, this region of uncertainty is shaded in grey. In this chapter, the problem of minimizing misclassification costs is handled using 2 different approaches. In the first approach, the uncertainty is resolved by considering a case called the undecided case. This entails redefining the definition of misclassification cost TC to account for the undecided case. In the latter approach, the uncertainty is resolved by allowing the base or original classifier to make the decision. In the latter approach, the standard misclassification cost TC_0 is not changed and does not account for any additional costs other than false positive and false negative costs.

5.1 Misclassification Cost, TC_0

In classification tasks, accuracy of an algorithm is based on the number of correct predictions made by the classification algorithm. Implicitly, this method factors in

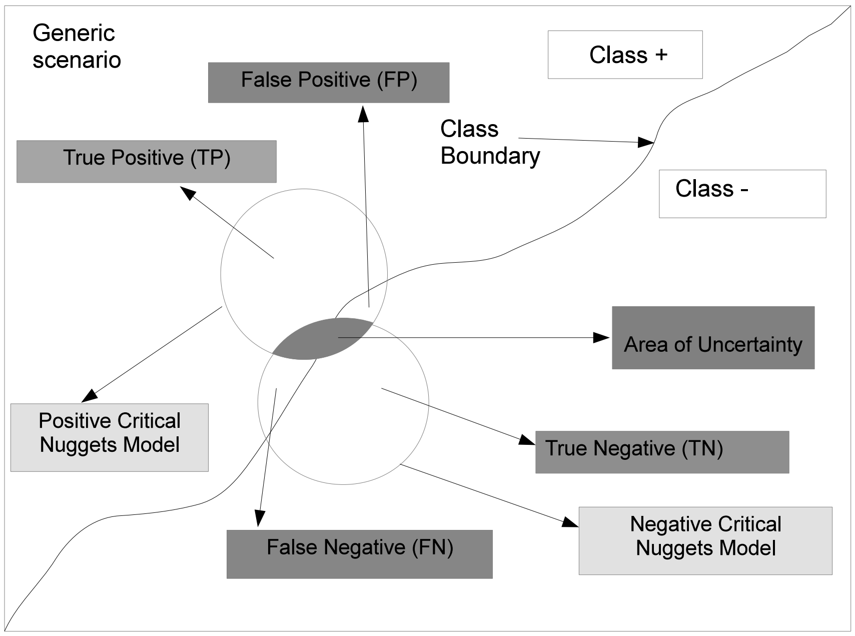


FIGURE 5.1: Illustrating the Minimize TC problem.

only true positive and true negative classifications. Based on the above definition, misclassification cost, TC_0, is defined as:

$$TC_0 = c_{FP} \times FP + c_{FN} \times FN \tag{5.1}$$

where c_{FP} is the cost of committing a false positive error and c_{FN} is the cost of committing a false negative error. FP and FN are the false positive and false negative rates respectively. A false positive rate, FP, is defined as the percentage of false positive errors when compared with the total number of data records being tested and the false negative rate, FN, is defined as the percentage of false negative errors when compared with the total number of data records being tested. In some cases, the misclassification cost TC_0 is computed by assigning the costs c_{FP} and c_{FN} as 1 (i.e., they are weighted equally).

5.2 Redefining TC_0 As TC

Some of the problems with the above approach are:

- This measure treats the false positive and the false negative errors equally. In certain cases, especially in disease diagnosis, it is more serious not to have one type of error than the other type of error. For example, in cancer diagnosis, it is more important not to make an error in classifying a malignant case as a benign case. To handle this scenario, the measure can be redefined with added penalties or weights assigned to handle the type of error one would like to avoid. In other words, c_{FP} and c_{FN} are not weighted equally and are assigned values based on the gravity of the type of error.

- In certain cases, a classification algorithm may not be able to decide on what prediction to make. In other words, the degree of certainty associated with a prediction is low. During the prediction of a class, especially during a medical diagnosis, it may be better to conclude a prediction as undecided instead of committing a false positive or false negative error. In an undecided situation, the accuracy measure can be redefined in such a way that it handles not only false positive and false negative errors but also the cost of undecided predictions.

The above drawbacks were first considered and discussed extensively in [45], [40] and [39]. To overcome, the above drawbacks, the work by [40] also suggested redefining TC as

$$TC = c_{FP} \times FP + c_{FN} \times FN + c_{UC} \times UC \tag{5.2}$$

where c_{UC} is the cost of the undecided or unclassifiable case and UC is the rate of unclassifiable cases (basically the percentage of unclassifiable cases when compared to the total number of test data instances). The new definition takes into account the unclassifiable cases.

In the light of the previous section on improving classification accuracy and to incorporate the unclassifiable case, one will have to revise how data records are assigned class labels using critical nuggets. Recall from the previous section that two models were created for each of the two classes using critical nuggets. When a new data point has to be assigned a class label, the following revised set of rules are applied.

1. If a data record is classified as a critical nugget (or I) by the positive critical nugget model and non-critical by the negative critical nugget model (or O), then the data record is classified as positive.

2. If a data record is classified as a critical nugget (or I) by the negative critical nugget model and non-critical by the positive critical nugget model (or O), then the data record is classified as negative.

3. If a data record is classified as a critical nugget (or I) by both the positive and negative critical nugget models, then the point is labelled as unclassifiable or labelled as U.

4. If a data record cannot be classified as a critical nugget (or O) by both the positive and negative critical nugget models, then instead of relying on the original base classifier (as in the previous study), assign the class label as 'unclassifiable' or U. This will account for the unclassifiable class and one can use this to calculate the revised TC score.

Comparing the revised rules above with those provided in Section 3.10, only the last rule is different. In Section 3.10, one relied on classification provided by the original base classifier to handle conflicts between classifications provided by the positive and negative critical nuggets model. In the revised rules, the classification is marked out as unclassifiable.

5.3 Using Critical Nuggets To Minimize TC

The computational experiments with critical nuggets (outlined in earlier chapters) have showed that critical nuggets tend to lie close to the class boundary separating the two classes in a data set. Areas near the class boundary also tend to be the places where the unclassifiable type is more likely to occur. These areas also tend to have the most likelihood of encountering false positive and false negative errors. Given that one now has a set of positive and negative critical nuggets, can their sizes be changed or adjusted so as to lead to a minimum misclassification cost TC? Recall from Chapter 3 that positive critical nuggets were built by growing a subset of positive data records that were within a distance R from a center p_0. For minimization of TC, the critical nuggets for a particular class (say, positive) are shrunk/grown by a certain scale factor. Since a critical nugget is built around a point say p_0, a scale factor of 0.5 would essentially yield a critical nugget that is built of data records that lie within $0.5 \times R$ from p_0.

The shaded grey region in the center of Figure 5.1 is the undecided region where the positive and negative critical models conflict. The point of the illustration is that by varying the sizes of the positive and negative critical nuggets models, one can alter the sizes of the problematic regions (false positive, false negative and undecided cases), thereby helping one to adjust the cost TC. The ideal scenario is a scenario where there are no false positive, false negative and undecided regions. However, such a scenario is also dependent on the data set and may not be realistic while considering real world data sets.

The knowledge of critical nuggets was used to conduct a series of experiments and find optimal sizes of critical nuggets that help minimizing the total misclassification cost TC. Given an a priori list of costs, c_{FP}, c_{FN} and c_{UC}, the size of critical nuggets (obtained using the $FindCriticalNuggets$ algorithm) can be varied (increased or decreased proportionately). For each variation of size of critical nuggets, the total cost TC is computed and stored.

The size of the critical nuggets can be varied using the following steps.

- For each of the positive critical nuggets, identify the center c_0 and the value of R used initially to build the neighborhood of positive points that formed the critical nuggets.

- Use c_0 to grow a neighborhood N by identifying all points (including positive and negative instances) that lie within distance $R \times p_1$ (where p_1 is a scale factor for positive critical nuggets and p_2 is a scale factor for negative critical nuggets).

- Use the newly identified neighborhood N in building the positive critical nuggets model, $M^{+}_{nuggets}$.

- Repeat the same procedure for the negative critical nuggets by replacing p_1 with p_2 and $M^{+}_{nuggets}$ with $M^{-}_{nuggets}$.

Given the above background on how to shrink and grow the critical nuggets, two search-based approaches are outlined that can help minimize TC.

5.4 Approach 1 - A Candidate Set Based Search (CNCS)

The first approach was to use a candidate set of critical nuggets sizes or scale factors (recall that critical nuggets were derived using the size parameter R). Evaluating every scale factor in the candidate set one can calculate the minimization cost, TC for different sizes of critical nuggets. After all the TC values are evaluated for the entire candidate set, one can identify the minimum TC cost. The main idea in finding the minimum TC score was the following: by expanding and shrinking the positive and negative critical nuggets, the models built around them ($M^{+}_{nuggets}$ and $M^{-}_{nuggets}$) change. This results in changing the false positive rates, false negative rates and the undecided case rates ultimately impacting the value of TC. By adopting a nested loop approach, one can shrink and expand the positive and negative critical nuggets, evaluate TC value for each iteration and then find the minimum TC value. One also finds a good balance between over-generalization and over-fitting of the data sets in question. After a range of sizes are considered for both positive and critical nuggets and TC values are evaluated at each iteration, two plots were constructed. One plot depicts total cost TC vs. Sizes of Positive Critical Nuggets (keeping the sizes of negative critical nuggets constant) and the other depicts total cost TC vs. Size of Negative Critical Nuggets (keeping the sizes of the positive critical nuggets constant). From either one of the plots, a minimum score can be visually identified by finding the point with the smallest score of TC.

The algorithm is outlined in Figure 5.2.

The algorithm $CNCS$ loops through various sizes of positive and negative critical nuggets. Elaborating on this, recall that each positive critical nugget comprises of one or more positive points and each negative critical nugget comprises of one or more negative points. Also, recall that the $FindCriticalNuggets$ algorithm yields a group of such positive and negative critical nuggets. Assuming that the size of each critical nugget (positive and negative) obtained from the $FindCriticalNuggets$ algorithm is an arbitrary unit of 1, one can shrink and expand each nugget by a proportional amount (say a 25% decrease would mean that the arbitrary unit of 1

```
Require: $c_{FP}$, $c_{FN}$, $c_{UC}$: the costs for false positive, false negative and undecided cases.
Require: : Critical Nuggets for classes '+' and '-' and data set, $T_r$.
1: $TCArray=\phi$
2: From the range of $R$ values used in calculating critical nuggets, find candidate set
   of size ratios, $CS_1$ and $CS_2$ to grow and shrink the respective positive and negative
   critical nuggets.
3: for each ratio $p_1$ in $CS_1$ do
4:     Proportionally change size of all Positive Critical Nuggets using ratio $p_1$.
5:     Generate Positive Critical Nuggets Model, $M^+_{nuggets}$, using new Positive Critical
       Nuggets.
6:     for each ratio $p_2$ in $CS_2$ do
7:         Proportionally change size of all Negative Critical Nuggets using ratio $p_2$.
8:         Generate Negative Critical Nuggets Model, $M^-_{nuggets}$.
9:         Use $M^+_{nuggets}$, $M^-_{nuggets}$ and $M_0$ to predict class labels for $T_r$.
10:        $FP$=Number of False Positives/$|T_r|$
11:        $FN$=Number of False Negatives/$|T_r|$
12:        $UC$ = Number of Undecided Cases/$|T_r|$
13:        $TC = (c_{FP}$ x $FP + c_{FN}$ x $FN + c_{UC}$ x $UC)$ x 100
14:        Append $TC$ to $TCArray$
15:    end for
16: end for
17: Find minimum from $TCArray$.
```

FIGURE 5.2: The $CNCS$ Algorithm.

becomes 0.75 or in other words, each critical nugget is shrunk to 75% of its original size, losing some of the members of the original set). In the algorithm $CNCS$, various levels of the proportional ratio p_1 are used to vary the size of the positive critical nuggets and various levels of the proportional ratio p_2 are used to vary the size of the negative critical nuggets. Once a critical nugget has been expanded and shrunk, two critical nuggets models (as in the previous section, $M^+_{nuggets}$ and $M^-_{nuggets}$) are derived by relabeling the data instances as I and O. Using the derived models, $M^+_{nuggets}$, $M^-_{nuggets}$ and M_0 (derived from the original classifier), one can predict the class labels of test data instances, using the rules derived in the previous sub-section. A 10-fold cross validation procedure was used for testing of data instances. For each iteration of p_1 and p_2, the total classification cost is computed and stored. After all the iterations are completed, the minimum TC is found from the stored list of classification costs.

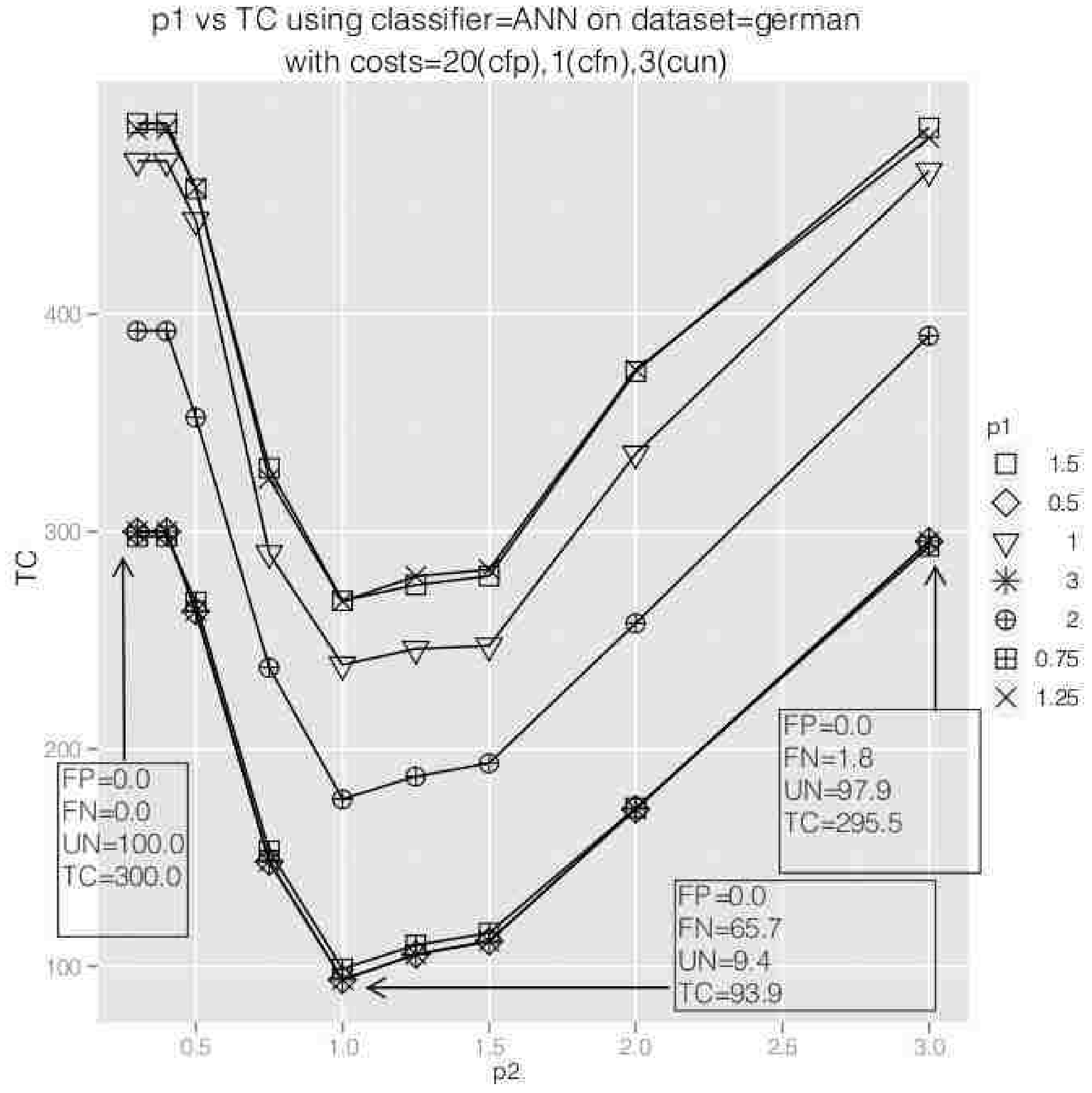


FIGURE 5.3: A graphical illustration of minimization of TC

An explanation of this methodology can be explained graphically using Figure 5.3. The graph depicts the proportion parameter p_1 (used to grow and shrink the sizes of positive critical nuggets) on x-axis and the classification cost TC on the y-axis. Each of the line graphs indicate different levels of proportion p_2 (used to grow and shrink the sizes of negative critical nuggets). For this illustration, the line depicting p_2 value of 0.5 is used and three points (each representing p_1 values of 0.3, 1.0 and 3.0 respectively) are considered (indicated by black arrows). The algorithm $CNCS$ was used on the German Credit data set using the classifier ANN(Artificial Neural Networks) in conjunction with critical nuggets. The cost parameters used were $20(c_{FP})$, $1(c_{FN})$ and $3(c_{UC})$. Notice in Figure 5.3, that for a constant level of $p_2 = 0.5$, as one varies p_1, the TC values decrease (dropping from an initial value of $TC = 300.0$) and hit a minimum value at $p_1 = 1.0$ ($TC = 93.9$). As one continues to increase p_1, then TC values increase again to a value of $TC = 295.5$. Also it can be observed that initially the undecided case rate (UC) is high and and it starts falling until one hits the minimum value of TC. UC then starts increasing as we continue to increase the value of p_1. Though the false positive rate (FP) does not change for different values of p_1, the false negative rate initially increases and after reaching a maximum at $p_1 = 1.0$, FN starts decreasing again. Notice the balance between the cost parameters and their respective rates. The cost parameters c_{FP} is significantly higher than the other parameters and hence the false positive rate FP remains steady. Also c_{UC} is higher than c_{FN} and hence the undecided rate UC drops for the minimum value of TC. A good balance between over-generalization and over-fitting is achieved as a result of using the $CNCS$ algorithm. A detailed set of experiments for various cost scenarios using different data sets and different classification algorithms is provided in the next chapter.

The complexity of CNCS can be derived as follows. In each loop of the nested loop algorithm, a positive and negative critical nuggets model is created and queried. Assuming the time to create the positive and negative critical nuggets models is $t(C)$ and the time to query is n, the total time is $t(C) + n$. Since this is a nested-loop algorithm, and since we loop through a candidate set, say c, the overall time complexity is $O(|c|^2(t(C)+n))$ (also note that $|c| << n$).

5.5 Approach 2 - A Genetic Algorithm (GA) Based Search, $CNGA$

While the prior approach utilized a candidate set to reduce the search space, the second approach uses a genetic algorithm (GA) based approach ([21] and [26]) to search for the optimal set of parameters that yield the minimum value of total classification cost, TC. This type of search technique has been used in prior misclassification cost studies such as [37], [48] and [40]. The genetic algorithm based approach considers a randomized approach in finding the optimal set of parameters. The fitness function considered for this methodology was:

$$TC = c_{FP} \times FP + c_{FN} \times FN + c_{UC} \times UC \tag{5.3}$$

The parameters used (chromosomes in GA-based terminology) were the expansion factors (α^+, α^-) for expanding the positive and negative critical nuggets respectively and contraction factors (β^+, β^-) for reducing the the positive and negative critical nuggets respectively. The chosen parameters are consistent with the approaches used in identifying critical nuggets. The chosen parameters were also used in similar prior studies such as [39] and [40]. For the purposes of this study an initial population size of 50 was considered. The number of evolutionary generations considered was 50. For the purposes of mutation in the GA-based approach, a gaussian distribution was used. The ranges for α^+, α^-, β^+, β^- were selected as appropriate for each of the data sets. The chromosomes were varied using appropriate crossover and mutation rates for each of the data sets. The variation of the genomes (chromosomes) was then used to evaluate the cost TC at each step. The evolutionary steps stop when there was no further improvement in the TC cost after successive mutations and crossovers of the genetic sequence.

The algorithm is outlined in Figure 5.4.

```
Require: $c_{FP}, c_{FN}, c_{UC}$: the costs for false positive, false negative and undecided errors.
Require: : Critical Nuggets for classes '+' and '-' and data set, $T_r$.
1: $TCArray = \phi$
2: Using GA initialize genome parameters - expansion parameter, $\alpha$ and contraction
   parameter, $\beta$
3:
4: while $TC$ not converging do
5:     Proportionally change size of all Positive Critical Nuggets using ratio $\alpha$.
6:     Generate Positive Critical Nuggets Model, $M^{+}_{nuggets}$, using new Positive Critical
       Nuggets.
7:     Proportionally change size of all Negative Critical Nuggets using ratio $\beta$.
8:     Generate Negative Critical Nuggets Model, $M^{-}_{nuggets}$.
9:     Use $M^{+}_{nuggets}$, $M^{-}_{nuggets}$ and $M_0$ to predict class labels for $T_r$.
10:    $FP$=Number of False Positives/$|T_r|$
11:    $FN$=Number of False Negatives/$|T_r|$
12:    $UC$ = Number of Undecided Cases/$|T_r|$
13:    $TC = (c_{FP}$ x $FP + c_{FN}$ x $FN + c_{UC}$ x $UC)$ x 100
14: end while
15: Minimum $TC$ is reached when there is no further improvement in $TC$.
```

FIGURE 5.4: The *CNGA* Algorithm.

The *CNGA* algorithm was coded in Python and used the genetic algorithms library, Pyevolve ([38]). The CNGA algorithm was applied on various data sets using different cost scenarios. The computational analysis using CNGA is provided in the next chapter. A time comparison analysis between CNGA and CNCS has been provided as well.

Chapter 6

Computational Analysis On Minimizing Misclassification Costs Using Critical Nuggets

A number of computational experiments were conducted to analyze the different approaches that had been proposed in the previous chapter. An overview of the the entire computational analysis mentioned in this chapter is outlined as an illustration in Figure 6.1

Given an a priori list of cost parameters (c_{FP}, c_{FN}, c_{UN}), extensive computational experiments were performed on the datasets to test the methodology of minimization of classification costs using critical nuggets. As described earlier, the algorithm outlined in Figure 5.2 was used to find the minimum TC for various data sets when critical nuggets were used in conjunction with a standard classification algorithm. For the following set of experiments, the classification algorithms used in conjunction with critical nuggets were J48, SVM (Support Vector Machines) and ANN (Artificial Neural Networks). These algorithms were chosen as they have been used in prior similar studies such as [39] and [40]. In [39], a heuristic called Homogeneity-Based Algorithm (or HBA) was introduced wherein a genetic algorithm based approach was used to find a minimum value of TC. The method was also computationally slow and took enormous time to complete a run. In [40], a remedial approach to HBA was proposed, called the Convexity Based Algorithm (CBA). The CBA proved to be computationally more efficient than the HBA, but the misclassification costs (TC) obtained using CBA were inferior (or greater in value) than those obtained by the HBA.

In this work, two approaches are outlined that seek to minimize TC values. One is a genetic algorithm based search procedure called CNGA and the other is a

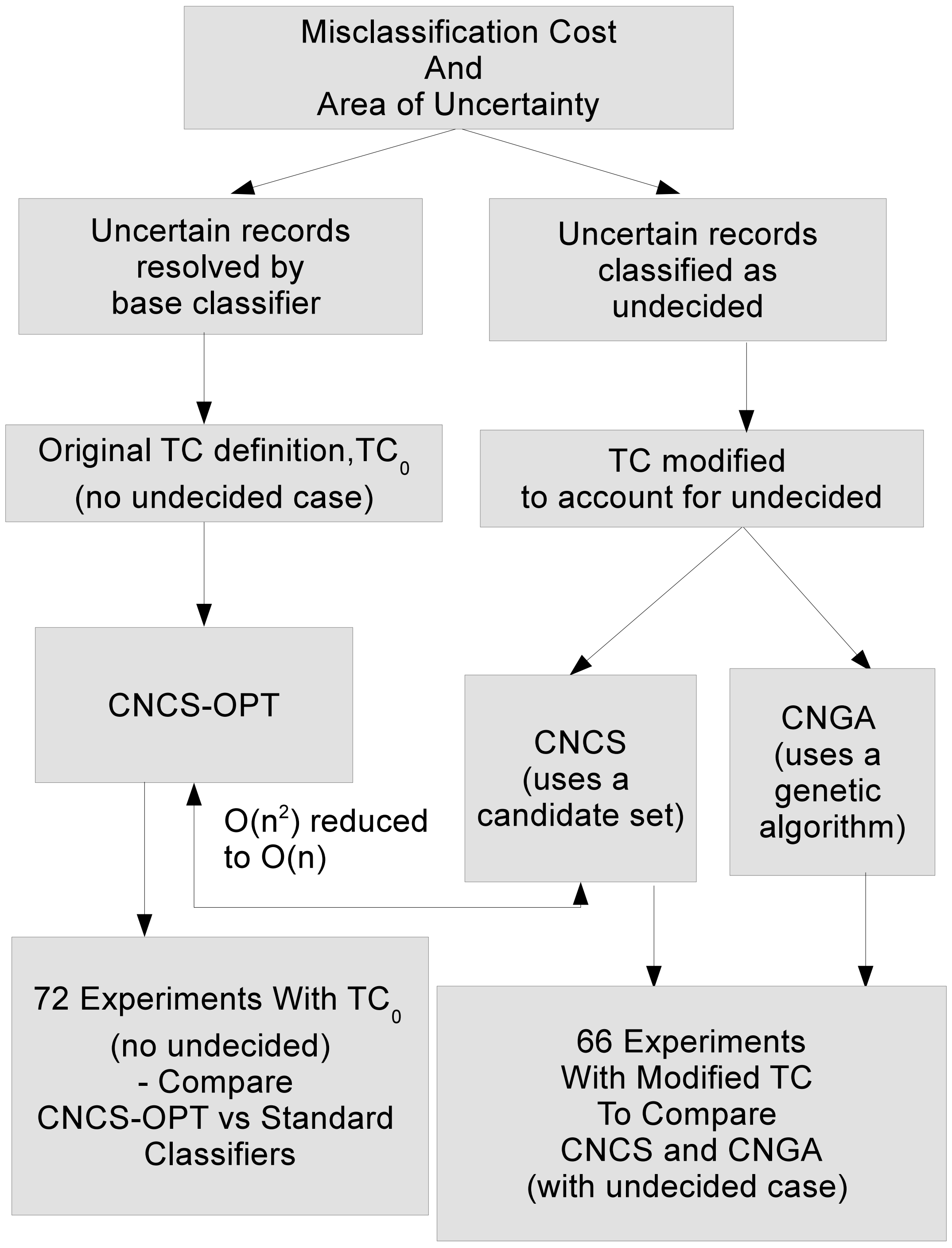

Misclassification Cost And Area of Uncertainty
Uncertain records resolved by base classifier
Uncertain records classified as undecided
Original TC definition,TC$_0$ (no undecided case)
TC modified to account for undecided
CNCS-OPT
CNCS (uses a candidate set)
CNGA (uses a genetic algorithm)
O(n^2) reduced to O(n)
72 Experiments With TC$_0$ (no undecided) - Compare CNCS-OPT vs Standard Classifiers
66 Experiments With Modified TC To Compare CNCS and CNGA (with undecided case)

FIGURE 6.1: An overview of the computational analysis for minimization of TC

candidate set based search called CNCS. 4 cost functions were used to analyze how critical nuggets help in minimizing the misclassification cost and to compare the relative performance of CNGA and CNCS. Each of the cost functions are now analyzed to demonstrate the utility of critical nuggets. The cost functions used in this study were also used in other prior studies such as [39] and [40]. All the data sets used in this experimental study form a subset of the datasets listed in Table 4.1. A description of the data sets used for this study is provided as Table 6.1. The software implementation for the classifiers (J48, SVM and ANN) used in [39] and [40] differs from that used in this study (alternatively, this study used the Weka data mining software library [22], software libraries such as RWeka [27] and programming languages, R and Python). The implementation aspects of the computational experiments (cross-validations and parameters used within the classification algorithms) differs from that used in prior studies such as [39] and [40]. Hence the results obtained in prior studies (such as [39] and [40]) cannot be directly compared with this study. This study uses a 10-fold cross validation approach. For J48 and ANN, the default options within Weka ([22]) have been used. For SVM, the options of $K = 3$ and $G = 2$ were set (parameters for radial basis function as the type of kernel and the gamma parameter for the kernel function respectively).

6.1 Comparing CNCS and CNGA

6.1.1 Using a Geographical Data Set

In the first set of experiments, a synthetic geographical data set, representing the outline of the political boundary of the state of Georgia was used. Three different cost ratios were used (20-1-3, 1-20-3, 3-1-20) and CNCS and CNGA was run for each of the cost functions. As an example, a cost ratio of 20-1-3 indicates a penalty of 20 for incurring a false positive error, a penalty of 1 for incurring a false negative

TABLE 6.1. Description Of Data Sets Used.

Data Set	Number of Instances	Number of Attributes	Class (Distribution)
Liver Disorders	345	6	A(145), B(200)
Ionosphere	351	34	good(225), bad(126)
Pima Indian Diabetes	768	8	Positive(268), Negative(500)
German Credit Data	1,000	24	Good(700), Bad(300)
Synthetic Geographical (Georgia, USA)	10,387	2	+ (2,649),- (7,738)
Cardiotocography2 [1]	1,950	22	Normal(1655),Suspect(295)

error and a penalty of 3 for the undecided case. The decision tree classifier, J48, was used as a base classifier. The results have been tabulated in Table 6.2. From Table 6.2 one can see that CNCS and CNGA perform on par with one another on two of the cost ratios (1-20-3 and 20-1-3). However, when the penalty for undecided case is increased to 20, CNCS performs poorly ($TC = 326.4$) when compared to CNGA ($TC = 204.1$).

TABLE 6.2. Results For Geographical Data Set - GA

	Using J48 as a classifier				
Cost Function	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
1-20-3	CNCS	10.4	1.7	19.8	104.4
	CNGA	7.1	2.9	11.5	99.4
20-1-3	CNCS	0.0	12.8	32.7	111.5
	CNGA	3.3	10.3	13.9	117.7
3-1-20	CNCS	12.1	4.9	14.3	326.4
	CNGA	14.2	5.9	7.8	204.1

6.1.2 Using Cost Function $TC = min(1 \times FP + 20 \times FN + 3 \times UC)$

In certain classification tasks, it is more important not to commit one type of error (false positive or false negative) over another. In this cost function, the cost of false negative rate is weighted heavily (= 20) when compared to false positive errors. In certain domains such as medicine, a less serious medical condition maybe wrongly misclassified as a very serious condition, leading to unnecessary psychological trauma for the patient in question. This cost function accounts for that by assigning a heavy penalty for committing a false negative error (where the negative class maybe compared to a serious condition and the positive class maybe compared to a benign or less serious condition) when compared to a false positive error. For the above set of cost parameters, the results are tabulated in Table 6.3.

Using this cost function, 18 computational runs (9 for CNGA and 9 for CNCS) were conducted. Medical data sets such as Cardiotocography2 were used for this study. CNCS and CNGA perform on par with one another in most cases except for some exceptions (such as the result for Cardiotocography with J48 where CNCS outperforms CNGA). However, note that the false negative rate is lower than both the false positive and undecided rates for all the experiments. Since the penalty for committing the false negative rate was the highest, the algorithms CNCS and CNGA have achieved their goals of reducing the rate (in this case false negative) that has the highest penalty. Figure 6.2 illustrates the analysis of this cost function on the Cardiotocography2 data set using the CNCS methodology.

6.1.3 Using Cost Function $TC = min(1 \times FP + 100 \times FN + 3 \times UC)$

In this case, the cost parameters are changed from the previous section by increasing the cost of false negative rate from 20 to 100. This cost function penalizes the false negative rate to a greater extent. The results have been provided in Table 6.4. Figure 6.3 provides a graphical representation of the results obtained using the $CNCS$ algorithm on the Liver Disorders Data Set.

TABLE 6.3. Results For $TC = min(1 \times FP + 20 \times FN + 3 \times UC)$

Pima Indian Diabetes					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
	CNCS	51.7	0.0	28.6	137.5
J48	CNGA	42.3	0.8	26.3	137.2
	CNCS	52.2	0.0	28.6	138.0
SVM	CNGA	41.1	0.9	26.7	139.2
	CNCS	52.2	0.0	28.8	138.6
ANN	CNGA	39.1	1.2	25.9	140.8

Liver Disorders					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
	CNCS	29.0	0.0	53.9	190.7
J48	CNGA	29.9	0.3	46.7	176.0
	CNCS	27.5	0.0	49.3	175.4
SVM	CNGA	28.1	0.0	46.9	168.8
	CNCS	33.0	0.0	42.9	161.7
ANN	CNGA	32.5	0.0	39.1	149.8

Cardiotocography2					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
	CNCS	4.7	0.0	33.7	105.8
J48	CNGA	2.9	3.7	28.1	161.2
	CNCS	4.8	0.0	33.4	105.0
SVM	CNGA	4.8	0.0	32.8	103.2
	CNCS	4.8	0.0	33.0	103.8
ANN	CNGA	4.3	0.5	32.5	111.8

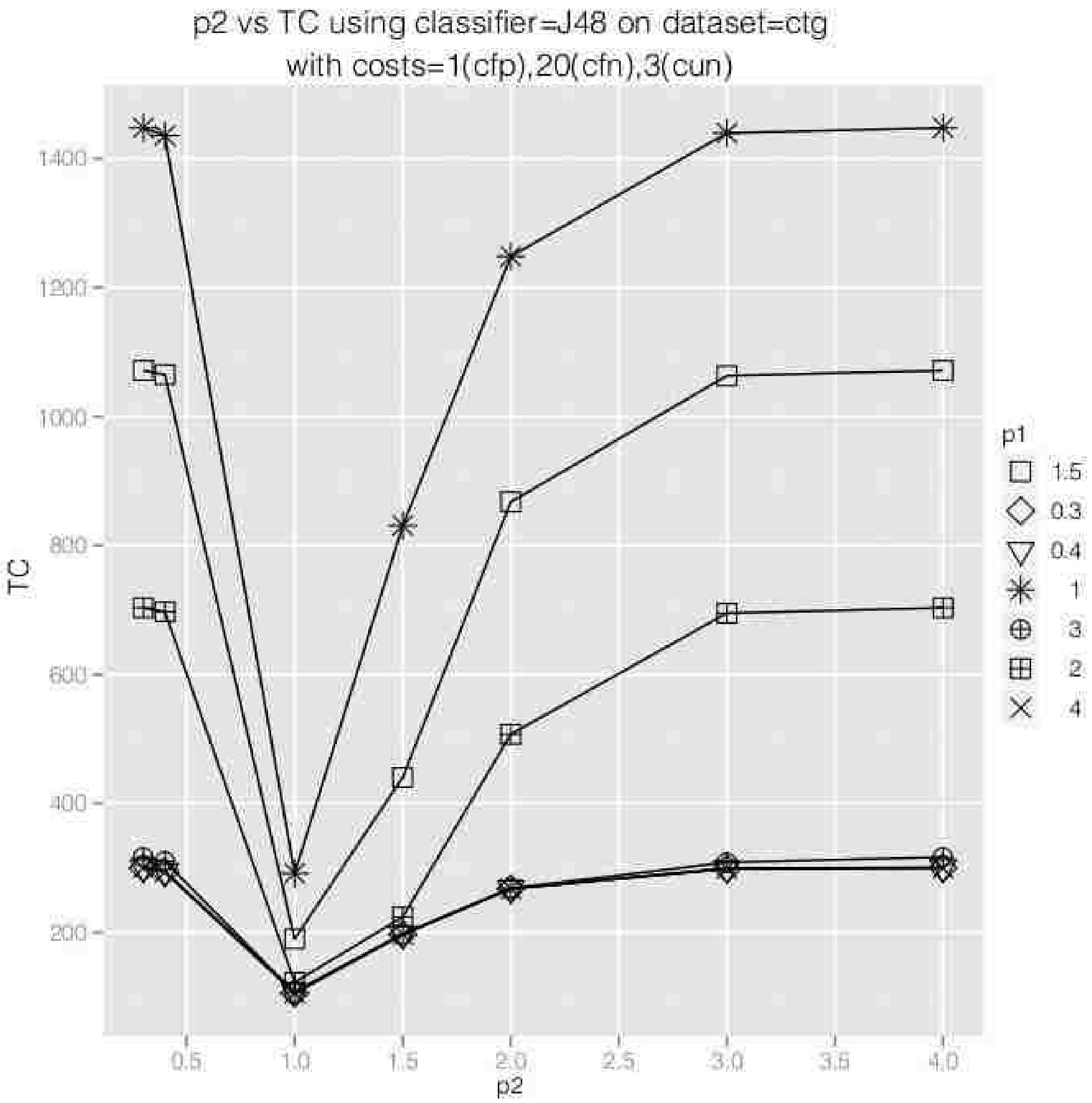


FIGURE 6.2: Applying $CNCS$ on 'Cardiotocography2' with $TC = min(1 \times FP + 20 \times FN + 3 \times UC)$

Using this cost function, a total of 18 experiments (9 for CNGA and 9 for CNCS) were conducted. Medical data sets were again used for this study. For this cost function, even small false negative rates are penalized heavily. So in this case, small false negative rates for CNGA inflates the TC cost (as an example see rows corresponding to the CNGA for the Pima Indian Diabetes dataset) well above the TC values for CNCS. For this cost function, CNCS outperforms CNGA in a majority of the experiments. One can also observe from Table 6.4 that the false negative rate is lower than both the false positive and undecided rates for all the experiments. Since the penalty for committing the false negative rate was the highest, the algorithms CNCS and CNGA have achieved their goals of reducing the rate (in this case, false negative) that has the highest penalty. Figure 6.3 illustrates the analysis of the Liver Disorders data set using the CNCS methodology.

TABLE 6.4. Results For $TC = min(1 \times FP + 100 \times FN + 3 \times UC)$

Pima Indian Diabetes					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
	CNCS	52.0	0.0	28.4	137.2
J48	CNGA	43.4	0.9	26.2	212.0
	CNCS	51.7	0.0	28.6	137.5
SVM	CNGA	40.1	1.0	26.2	218.7
	CNCS	50.9	0.0	28.3	135.8
ANN	CNGA	41.7	0.8	27.0	202.7
Liver Disorders					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
	CNCS	27.8	0.0	53.9	189.5
J48	CNGA	29.9	0.0	48.4	175.1
	CNCS	28.7	0.0	47.5	171.2
SVM	CNGA	29.3	0.0	45.5	165.8
	CNCS	33.9	0.0	42.6	161.7
ANN	CNGA	34.8	0.0	39.7	153.9
Cardiotocography2					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
	CNCS	4.7	0.0	33.0	103.7
J48	CNGA	2.8	3.4	28.3	427.7
	CNCS	4.7	0.0	33.6	105.5
SVM	CNGA	4.8	0.0	32.6	102.6
	CNCS	4.9	0.0	33.5	105.4
ANN	CNGA	4.3	0.7	32.1	170.6

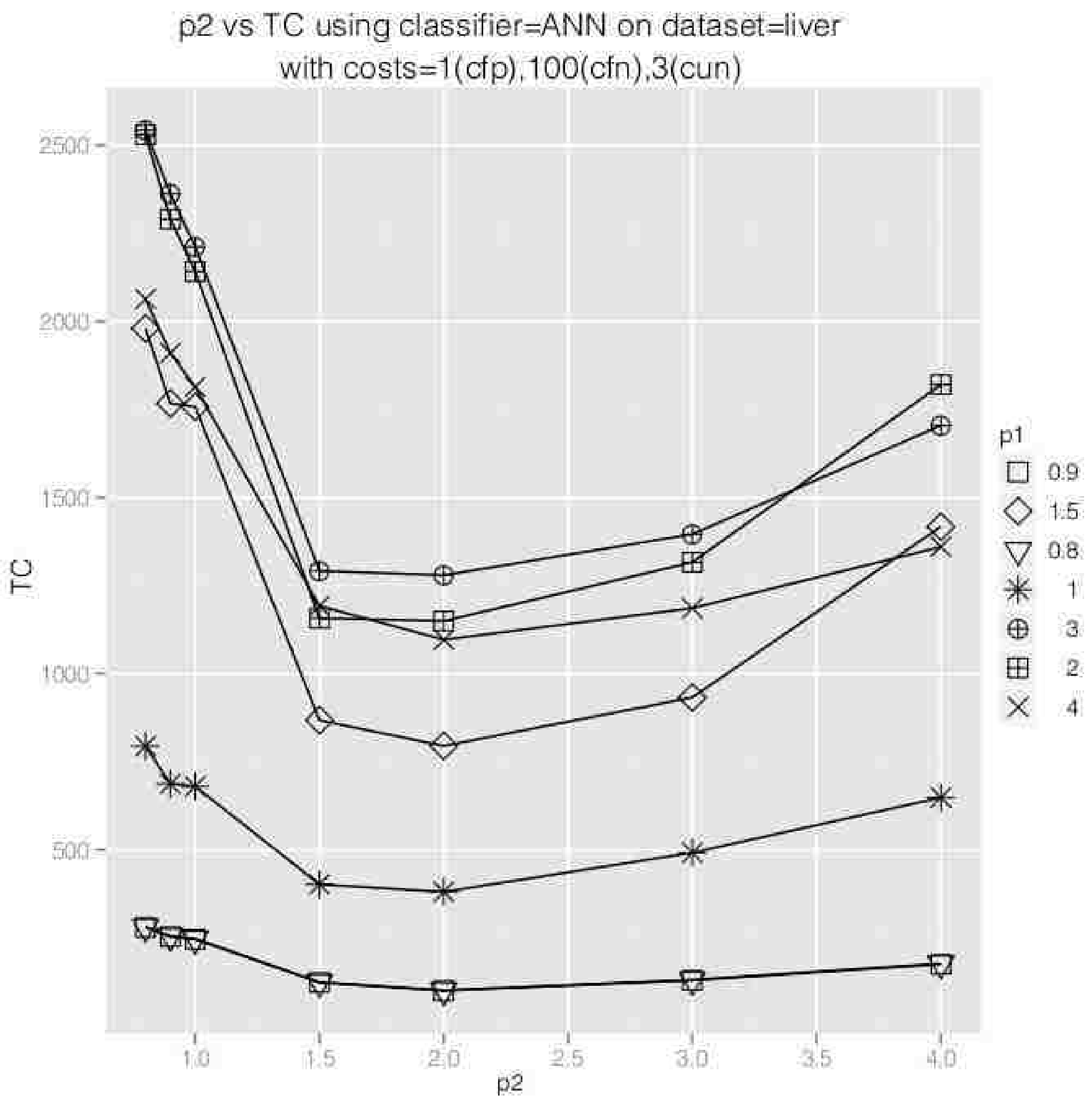


FIGURE 6.3: Applying $CNCS$ on 'Liver Disorders' with $TC = min(1 \times FP + 100 \times FN + 3 \times UC)$

6.1.4 Using $TC = min(20 \times FP + 1 \times FN + 3 \times UC)$

This cost function considers the case where it is costlier to commit a false positive error when compared to a false negative error. Two data sets are studied here - the German Credit data set and the Ionosphere data set. A total of 24 experiments were conducted (12 for CNGA and 12 for CNCS). The results are reported in Table 6.5. Figure 6.4 represents a graphical representation of the analysis of this cost function on the German Credit data set using CNCS. From Table 6.5, one can observe that CNCS and CNGA perform on par with one another for the German Credit Data set and for the Ionosphere data set, CNGA performs better than CNCS. One can also observe from the table that CNCS and CNGA are fulfilling the roles of reducing the rates that have the highest penalty (in this case the false positive rate).

TABLE 6.5. Results For $TC = min(20 \times FP + 1 \times FN + 3 \times UC)$

German Credit Data					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
J48	CNCS	0.0	56.9	21.1	120.2
	CNGA	0.0	56.9	21.1	120.2
SVM	CNCS	0.0	65.3	10.9	98.0
	CNGA	0.0	65.2	10.2	95.8
ANN	CNCS	0.0	65.7	9.4	93.9
	CNGA	0.0	62.6	13.9	104.3
Ionosphere					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
J48	CNCS	0.9	18.5	62.4	223.7
	CNGA	0.9	10.5	45.9	166.2
SVM	CNCS	0.6	17.7	63.0	218.7
	CNGA	1.1	9.1	45.6	167.9
ANN	CNCS	0.9	16.5	48.1	178.8
	CNGA	1.4	16.2	40.2	164.8

6.1.5 Using Cost Function $TC = min(100 \times FP + 1 \times FN + 3 \times UC)$

This function is similar to the previous cost function, except that in this case, the false positive rate is penalized even higher when compared to the false negative

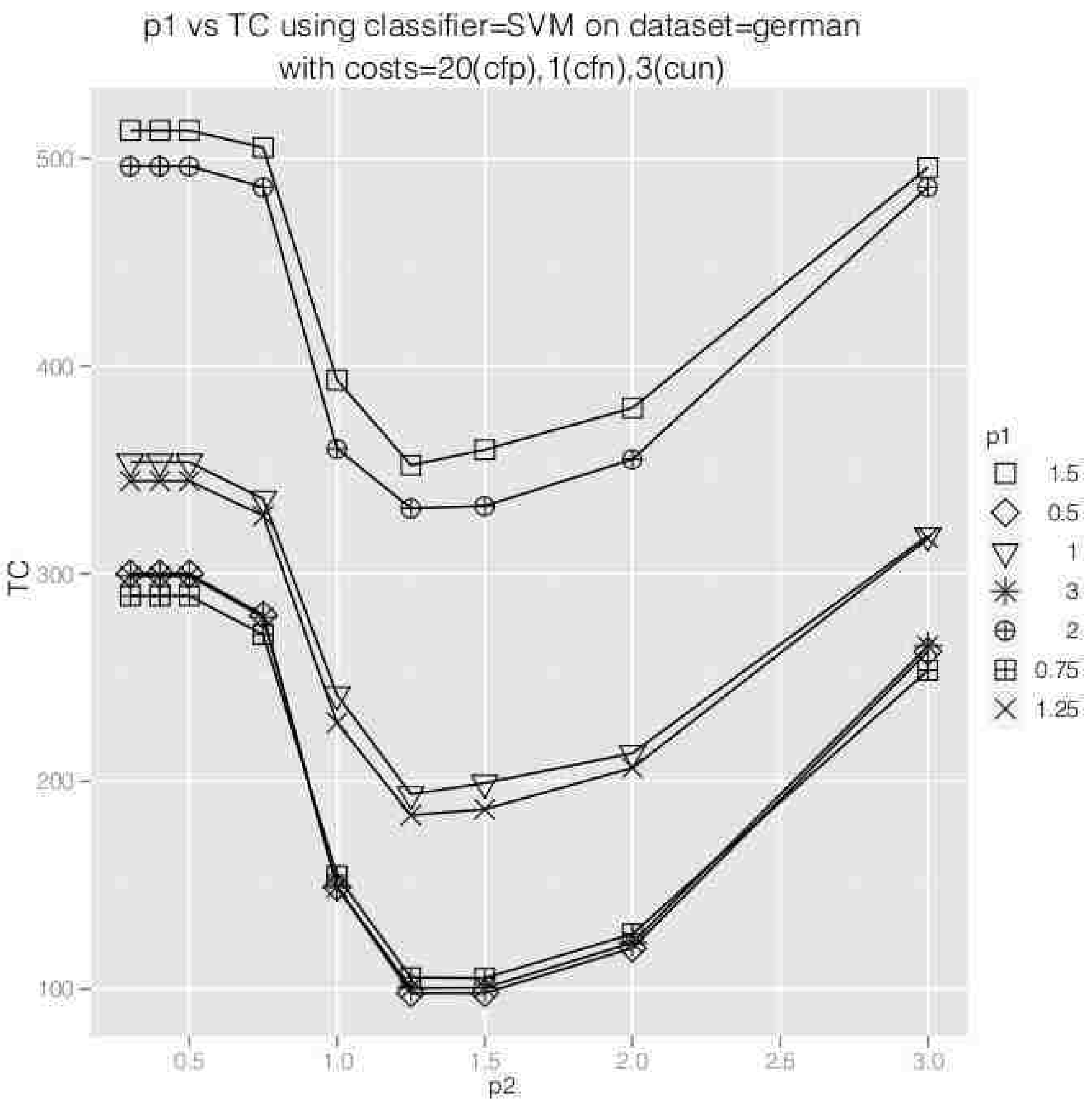


FIGURE 6.4: Applying $CNCS$ on 'German Credit' with $TC = min(20 \times FP + 1 \times FN + 3 \times UC)$

rate. The results are tabulated in Table 6.6. Figure 6.5 represents a graphical representation of the analysis of this cost function on the German Credit data set using CNCS.

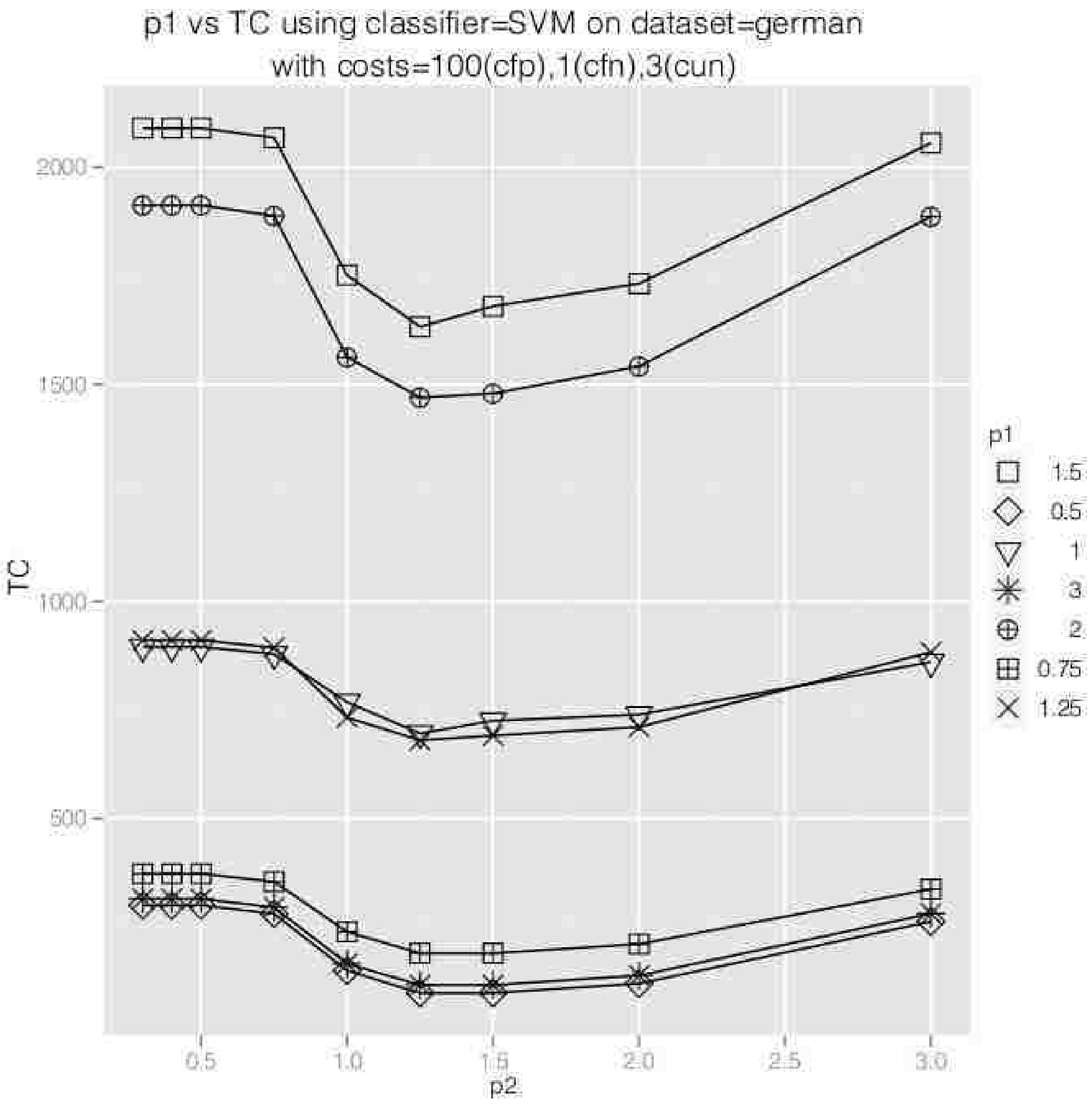


FIGURE 6.5: Illustration for $CNCS$ on $TC = min(100 \times FP + 1 \times FN + 3 \times UC)$

6.1.6 Statistical Comparisons between CNCS and CNGA

The computational test results of CNCS and CNGA (66 experimental results) were compared using the non-parametric statistical test, Wilcoxon test [49] (as

TABLE 6.6. Results For $TC = min(100 \times FP + 1 \times FN + 3 \times UC)$

German Credit Data					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
J48	CNCS	0.0	57.0	21.0	120.0
	CNGA	0.0	57.0	21.0	120.0
SVM	CNCS	0.0	65.3	10.9	98.0
	CNGA	0.0	65.2	10.2	95.8
ANN	CNCS	0.0	65.7	9.4	93.9
	CNGA	0.0	65.6	9.4	93.8
Ionosphere					
Base Classifier	Method	$FP(\%)$	$FN(\%)$	$UC(\%)$	TC
J48	CNCS	0.0	3.4	78.1	237.7
	CNGA	0.0	17.7	52.1	174.0
SVM	CNCS	0.0	17.9	81.5	262.4
	CNGA	0.0	17.7	53.3	177.6
ANN	CNCS	0.0	21.7	54.4	184.9
	CNGA	0	23.9	49.6	172.7

recommended for comparing two classifiers by studies such as [10] and [11]). The Wilcoxon test yielded a p-value of 0.4506 at $\alpha = 0.05$. Hence the null hypothesis that the two methods CNCS and CNGA are the same is NOT rejected. In other words, the differences between CNCS and CNGA are statistically not significant and yield similar performance.

6.1.7 Computational Time Comparisons between CNCS and CNGA

In this subsection, the time to run CNGA and CNCS across various data sets, cost functions and base classifiers is compared.

For each of the data sets in question, the average run time using CNCS and CNGA were computed. The results are outlined in Table 6.7. Note that the CNCS method consistently runs faster than the CNGA approach. The reason for the faster run time for CNCS is that it uses a smaller candidate set of values. On the other hand, CNGA considers a wider, randomized range of values to search for lower TC values than the smaller range of values used by CNCS. Hence the time to compute, iteratively, the different crossover and mutating gene sequences in CNGA yields a longer average computation time than CNCS.

6.1.8 Summary

In summary, the above study establishes the following conclusions. CNCS and CNGA were applied on a number of data sets using different cost functions. In this section, a total of 66 computational experiments were conducted using different data sets and cost functions. In a majority of the 66 experiments, CNCS performs on par with CNGA. Though CNGA investigates a lot more possibilities than CNCS (which uses a candidate search space), CNCS and CNGA consistently track each other's performance closely. One can observe from the 66 experiments that CNGA provides only a marginal improvement over CNCS (in terms of the number of times it outperforms CNCS and the lowering of TC values using CNGA is only marginal). However, when comparing the average time to run CNGA and CNCS, CNCS outshines CNGA by a maximum speedup factor of approximately 33.5 for one dataset and by a minimum speedup factor of 4.8 for another dataset. Clearly from Table 6.7, the time to compute minimum TC using CNCS is much shorter than CNGA. In comparison, the gains obtained in minimizing TC using CNGA is marginal when compared to the overall gain of running a faster CNCS to obtain TC values that are close to the values obtained by CNGA. In summary, the shorter computation time for CNCS far outweighs the marginal improvements in TC values using CNGA. CNCS is a better choice for minimizing TC values.

TABLE 6.7. Computational Time Comparison Between CNCS and CNGA

Data set	Method	Average Run Time (in Hours)	Speedup
Liver Disorders	CNCS	0.1	30
	CNGA	3.0	
Pima	CNCS	0.2	33.5
	CNGA	6.7	
German	CNCS	0.6	7.3
	CNGA	4.4	
Geographical Data Set, GA	CNCS	2.7	9.2
	CNGA	24.8	
Ionosphere	CNCS	0.4	5.0
	CNGA	2.0	
Cardiotography	CNCS	1.6	4.8
	CNGA	7.7	

6.2 Optimizing CNCS (The $CNCS$-OPT algorithm)

6.2.1 Using $CNCS - OPT$ To Minimize TC

The $CNCS$ algorithm is a nested loop algorithm with the core loop having a $O(n^2)$ complexity. However, from the graphs depicted in Figures 6.2, 6.3, 6.4 and 6.5, one can see that the graphs have a point of inflexion for each of the line graphs that depict TC vs. p_1 for different values of p_2 and TC vs. p_2 for different values of p_1. Assuming one starts with a fixed p_1 value and then a study is conducted for the relationship, p_2 vs. TC by varying p_2 alone, one can find a minimum value of TC for a given, fixed p_1. The value of p_2 that yielded the minimum TC is now fixed and for a different range of values of p_1, TC is computed again. Using this newly computed array of TC values, one can find the minimum TC value. Essentially, the $O(n^2)$ component of the CNCS algorithm's time complexity is now reduced to $O(2n)$, leading to a decrease in computation time for CNCS. This optimized methodology is outlined as CNCS-OPT (for CNCS-Optimized) and is outlined in Figure 6.6. On the same note, the $CNCS - OPT$ and $CNCS$ are similar algorithms, yielding the same results (TC values) as each other. The only difference between the two methodologies is that $CNCS - OPT$ is faster than $CNCS$. Hence the TC results provided for $CNCS$ in Section 6.1 hold for $CNCS - OPT$ as well.

6.2.2 Comparing $CNCS$-OPT and $CNCS$

The computational performance of $CNCS - OPT$ and $CNCS$ was compared by conducting a series of experiments on various data sets. The average computation time for each data set was computed and has been tabulated in Table 6.8. From the table, one can observe the significant boost in performance using $CNCS - OPT$ when compared to using $CNCS$.

TABLE 6.8. Computational Time Analysis For CNCS and CNCS-OPT

Data set	Method	Average Run Time (in Hours)	Speedup
Liver Disorders	CNCS	0.1	
	CNCS-OPT	0.03	3.3
Pima	CNCS	0.2	
	CNCS-OPT	0.06	3.3
German	CNCS	0.6	
	CNCS-OPT	0.2	3.0
Geographical Data Set, GA	CNCS	2.7	
	CNCS-OPT	1.1	2.5
Ionosphere	CNCS	0.4	
	CNCS-OPT	0.1	4.0
Cardiotocography2	CNCS	1.6	
	CNCS-OPT	0.4	4.0

6.3 $CNCS - OPT$ using Original Misclassification Cost Definition TC_0

6.3.1 Using $CNCS - OPT$ To Minimize TC_0

Recall from Figure 5.1, the area of uncertainty where the positive and negative critical nuggets model conflict. In the earlier section, the TC definition factored in the undecided case to account for the area of uncertainty. In this section, the original TC definition, TC_0 is used which includes only the false positive and false negative rates. In this case, the data records lying in the area of uncertainty are

```
Require: $c_{FP}$, $c_{FN}$, $c_{UC}$: the costs for false positive, false negative and undecided cases.
Require: : Critical Nuggets for classes '+' and '-' and data set, $T_r$.
 1: $TCArray$=$\phi$
 2: From the range of $R$ values used in calculating critical nuggets, find candidate set of
    size ratios, $CS_1$ and $CS_2$ for positive and negative critical nuggets respectively.
 3: Select a $p_1$ randomly from the set $(CS)_1$.
 4: Proportionally change size of all Positive Critical Nuggets using ratio $p_1$.
 5: Generate Positive Critical Nuggets Model, $M^+_{nuggets}$, using new Positive Critical
    Nuggets.
 6: for each ratio $p_2$ in $CS_2$ do
 7:     Proportionally change size of all Negative Critical Nuggets using ratio $p_2$.
 8:     Generate Negative Critical Nuggets Model, $M^-_{nuggets}$.
 9:     Use $M^+_{nuggets}$, $M^-_{nuggets}$ and $M_0$ to predict class labels for $T_r$.
10:     $FP$=Number of False Positives/$|T_r|$
11:     $FN$=Number of False Negatives/$|T_r|$
12:     $UC$ = Number of Undecided Cases/$|T_r|$
13:     $TC = (c_{FP}$ x $FP + c_{FN}$ x $FN + c_{UC}$ x $UC)$ x 100
14:     Append $TC$ to $TCArray$
15: end for
16: $tempMin$=Minimum from $TCArray$.
17: $tempIndx$ = Index of $TCArray$ where $tempMin$ occurs.
18: Reset $TCArray$=$\phi$
19: $p_2$ = Find $tempIndx$ element from $CS_2$
20: Proportionally change size of all Negative Critical Nuggets using ratio $p_2$.
21: Generate Negative Critical Nuggets Model, $M^-_{nuggets}$.
22: for each ratio $p_1$ in $CS_1$ do
23:     Proportionally change size of all Positive Critical Nuggets using ratio $p_1$.
24:     Generate Positive Critical Nuggets Model, $M^+_{nuggets}$, using new Positive Critical
        Nuggets.
25:     Use $M^+_{nuggets}$, $M^-_{nuggets}$ and $M_0$ to predict class labels for $T_r$.
26:     $FP$=Number of False Positives/$|T_r|$
27:     $FN$=Number of False Negatives/$|T_r|$
28:     $UC$ = Number of Undecided Cases/$|T_r|$
29:     $TC = (c_{FP}$ x $FP + c_{FN}$ x $FN + c_{UC}$ x $UC)$ x 100
30:     Append $TC$ to $TCArray$
31: end for
32: Find minimum $TC$ from $TCArray$.
```

FIGURE 6.6: The $CNCS - OPT$ Algorithm.

classified using the original base classifier's prediction. Since $CNCS - OPT$ is the best choice both in terms of computation time and the ability to minimize TC (with undecided case factored), $CNCS - OPT$ is applied now to TC_0. Since standard data mining software implementations and research literature conform to the standard misclassification cost definition of TC_0, it provides a basis for comparing $CNCS - OPT$'s performance with other standard base classifiers.

For the computational analysis with $CNCS - OPT$ and TC_0 minimization, 72 computational experiments were performed using various data sets, cost functions and base classifiers. The base classifiers used were the same as earlier in the chapter - J48, SVM and ANN.

6.3.2 Computational Analysis For Different Cost Functions

Tests were conducted using 4 different cost functions. The cost functions used in the study were the following:

1. $TC_0 = min(1 \times FP + 20 \times FN)$. This cost function penalizes the false negative rate with a cost that is 20 times higher than penalty for the false positive rate.

2. $TC_0 = min(1 \times FP + 100 \times FN)$. This cost function penalizes the false negative rate with a cost that is 100 times higher than penalty for the false positive rate.

3. $TC_0 = min(20 \times FP + 1 \times FN)$. This cost function penalizes the false positive rate more heavily than the false negative rate.

4. $TC_0 = min(100 \times FP + 1 \times FN)$. This cost function penalizes the false positive rate with a penalty that is 100 times that of a false negative rate.

The $CNCS - OPT$ algorithm was run on different data sets using each of the cost functions. For the first 2 cost functions, a total of 36 experiments were conducted. For the latter 2 cost functions, 36 experiments were conducted. In total, 72 computational experiments were conducted using 3 base classifiers and 5 different real-world data sets. In these set of tests, the TC_0 values obtained by $CNCS - OPT$ can be compared with the TC_0 values obtained by the standard classifiers such as J48, SVM and ANN. The results for each of the cost functions have been tabulated in Tables 6.9, 6.10, 6.11 and 6.12.

TABLE 6.9. Results For $TC_0 = min(1 \times FP + 20 \times FN)$

		Base Classifier			With CNCS-OPT		
Dataset	Classifier	FP	FN	TC_0	FP	FN	TC_0
Pima	J48	10.6	14.7	304.6	58.1	4.9	156.1
	SVM	7.8	15.8	323.8	54.4	5.9	172.4
	ANN	10.5	14.6	302.5	57.7	5.5	167.7
Liver Disorders	J48	19.6	14.1	301.6	35.9	11.0	255.9
	SVM	23.3	6.7	157.3	30.4	12.2	274.4
	ANN	18.1	13.1	280.1	41.7	7.0	181.7
Cardiotocography2	J48	2.5	3.7	76.5	7.3	1.6	39.3
	SVM	1.9	4.8	97.9	8.3	1.4	36.3
	ANN	2.8	3.3	68.8	6.9	2.1	48.9
German	J48	15.6	11.4	243.6	23.0	5.6	135.0
	SVM	26.9	1.5	56.9	29.3	0.6	41.3
	ANN	15.7	13.0	275.7	25.2	5.0	125.2
Ionosphere	J48	6.2	3.9	84.2	9.4	1.1	31.4
	SVM	1.3	5.7	115.3	7.1	3.1	69.1
	ANN	6.9	1.2	30.9	9.7	0.9	27.7
Synthetic Geographical Data Set(GA)	J48	0.6	1.0	20.6	11.2	2.2	55.2
	SVM	2.0	2.4	50.0	9.3	4.0	89.3
	ANN	7.8	14.6	299.8	37.4	1.3	63.4

6.3.3 Statistical Comparisons between CNCS-OPT and Base Classifiers

The computational test results of CNCS-OPT and the standalone or base classifiers (72 experimental results) were compared using the non-parametric statistical test,

TABLE 6.10. Results For $TC_0 = min(1 \times FP + 100 \times FN)$

Dataset	Classifier	Base Classifier FP	Base Classifier FN	Base Classifier TC_0	With CNCS-OPT FP	With CNCS-OPT FN	With CNCS-OPT TC_0
Pima	J48	11.4	14.4	1451.4	57.4	4.9	547.4
	SVM	7.8	15.7	1577.8	55.6	5.3	585.6
	ANN	11.0	13.9	1401.0	53.9	4.6	513.9
Liver Disorders	J48	20.1	14.2	1440.1	34.2	13.0	1334.2
	SVM	23.5	6.6	683.5	31.0	11.9	1221.0
	ANN	18.6	12.7	1288.6	38.8	6.1	648.8
Cardiotocography2	J48	2.6	3.6	362.6	7.9	1.4	147.9
	SVM	1.9	4.7	471.9	8.2	1.4	148.2
	ANN	2.9	3.5	352.9	7.4	1.9	197.4
German	J48	15.5	11.4	1155.5	23.6	4.4	463.6
	SVM	26.9	1.5	176.9	28.8	0.4	68.8
	ANN	15.1	13.6	1375.1	24.6	4.7	494.6
Ionosphere	J48	6.6	3.6	366.6	11.4	1.7	181.4
	SVM	1.4	5.6	561.4	6.8	2.8	286.8
	ANN	7.4	1.7	177.4	10.0	0.6	70.0
Synthetic Geographical Data Set(GA)	J48	0.6	1.0	100.6	5.8	1.8	185.8
	SVM	2.0	2.4	242.0	25.5	2.7	295.5
	ANN	6.6	15.7	1576.6	71.4	0.5	121.4

TABLE 6.11. Results For $TC_0 = min(20 \times FP + 1 \times FN)$

Dataset	Classifier	Base Classifier FP	Base Classifier FN	Base Classifier TC_0	With CNCS-OPT FP	With CNCS-OPT FN	With CNCS-OPT TC_0
Pima	J48	11.8	14.1	250.1	8.7	20.1	194.1
	SVM	8.1	15.6	177.6	5.6	22.1	134.1
	ANN	11.5	13.7	243.7	11.2	14.7	238.7
Liver Disorders	J48	20.6	13.7	425.7	5.5	28.4	138.4
	SVM	23.2	6.7	470.7	3.2	29.9	93.9
	ANN	17.7	13.5	367.5	7.8	22.3	178.3
Cardiotocography2	J48	2.6	3.7	55.7	3.3	2.6	68.6
	SVM	1.9	4.7	42.7	4.7	2.1	96.1
	ANN	2.7	3.4	57.4	2.5	12.2	62.2
German	J48	16.0	12.0	332.0	4.1	60.9	142.9
	SVM	27	1.6	541.6	5.0	65.8	165.8
	ANN	15.6	13.1	325.1	2.1	68.1	110.1
Ionosphere	J48	6.8	4.3	140.3	6.6	4.3	136.3
	SVM	1.4	5.7	33.7	1.4	5.1	33.1
	ANN	7.0	1.8	141.8	6.0	10.3	130.3
Synthetic Geographical Data Set(GA)	J48	1.0	1.0	21.0	2.6	11.9	63.9
	SVM	2.0	2.0	42.0	2.7	12.2	66.2
	ANN	7.3	15.0	161.0	4.9	19.3	117.3

TABLE 6.12. Results For $TC_0 = min(100 \times FP + 1 \times FN)$

		Base Classifier			With CNCS-OPT		
Dataset	Classifier	FP	FN	TC_0	FP	FN	TC_0
Pima	J48	11.3	14.2	1144.2	10.2	16.5	1036.5
	SVM	7.8	15.6	795.6	7.0	17.3	717.3
	ANN	10.5	14.3	1064.3	11.5	13.2	1163.2
Liver Disorders	J48	20.6	14.0	2074.0	4.1	30.1	440.1
	SVM	23.1	6.9	2316.9	2.9	29.3	319.3
	ANN	18.4	11.9	1851.9	6.1	26.1	636.1
Cardiotocography2	J48	2.6	3.7	263.7	2.5	69.7	319.7
	SVM	1.9	4.7	194.7	2.0	73.1	273.1
	ANN	2.9	3.5	293.5	2.1	69.1	279.1
German	J48	15.9	11.3	1601.3	4.6	60.3	520.3
	SVM	26.9	1.4	2691.4	5.2	65.6	585.6
	ANN	14.9	13.6	1503.6	1.7	67.5	237.5
Ionosphere	J48	7.0	3.8	703.8	6.3	4.0	634.0
	SVM	1.4	5.8	145.8	1.4	6.3	146.3
	ANN	7.3	1.6	731.6	5.1	22.5	532.5
Synthetic Geographical Data Set(GA)	J48	0.6	1.0	61.0	1.7	12.2	182.2
	SVM	2	2.4	202.4	2.7	12.2	282.2
	ANN	8.4	14.1	854.1	5.3	19.5	549.5

Wilcoxon test [49] (as recommended for comparing two classifiers by studies such as [10] and [11]). The results of the test are tabulated in Table 6.13. One can observe from Table 6.13 that CNCS-OPT's reductions in misclassification cost are statistically significant at 99% confidence level when used in conjunction with J48 and ANN and significant at 90% confidence level when used in conjunction with SVM.

6.4 Summary

As a summary for this chapter, two new approaches - CNCS (Critical Nuggets With a Candidate Set) and CNGA (Critical Nuggets and a Genetic Algorithm) have been proposed to minimize misclassification cost, TC. Two misclassification cost functions were considered - one was the standard misclassification cost (TC_0) that included only false positive and false negative costs while the other was a

TABLE 6.13. Significance of Misclassification Cost Reductions

Comparison	p-value	Significance
J48 vs. CNCS-OPT with J48	0.0027	Significant (at $\alpha = 0.01$)
SVM vs. CNCS-OPT with SVM	0.0787	Significant (at $\alpha = 0.1$)
ANN vs. CNCS-OPT with ANN	0.0000	Significant (at $\alpha = 0.01$)

modified cost function that included the undecided case. Using the modified TC cost function which accounts for the undecided case, a total of 66 experiments (33 for CNCS and 33 for CNGA) were conducted to analyze the performance of CNCS and CNGA. This analysis was conducted using various cost functions, different base classifiers and various real-world and synthetic data sets. Summarizing the results, overall the performance of CNCS and CNGA is on par with each other. In other words, given the randomized input data (due to a 10-fold cross-validation), the performance of CNCS and CNGA in terms of output TC scores was relatively similar and statistically not different. However, the time to compute CNCS is far shorter than CNGA. This makes the algorithm CNCS a better choice when compared to CNGA for the minimization of TC (a TC function that factors in the undecided cost).

The CNCS algorithm was optimized further by the proposing of CNCS-OPT - an optimized version of CNCS. This reduced the complexity of CNCS by an order of magnitude and helped speed up the computations further. The CNCS-OPT was compared with CNCS by running the same set of 66 experiments using CNCS-OPT. CNCS-OPT was on average 3 times faster than CNCS.

Using CNCS-OPT, 72 additional experiments were conducted using the non-modified TC function, TC_0 that factors in only false positive and false negative rates. The TC_0 values produced by CNCS-OPT and standard classification algorithms were compared and CNCS-OPT performed better than the standard classification algorithms in 56 out of 72 experiments. The 16 cases where CNCS-OPT failed to reduce misclassification costs maybe attributed to the highly representative nature of the data sets such as the Cardiotocography2 and the Synthetic Geographical data set (GA). Higher the representation of the training data set, better will be the accuracy of data sets such as Cardiotocography2 and GA (as indicated in Table 4.6). Higher the accuracy, lower are the misclassification costs. In such a case, running CNCS-OPT, where areas near the boundary expand and grow, may nullify the pre-existing low misclassification costs (prior to running CNCS-OPT). Overall, the reductions in misclassification costs by CNCS-OPT in comparison to the base classifiers is statistically significant and CNCS-OPT outperforms the base classifiers. This is confirmed by the Wilcoxon test results illustrated in Table 6.13.

Chapter 7
Additional Computational Analysis

Additional computational analysis was conducted to study the relationship between the effect of false positive rates and false negative rates when their respective penalties or costs are varied. Two such relationships were evaluated:

- one in which the false negative rate (FN) was kept constant and the impact of false positive rate FP was studied by varying the false positive cost c_{FP}.
- the other in which the false positive rate (FP) was kept constant and the impact of false negative rate (FN) was studied by varying the false negative cost (c_{FN}).

7.1 Analyzing FP vs c_{FP} by keeping c_{FN} constant

To conduct this analysis, the CNCS-OPT algorithm was applied on the modified TC function that includes the undecided case. The false positive cost was varied and the false negative cost was kept constant. The goal was to study the relationship between false positive rate and the false positive cost by keeping the false negative cost constant.

Figure 7.1 depicts the results of one such experiment using SVM as the base classifier on the German Credit Data Set. Note that as the false positive cost rises, initially the false positive rate hold steady at 0.7 and then it falls to 0 as one continues to increase the cost. A step-wise pattern in the graph is an interesting observation as well as there exist levels where false positive rates hold steady and after a certain cost threshold is crossed, then the false positive rate falls. Also as cost increases, the false positive rate falls.

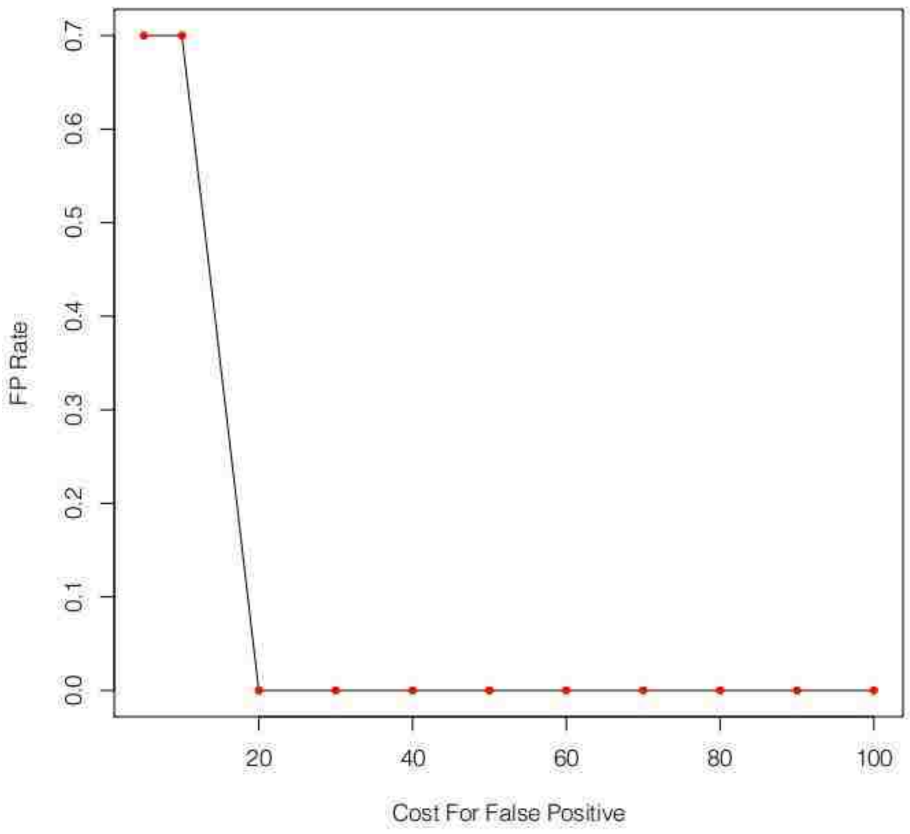


FIGURE 7.1: Analyzing c_{FP} vs FP using SVM on German Credit Data Set.

7.2 Analyzing FN vs c_{FN} by keeping c_{FP} constant

In this section, $CNCS-OPT$ is again applied using the modified TC equation. This time, however, the false negative rate is studied in relationship to increasing false negative cost. One such experiment in this category is outlined as Figure 7.2. One can again observe a step-wise pattern in the graph. As one increases the cost, the false negative rate decreases in a step-wise pattern.

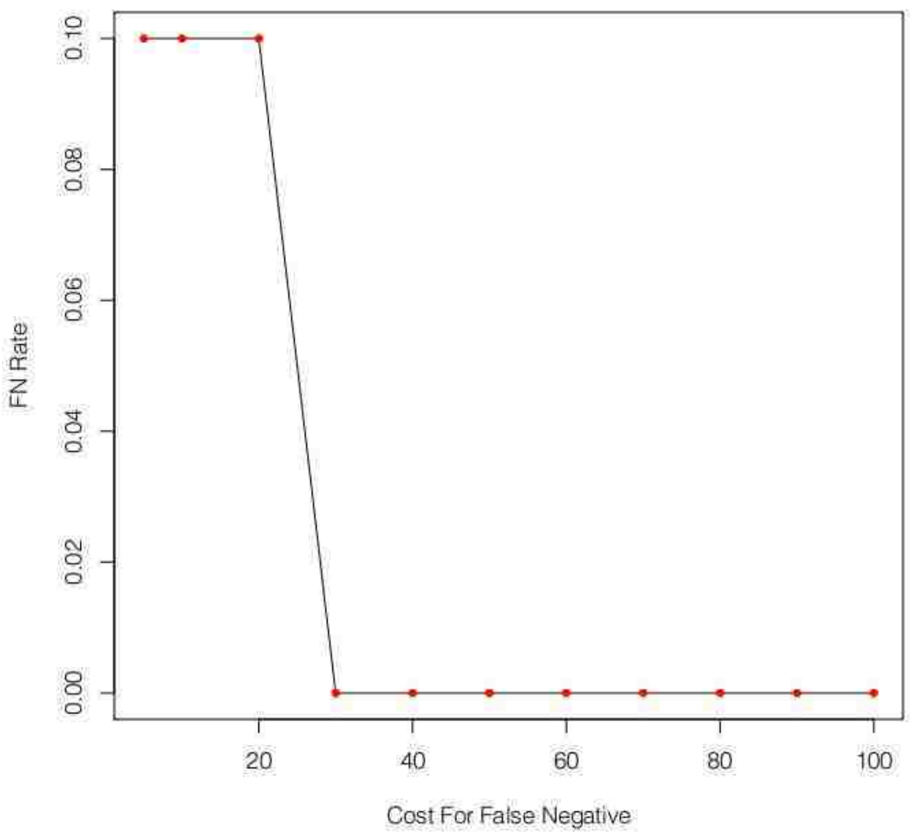


FIGURE 7.2: Analyzing c_{FP} vs FP using J48 on the Cardiotocography2 Data Set.

Chapter 8
Conclusions

This research work presents the notion of critical nuggets. A new metric, the CR_{score}, was introduced for measuring criticality of a subset or nugget. A simple rotation test was proposed to resolve conflicting scores when they occur. The proposed score was used to identify critical nuggets. The tests on a number of 2-dimensional synthetic data sets provided a visual validation that such nuggets are more likely to lie near class boundaries and in close proximity to the complex features along the class boundaries. Reducing the search to near the class boundaries saves computation time in identifying such nuggets. The $FindCriticalNuggets$ algorithm was outlined that used the boundary estimation method and the CR_{score} to identify critical nuggets. Some important properties such as the dual nature of critical nuggets were discussed and the properties were validated through some sets of experiments. The proposed ideas were tested on some multi-dimensional real world data sets. Results from the experiments on the real world data sets revealed that only a very small number of subsets qualified as critical nuggets. Experimental results from the real world data sets also indicated the importance of finding such subsets in large databases. The knowledge of critical nuggets also helped to reduce the number of false positives and false negatives and thus significantly improving the overall accuracy of classification tasks.

A detailed study was undertaken to study the application of critical nuggets in lowering the classification costs. Two definitions of classification costs were considered. A modified classification cost (TC) accounted for the unclassifiable case and another cost definition (TC_0) included only the false positive and false negative rates. 2 algorithms, $CNCS$ and $CNGA$, were introduced and applied on the modified definition TC. Both algorithms helped lower the total classification cost. Empirical studies were carried out for various cost scenarios and for a given search space, the optimal minimum classification cost was obtained (this minimum cost also provided for the best combination of the sizes of positive and negative critical nuggets). The $CNCS$ algorithm was further improved and optimized as $CNCS - OPT$ algorithm. $CNCS - OPT$ was then applied on the original cost definition TC_0 and empirical studies proved that the critical nuggets approach fares better (statistically significant) than the stand-alone algorithms in reducing misclassification costs.

This work concentrated on finding a subset of data records that are critical. In combination with critical data records, there maybe some attributes that may be more important that the others. Critical nuggets with critical attributes has immense applications (for instance, finding undecided voters (critical data records) in an election data set and the specific, core issues (critical attributes) that mark their undecidedness). Future work can be done towards improving the $O(n^2)$ complexity of the boundary approximation algorithm used in this research work. The work was limited to data sets with 2 classes and to data sets that have numerical attributes. Future work can be directed towards extending these ideas to data sets with multiple classes (greater than 2) and data sets with mixed attributes(numeric and categorical). The post-processing methodology of improving classification accuracy proposed in this work can also be compared with other techniques (including resampling techniques and other cost-sensitive classification methods) in the field of classification algorithms. Ideas used in reducing misclassification costs using critical nuggets can also be extended towards designing critical nuggets based cost-sensitive classifiers.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Database mining: A performance perspective. *IEEE Trans. Knowl. Data Eng.*, 5(6):914–925, 1993.

[2] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] Fabrizio Angiulli, Stefano Basta, and Clara Pizzuti. Distance-based detection and prediction of outliers. *IEEE Trans. Knowl. Data Eng.*, 18(2):145–160, 2006.

[4] Fabrizio Angiulli and Clara Pizzuti. Outlier mining in large high-dimensional data sets. *IEEE Trans. Knowl. Data Eng.*, 17(2):203–215, 2005.

[5] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors, *KDD*, pages 29–38. ACM, 2003.

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, 2000.

[7] Bruce Chan, Carmen Lewis. A basic primer on data mining. *Information Systems Management*, 19(4):56, 2002.

[8] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), 2009.

[9] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[10] J Demsar. Statistical comparisons of classifiers over multiple data sets. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:1–30, JAN 2006.

[11] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[12] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

[13] Lian Duan, Lida Xu, Ying Liu, and Jun Lee. Cluster-based outlier detection. *Annals of Operations Research*, 168(1):151–168, April 2009.

[14] Dara M. Farrell, Barbara S. Minsker, David Tcheng, Duane Searsmith, Jane Bohn, and Dennis Beckman. Data mining to improve management and reduce costs of environmental remediation. *JOURNAL OF HYDROINFORMATICS*, 9(2):107–121, APR 2007.

[15] U Fayyad, G PiatetskyShapiro, and P Smyth. From data mining to knowledge discovery in databases. *AI MAGAZINE*, 17(3):37–54, FAL 1996.

[16] U Fayyad and R Uthurusamy. Data mining and knowledge discovery in databases. *COMMUNICATIONS OF THE ACM*, 39(11):24–26, NOV 1996.

[17] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[18] Alex Alves Freitas. Understanding the crucial differences between classification and discovery of association rules - a position paper. *SIGKDD Explorations*, 2(1):65–69, 2000.

[19] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38, September 2006.

[20] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Min. Knowl. Discov.*, 16(3):349–364, 2008.

[21] David E. Goldberg and John H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3:95–99, 1988.

[22] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[23] D. J. Hand, C. Whitrow, N. M. Adams, P. Juszczak, and D. Weston. Performance criteria for plastic card fraud detection tools. *JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY*, 59(7):956–962, JUL 2008.

[24] David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of data mining*. MIT Press, Cambridge, MA, USA, 2001.

[25] D. Hawkins. *Identification of Outliers (Monographs on Statistics and Applied Probability)*. Springer, 1980.

[26] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[27] Kurt Hornik, Christian Buchta, and Achim Zeileis. Open-source machine learning: R meets Weka. *Computational Statistics*, 24(2):225–232, 2009.

[28] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[29] Edwin M. Knorr, Raymond T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB J.*, 8(3-4):237–253, 2000.

[30] Anna Koufakou and Michael Georgiopoulos. A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes. *Data Mining and Knowledge Discovery*, 20(2, Sp. Iss. SI):259–289, MAR 2010.

[31] Anna Koufakou, Jimmy Secretan, and Michael Georgiopoulos. Non-derivable itemsets for fast outlier detection in large high-dimensional categorical data. *KNOWLEDGE AND INFORMATION SYSTEMS*, 29(3):697–725, DEC 2011.

[32] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. *Machine Learning*, 95(1-2):161–205, 2005.

[33] Max A. Little, Patrick E. McSharry, Eric J. Hunter, Jennifer L. Spielman, and Lorraine O. Ramig. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE Trans. Biomed. Engineering*, 56(4):1015–1022, 2009.

[34] Kenneth McGarry. A survey of interestingness measures for knowledge discovery. *Knowledge Eng. Review*, 20(1):39–61, 2005.

[35] Fernando E. B. Otero, Alex Alves Freitas, and Colin G. Johnson. A new sequential covering strategy for inducing classification rules with ant colony algorithms. *IEEE Trans. Evolutionary Computation*, 17(1):64–76, 2013.

[36] Navneet Panda, Edward Y. Chang, and Gang Wu. Concept boundary detection for speeding up svms. In William W. Cohen and Andrew Moore, editors, *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 681–688. ACM, 2006.

[37] Parag Pendharkar and Sudhir Nanda. A misclassification cost-minimizing evolutionary-neural classification approach. *NAVAL RESEARCH LOGISTICS*, 53(5):432–447, AUG 2006.

[38] C. S. Perone. Pyevolve 0.6, 2010.

[39] Huy Nguyen Anh Pham and Evangelos Triantaphyllou. An application of a new meta-heuristic for optimizing the classification accuracy when analyzing some medical datasets. *Expert Syst. Appl.*, 36(5):9240–9249, 2009.

[40] Huy Nguyen Anh Pham and Evangelos Triantaphyllou. A meta-heuristic approach for improving the accuracy in some classification algorithms. *Computers & OR*, 38(1):174–189, 2011.

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

[42] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0.

[43] David Sathiaraj and Evangelos Triantaphyllou. On identifying critical nuggets of information during classification tasks. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2012.

[44] Yufei Tao, Xiaokui Xiao, and Shuigeng Zhou. Mining distance-based outliers from large databases in any metric space. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *KDD*, pages 394–403. ACM, 2006.

[45] E. Triantaphyllou. *Data Mining and Knowledge Discovery via Logic-Based Methods*. Springer, 2010.

[46] G. van Rossum et al. *Python:An object oriented programming language*, 1991.

[47] R. Andrew Weekley, Robert K. Goodrich, and Larry B. Cornman. An Algorithm for Classification and Outlier Detection of Time-Series Data. *Journal Of Atmospheric AND Oceanic Technology*, 27(1):94–107, JAN 2010.

[48] Yael Weiss, Yuval Elovici, and Lior Rokach. The CASH algorithm-cost-sensitive attribute selection using histograms. *INFORMATION SCIENCES*, 222:247–268, FEB 10 2013.

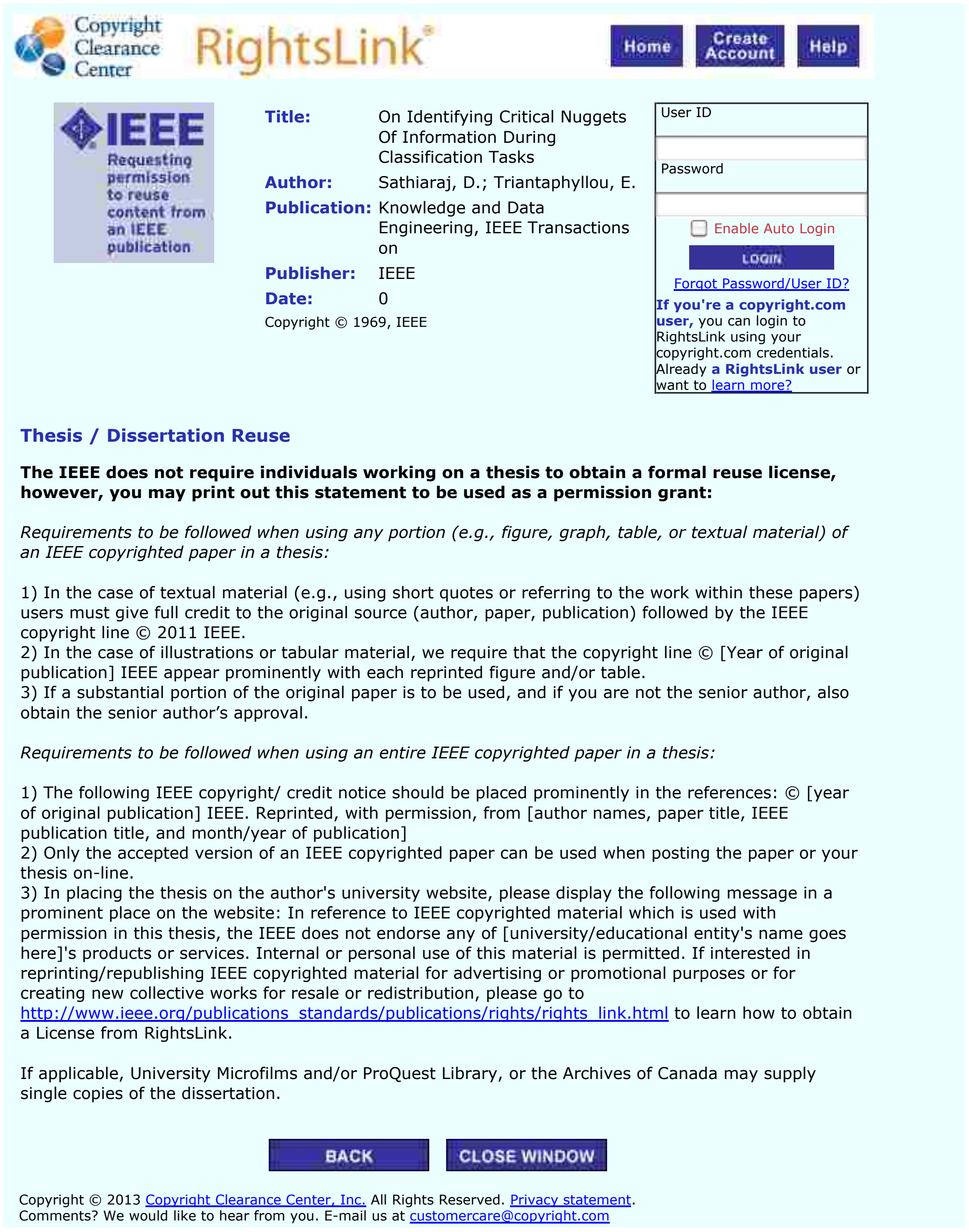
[49] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[50] Tianyi Wu, Yuguo Chen, and Jiawei Han. Re-examination of interestingness measures in pattern mining: a unified framework. *Data Min. Knowl. Discov.*, 21(3):371–397, 2010.

[51] Mao Ye, Xue Li, and Maria E. Orlowska. Projected outlier detection in high-dimensional mixed-attributes data set. *Expert Systems With Applications*, 36(3, Part 2):7104–7113, APR 2009.

[52] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*, pages 435–. IEEE Computer Society, 2003.

Appendix


Copyright Clearance Center
RightsLink
Home
Create Account
Help
IEEE
Requesting permission to reuse content from an IEEE publication

Title: On Identifying Critical Nuggets Of Information During Classification Tasks
Author: Sathiaraj, D.; Triantaphyllou, E.
Publication: Knowledge and Data Engineering, IEEE Transactions on
Publisher: IEEE
Date: 0
Copyright © 1969, IEEE

User ID
Password
Enable Auto Login
LOGIN
Forgot Password/User ID?
If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK   CLOSE WINDOW

Copyright © 2013 Copyright Clearance Center, Inc. All Rights Reserved. Privacy statement.
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Vita

David Sathiaraj was born in Hyderabad, India. He finished his undergraduate degree in mechanical (production) engineering at Osmania University, India, in August 1998. He obtained two master's degrees in industrial engineering and computer (systems) science from Louisiana State University. He is currently a candidate for the doctoral degree in computer science at Louisiana State University. For the past 10 years, he has been involved in software systems development in his role as the IT Manager for the NOAA Southern Regional Climate Center. He has been an active developer of the Applied Climate Information Systems (ACIS) project - a distributed computing system that includes one of the largest climate data warehouses in the world and a software architecture that collects, analyzes and delivers climate data products. His research interests include data mining, big data analytics, visualization, distributed computing and geographic information systems.