2013

Exploring the Learnability of Numeric Datasets

Di Lin
Louisiana State University and Agricultural and Mechanical College, fengzhongdi@gmail.com

EXPLORING THE LEARNABILITY OF NUMERIC DATASETS

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 School of Electrical Engineering and Computer Science
Computer Science and Engineering Division

by
Di Lin
B.S., FuZhou University, 2003
M.S., Louisiana State University, 2011
August 2013

Acknowledgements

Over the past six years I have received support and encouragement from a great number of individuals. Dr.Evangelos Triantaphyllou has been a mentor, colleague, and friend. His guidance has made this a thoughtful and rewarding journey. I would like to thank my dissertation committee of Dr.Jian Zhang and Dr. Jianhua Chen, for their support over the past couple years as I moved from an idea to a completed study.

Table of Contents

List of Tables

List of Figures

Abstract

When doing classification, it has often been observed that datasets may exhibit different levels of difficulty with respect to how accurately they can be classified. That is, there are some datasets which can be classified very accurately by many classification algorithms, and there also exist some other datasets that no classifier can classify them with high accuracy. Based on this observation, we try to address the following problems: a)what are the factors that make a dataset easy or difficult to be accurately classified? b) how to use such factors to predict the difficulties of unclassified datasets? and c) how to use such factors to improve classification. It turns out that the monotonic features of the datasets, along with some other closely related structural properties, play an important role in determining how difficult datasets can be accurately classified. More importantly, datasets which are comprised of highly monotonic data, can usually be classified more accurately than datasets with low monotonically distributed data. By further exploring these monotonicity based properties, we observed that datasets can always be decomposed into a family of subsets while each of them is highly monotonic locally. Moreover, it is proposed in this dissertation a methodology to use the classification models inferred from the smaller but highly monotonic subsets to construct a highly accurate classification model for the original dataset. Two groups of experiments were implemented in this dissertation. The first group of experiments were performed to discover the relationships between the data difficulty and data monotonic properties, and represent such relationships in regression models. Such models were later used to predict the classification difficulty of unclassified datasets. It seems that in more than 95% of the predictions, the deviations between the predicted value and the real difficulty are smaller than 2.4%. The second group of experiments focused on the performance of the proposed meta-learning approach. According to the experimental results, the proposed approach can consistently achieve significant improvements.

Keywords:Data mining, classification models, monotonic property, decomposition approaches, binary datasets, monotonicity, monotonic characteristics, monotone functions, regression models, data analysis.

Chapter 1

Introduction

How to efficiently and effectively analyze datasets of data grouped into classes has always been a crucial challenge in data mining research. Currently, there are numerous algorithms in use which infer classification models from such datasets. Next, these classification models may be used to infer the class values of new data points for which the class values are unknown. Such algorithms infer classification models by implementing various, and often times, quite diverse strategies. Examples include support vector machines (SVMs) [39, 7, 9], neural networks [30, 40, 47], decision trees [6, 36, 1], and logic-based approaches [43, 26, 24, 17, 13], just to name a few.

An interesting observation derived from numerous studies (see, for instance, [39, 24, 40]) is that often times some datasets may be analyzed very accurately by a wide spectrum of classifiers while other datasets may not be analyzed as easily. In other words, it seems like there are "easy" datasets, "difficult" datasets and datasets of intermediate degrees of difficulty when one is interested on how easily they can be analyzed accurately by classifiers. Therefore, in this study the difficulty of a dataset is evaluated by the average classification accuracy when it is analyzed by various classifiers, the lower the accuracy, the more difficult the dataset is. Furthermore, in this dissertation this property is also defined as the **learnability** of datasets.

This study focuses on this very issue. That is, the main research question studied here is how one can predict whether a given dataset would be analyzed accurately by a wide spectrum of classifiers. A theoretical analysis and some computational results provided in the following sections indicate that a property in data known as the monotonicity property plays a central role in determining whether a given dataset is "easy," "difficult" or of intermediate degree of difficulty when one focuses on the above classification task. More importantly, it is also observed that some monotonic based properties are strongly related to their monotonic property, and which could be used to accurately predict the learnability of datasets.

At present, if one wishes to determine whether a given dataset can be easily classified with high accuracy, then that dataset has to be analyzed by many classifiers. The inferred models are evaluated in terms of how accurate they are when they are fed with new data points of hidden class values. However, such an approach may have a number of weaknesses. First, it might be a time consuming approach, as many different classifiers need to be employed. Second, even if the results are highly conclusive at the end of such a tedious study, one still does not really know what makes a dataset easy or difficult to be classified accurately. Furthermore, if a dataset is deemed as a difficult one because a number of classifiers have difficulty inferring accurate models, does it mean that this dataset is truly a hard one? After all, which classifiers should be used in such studies? How many of them? Currently, such questions cannot be answered objectively.

Therefore, a theoretical understanding of what makes a dataset easy or hard, based on its structure alone, is of paramount importance in this area of data mining research. Furthermore, if one knows that a given dataset is a hard one because of properties pertinent to this dataset, then a new classifier which outperforms existing ones even by a few percentage points, might be considered as an important contribution. On the other hand, if a new classifier performs only slightly better when dealing with easy datasets, then such news might not be as important.

It is observed that when datasets are comprised of highly monotonic data, then they can be classified accurately by most methods while datasets that are not comprised of highly monotonic data, tend to be more difficult to be accurately analyzed by classifiers. Then the challenge becomes what happens if a given dataset does not exhibit strong monotonicity, is there any way to make it easier to be accurately classified, perhaps after some data manipulations? The present dissertation provides an answer to this very important question.

In summary, our research is concerned with the following issues: what are the factors that impact the difficulty of datasets, how to derive such factors, how to use them to evaluate the difficulty of the datasets objectively, under what conditions is the proposed approach applicable, and more importantly, how to use the previous results on poorly monotonic data, such that the classification accuracies can be improved.

The rest of the dissertation is organized as follows. Section 2 introduces the notion of monotonicity and also provides the definitions of the monotonic features considered in this study. The third section illustrates how some binary datasets were generated for this study. The fourth section demonstrates how the monotonic characteristics of datasets can be used to predict their classification difficulty. This is done for some binary datasets and some continuous datasets. The fifth section provides a way to pre-process the raw data to make them easier to accurately classify, while the sixth section shows a meta-learning approach to improve the classification on any numeric datasets. Finally, this dissertation ends with the main conclusions of our study.

Chapter 2

Introduction to Monotonicity

2.1 The Monotonic Property in Datasets

Because monotonicity plays a central role in the developments described in our research, this section presents a brief discussion of the notion of monotonicity and some key developments in this area. This discussion on monotonicity is important even for datasets that are not purely monotonic. This is true because as it is explained later, even when monotonicity is partially present in a dataset, then one may still be able to reach certain important conclusions when the learnability of that dataset is concerned.

For a simple and intuitive illustration of what is monotonicity in data, let us consider the following hypothetical situation. Suppose an analyst is interested in studying how a personal computer (PC) may crash under various software application loading scenarios. This analyst has observed that when certain software applications are loaded simultaneously, then the PC may crash. As there are n possible applications to be loaded, the state of the PC may be represented in terms of binary vectors in n dimensions. The analyst has observed that under certain loading scenarios the PC may crash (class value 1), while under some other loading scenarios the PC may not crash (class value 0).

For instance, if a word processor, a photo editor and a video editor are loaded simultaneously, then the PC may crash (class value 1). Suppose that for n=10 (i.e., there are up to 10 applications to be loaded), the previous scenario is represented by the following binary vector: V_1 = (0100101000), where the three 1s represent the loading of the previous three applications, respectively. Then one may argue, that if the previous three applications have been loaded and then two more are loaded in addition (such as the ones represented by the following vector: U_1 = (0110101010)), then the PC may crash as well (class value 1).

This is reasonable to assume because the new state of the PC describes a situation that is even more strenuous than the previous one, under which nevertheless the PC would crash. Please also observe that for these two vectors the following is true: $U_1 > V_1$. In a similar manner, if the PC does not crash under a given software loading scenario, say, the one represented by the vector V_2 = (0100001110), then most likely it will not crash under the loading scenario U_2 = (0100001010), which represents a lighter case. In other words, if a Boolean function f exhibits the following property: $f(U) \geq f(V)$, for any two vectors U and V such that $U \geq V$, then we say that the function f is monotonically increasing.

More formally, let $\{0,1\}^n$ denote the binary space defined on n Boolean attributes. A **Boolean function** defined on this space is a mapping from $\{0,1\}^n$ into {0,1}. Suppose that two binary vectors U and V from $\{0,1\}^n$ are given where $U = (u_1, u_2, u_3, \ldots, u_n)$ and $V = (v_1, v_2, v_3, \ldots, v_n)$, and u_i, v_i = 0 or 1 for any i = 1,2,3,...,n. Then, there are three possible cases as described in the following definitions.

Definition 2.1: *A binary vector $U \in \{0,1\}^n$ is said to be* ***greater than or equal to*** *another vector $V \in \{0,1\}^n$, denoted as $U \geq V$, if and only if (iff) $u_i \geq v_i$, for $i = 1,2,3,\ldots,n$, where $u_i(v_i)$ denotes the i^{th} element of vector $U(V)$.*

Definition 2.2: *A binary vector $U \in \{0,1\}^n$ is said to be* ***less than or equal to*** *another vector $V \in \{0,1\}^n$, denoted as $U \leq V$, iff $u_i \leq v_i$, for $i = 1,2,3,\ldots,n$, where $u_i(v_i)$ denotes the i^{th} element of vector $U(V)$.*

Definition 2.3: *Given two vectors $U,V \in \{0,1\}^n$, they are said to be* ***unrelated*** *to each other iff neither one is greater than or equal to the other. Otherwise, they are called* ***related****.*

If a vector U is greater than another vector V, then one can also say that **vector U proceeds vector V**, or in other words, that **vector V follows vector U**. For some illustrative examples of the above, consider the four vectors in $\{0,1\}^4$ defined as follows: V = (1011), W = (0011), P = (0001) and Q = (1001). Then according to the previous definitions, it follows that $V > W$ and $W > P$, and it is easy to get $V > P$ as well. Moreover, the vectors W and Q are unrelated. The following definition provides the notion of monotone Boolean functions formally.

Definition 2.4: *A Boolean function f defined on $\{0,1\}^n$, is called* ***monotonically increasing*** *iff $f(U) \geq f(V)$ for all $U,V \in \{0,1\}^n$, and $U \geq V$. If $f(U) \leq f(V)$ for all pairs of vectors $U,V \in \{0,1\}^n$ such that $U \geq V$, then such Boolean function is called* ***monotonically decreasing****.*

In this dissertation when we say monotone Boolean function we will mean an increasing one unless it is otherwise specified. The first known study on monotonicity is due to Dedekind [14]. Other early studies are due to Church [11] and Ward [45] where they studied the number of all Boolean monotone functions for dimensions $n = 1,2,3,\ldots,7$. An interesting development is due to Hansel [23] who introduced the notion of Hansel chains. Some early learning complexity issues for inferring monotone Boolean functions were studied in [38, 22, 19, 28, 3, 32]. More recent studies on learning monotone Boolean functions from training data can be found in [41, 43].

2.2 Key Definitions Related to Monotonicity

Given a monotone Boolean function f from $\{0,1\}^n$ into $\{0,1\}$, some vectors in $\{0,1\}^n$ play a critical role in defining this function f. These are what is known as upper zero and lower unit vectors [23, 41, 43]:

Definition 2.5: *A vector V^* is called a* ***lower unit of a monotone Boolean function*** *f iff $f(V^*)$ = 1 and $f(V) < f(V^*)$, for any vector $V < V^*$.*

Definition 2.6: *A vector V^* is called an* ***upper zero of a monotone Boolean function*** *f iff $f(V^*)$ = 0 and $f(V) > f(V^*)$, for any vector $V > V^*$.*

Definition 2.7: *Given a monotone Boolean function$f$ the union of the set of the lower units with the set of the upper zeros is the set of its* ***border vectors****.*

For any monotone Boolean function f, the set of all lower units and the set of all upper zeros are unique, and either one of these two sets uniquely identifies f [41].

2.2.1 Purely Monotonic Binary Datasets

Next, it is assumed that given is a binary dataset, say D, comprised of two mutually exclusive and exhaustive subsets of binary vectors classified by some monotone Boolean function f. The analyst may or may not know the definition of this function f. The first subset, denoted as D^+, has all the vectors classified as positive, while the second subset, denoted as D^-, has all the vectors classified as negative. Obviously, $D = D^+ \cup D^-$.

Definition 2.8: *A binary dataset D is a* ***purely monotonic binary dataset*** *iff any positive vector $V \in D^+$ is either greater than any negative vector $U \in D^-$ or the vectors V and U are unrelated.*

If the condition described in the previous definition is not satisfied, the dataset is called a **non-purely monotonic binary dataset**. Next, the previous concepts of upper zeros, lower units, and border points of monotone Boolean functions can be easily adapted in the context of a purely monotonic binary dataset D. This is accomplished in terms of the following definitions:

Definition 2.9: *Given a purely monotonic binary dataset D, a vector $V^* \in D^+$ is called a* ***lower unit of the dataset*** *D iff for any $V \in D^+$, $V^* < V$ or the vectors V^* and V are unrelated.*

Definition 2.10: *Given a purely monotonic binary dataset D, a vector $V^* \in D^-$ is called an* ***upper zero of the dataset*** *D iff for any $V \in D^-$, $V^* > V$ or the vectors V^* and V are unrelated.*

Definition 2.11: *Given a purely monotonic binary dataset D, the union of the set of its lower units with the set of its upper zeros is the set of its* ***border points****.*

The previous concepts can be easily extended to functions and datasets that are not binary. This can happen if the attributes take on ordinal values and thus one can compare vectors as was the case with only binary attributes. Algorithm 1 can be used to identify the border points of a purely monotonic dataset (no necessarily only binary). As illustrative examples of these concepts, consider the dataset in $\{0,1\}^4$ depicted in Table 2.1. The lower units of D are the vectors: $\{$(1100), (1010)$\}$, while the upper zeros of D are the vectors: $\{$(0110), (1001), (0101)$\}$. The set of its border points is the union of the previous two sets.

TABLE 2.1: An example of a purely monotonic dataset D in $\{0,1\}^4$.

Positive vectors	(1111), (1110), (1101), (1011), (1100), (1010)
Negative vectors	(0110), (1001), (0101), (0100), (0001), (0000)

Algorithm 1: Find all the border points in a purely monotonic dataset D.

```
Input  : $E^+$, $E^-$; /*Two mutually exclusive sets of positive and negative vectors in $D$,
         respectively.*/
Output: LU,UZ; /*Two sets of the lower units and the upper zeros, respectively.*/
for each $e_i^+ \in E^+$ do
    for each $e_j^+ \in E^+$ do
        if $e_j^+ > e_i^+$ then
            Remove $e_j^+$ from $E^+$;
        end
    end
end
for each $e_i^- \in E^-$ do
    for each $e_j^- \in E^-$ do
        if $e_j^- < e_i^-$ then
            Remove $e_j^-$ from $E^-$;
        end
    end
end
$LU \leftarrow E^+$ and $UZ \leftarrow E^-$;
Return LU, UZ;
```

2.2.2 Non-purely Monotonic Binary Datasets

Some real-life datasets, however, may not be purely monotonic. That is, in such datasets one may encounter positive vectors which may be less than some negative vectors. When this happens, then the previous defined concepts should be modified to become the extended lower units, the extended upper zeros, and the extended border points of the dataset.

To be more specific, if a binary dataset is not purely monotonic then it can be decomposed into groups of vectors such that within each pair of groups the monotonic property holds locally. In particular, such dataset can be divided into several unique class groups. Each group contains the vectors with same class value, while the class values between adjacent groups are different. Moreover, for any two groups of such vectors with different class values, all the vectors in one group precede or are unrelated to the vectors in the other group. In this way, any two groups of vectors of opposite class values can comprise a purely monotonic binary dataset.

Suppose that given is a binary dataset *D* which is not purely monotonic. Then, its sub-groups can be derived as follows:

1. Find out all the positive vectors in *D* which precede or are unrelated to all negative vectors. Next, they are removed from *D* to form a positive sub-group.

2. From the remaining vectors, find out all the negative vectors which precede or are unrelated to the rest of the positive vectors. Next, they are removed from *D* to form a negative sub-group.

3. Repeat steps (1) and (2) until the dataset *D* becomes empty.

Based on such unique class groups, the concept of extended border points can be defined as follows:

Definition 2.12: *Given a positive sub-group G, a vector $V^* \in G$ is called an* ***extended lower unit of the sub-group*** *G iff for any $V \in G$, then $V^* < V$ or the vectors V^* and V are unrelated.*

Definition 2.13: *Given a negative sub-group G, a vector $V^* \in G$ is called an* ***extended upper zero of the sub-group*** *G iff for any $V \in G$, then $V^* > V$ or the vectors V^* and V are unrelated.*

Definition 2.14: *Given a non-purely monotonic binary dataset D, the union of the set of its extended lower units with the set of its extended upper zeros forms the set of its* ***extended border points****.*

Once the sub-groups are determined as above, the border points of each sub-group can be determined by implementing Algorithm 1 within each sub-group. As was the case with the determination of the (regular) lower unit and upper zero vectors, an algorithm can be easily designed to determine the extended lower unit and extended upper zero vectors. Algorithm 2 shows such an approach with a time complexity of O(m^2), where *m* is the number of vectors in the training dataset.

Algorithm 2: General approach to find extended border points in a non-purely monotonic dataset.

```
Input : $E^+$, $E^-$; /*Two mutually exclusive sets of positive and negative vectors,
        respectively.*/
Output: ELU,EUZ; /*Two sets of the extended lower units and the extended upper zeros,
        respectively.*/
$ELU \leftarrow \phi, EUZ \leftarrow \phi$;
while $E^+ \neq \phi$ or $E^- \neq \phi$ do
    $Subset^+ \leftarrow \phi, Subset^- \leftarrow \phi$;
    for each $e_i^+ \in E^+$ do
        if There is no $e_j^- \in E^-$ such that $e_j^- > e_i^+$ then
            $Subset^+ \leftarrow Subset^+ \cup e_i^+$;
            Remove $e_i^+$ from $E^+$;
        end
    end
    for each $e_i^- \in E^-$ do
        if There is no $e_j^+ \in E^+$ such that $e_j^+ > e_i^-$ then
            $Subset^- \leftarrow Subset^- \cup e_i^-$;
            Remove $e_i^-$ from $E^-$;
        end
    end
    $ELU \leftarrow ELU \cup findLowerUnits(Subset^+)$; /* Apply Algorithm 1 */
    $EUZ \leftarrow EUZ \cup findUpperZeros(Subset^-)$; /* Apply Algorithm 1 */
end
Return ELU, EUZ;
```

As illustrative examples of the concepts of extended lower units, extended upper zeros, and extended border points, one can consider the dataset in $\{0,1\}^5$ depicted in Table 2.2. The extended lower units of D are the vectors:{(11001), (10111), (00001), (01100), (10110)}, while the extended upper zeros of D are the vectors:{(11110), (01101), (10011), (10010), (00110)}. The set of its border points is the union of the previous two sets.

TABLE 2.2: An example of a non-purely monotonic dataset D in $\{0,1\}^5$.

Positive vectors	(11111), (11101), (10111), (11001), (11100), (10110), (01001), (01100), (00001)
Negative vectors	(11110), (01101), (10011), (00110), (00101), (10010), (10000), (00100)

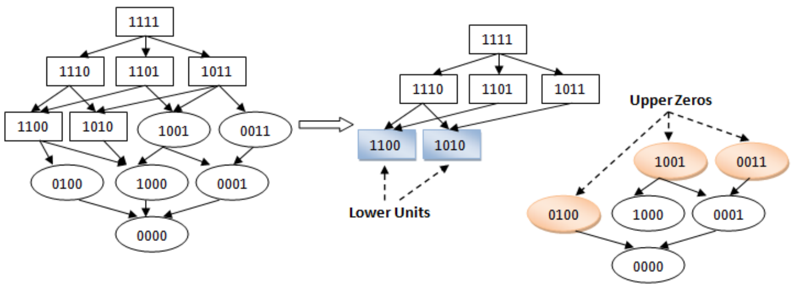

FIGURE 2.1: The poset for the dataset in Table 2.1 and its border points.

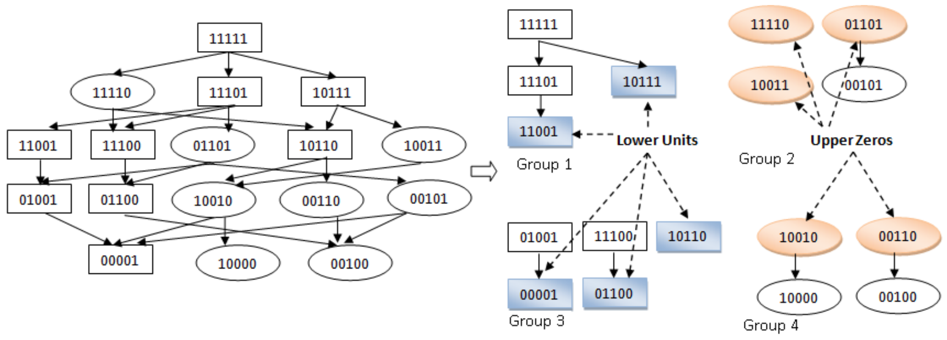

FIGURE 2.2: The poset for the dataset in Table 2.2 and its border points.

2.3 Graphical Representation in Two Dimensions

The previous concepts of the various types of border vectors become easier to comprehend if one considers them in the context of a 2-dimensional partially ordered set or poset (see, for instance, [41, 43]). Such a poset is defined as follows:

Definition 2.15: *Given a dataset D, its* ***two dimensional poset representation*** *is a graph whose nodes correspond one-to-one to vectors of D. Furthermore, there is an arrow from a node U to a node V, iff $U > V$.*

When a poset representation is used, the data depicted in Table 2.1 correspond to Figure 2.1, while the data in Table 2.2 correspond to Figure 2.2. For the sake of simplicity of the presentation, arrows are shown only for adjacent nodes. One may observe that these graphs can be organized in terms of layers as shown in these figures. The corresponding border vectors (upper zeros and lower units) are also shown in these two figures.

2.4 Data with both Positive and Negative Attributes

In the previous considerations it was assumed that all the binary attributes are *positive*. That is, if an attribute has the value 1, then somehow this contributes to the chances for a given vector to be of the positive class value. For instance, in the earlier PC related example, if a particular software item is loaded (i.e., the attribute that corresponds to the loaded/not-loaded state for that software item has value 1), then this contributes to the chances for a vector to be of the positive class value (i.e., the "PC crashes" class value).

Although this may not be as critical for the binary case (as a given binary attribute may be assigned value 1 or 0 depending on how one defines them), it becomes critical when one considers the case when attributes have continuous values. For instance, in a setting with continuous attribute values one may wish to study, say, the performance of a car engine. Then, one may have two mutually exclusive and exhaustive states as follows: The engine needs immediate maintenance (this is the "positive" class or class value 1) or the engine does not need immediate maintenance (this is the "negative" class or class value 0).

In this hypothetical example an attribute that expresses the noise level of the engine could be a **positive attribute**. This is true because the higher the noise level is, the more likely is that the engine needs immediate maintenance (i.e., it is of the positive class). On the other hand, the number of miles per gallon of fuel could be a **negative attribute** as the higher the value of that attribute is, the less likely is that the engine to need immediate maintenance.

It is important to state here that in some real-life applications, attributes may not be purely positive or negative. That is, as the value of such attributes increases, it is possible to have certain levels beyond which the chances to be in one class value versus another, may alternate.

For instance, consider a healthy life-style study where among other attributes one of the attributes is the amount of daily exercise a person may pursue. For simplicity, assume that the class values are "healthy life-style" (the positive class) and "not healthy life-style" (the negative class). As the amount of exercise increases, say, from 20 minutes per day to 40 minutes, then to 60 minutes and so on, the chances that we will be in the positive class increase accordingly. However, there is some value for this particular attribute where more increase to its value may not lead to higher chances to be in the positive class. For instance, if one exercises at abusive levels, say 12 or even 14 hours a day, then that may cause some health related problems. For the previous reasons, in this study positive and negative attributes are defined in a non-rigorous manner as follows:

Definition 2.16: *An attribute with* ***binary values*** *is called a* ***positive attribute*** *if when it has value 1, then it is more likely for vectors to be in the positive class. Otherwise, it is called a* ***negative attribute****.*

Definition 2.17: *An attribute with* ***ordinal values*** *is called a* ***positive attribute****, if when the values increase, then it is more likely for vectors to be in the positive class. Otherwise, it is called a* ***negative attribute****.*

The following sections will distinguish between positive and negative attributes only. Cases like the one described above, may lead to situations where datasets are not purely monotonic. As it was explained earlier, for non-purely monotonic binary datasets one can decompose them into regions where monotonicity holds locally. Obviously, a dataset with lots of such regions is less overall non-purely monotonic than a dataset with just a few such regions. How this factor and other ones related to monotonicity may impact the learnability of datasets is studied in the following sections.

2.4.1 Identifying the Positive and Negative Attributes

From the previous discussion it follows that there are alternative ways to define what is positive and negative attributes. Thus, there are alternative ways to quantify the way how to determine such attributes. In this study the following approach is used.

Each pair of positive-negative vectors is considered. For each attribute i, let $N_{p,i}$ indicate the number of times attribute i in the positive vectors is greater than the same attribute i in the negative vectors when all pairs are considered. Similarly, let $N_{q,i}$ indicate the number of times attribute i in the negative vectors is greater than the same attribute i in the positive vectors. Then, if $N_{p,i} > N_{q,i}$, the attribute i is assumed to be a positive attribute. If $N_{p,i} < N_{q,i}$, it is assumed to be a negative attribute. Finally, if $N_{p,i} = N_{q,i}$, then attribute i is designated as either positive or negative with probability 50%. One may observe here that as more training data are added to a given dataset, the groups of positive and negative attributes may change.

2.4.2 Comparing Vectors with Both Positive and Negative Attributes

Once the positive and the negative attributes have been determined, two n-dimensional vectors V and W (with binary and/or ordinal attributes) can be defined as follows: $V = ((v_{p1}, v_{p2}, v_{p3}, \ldots, v_{px}), (v_{n1}, v_{n2}, v_{n3}, \ldots, v_{ny}))$, and $W =((w_{p1}, w_{p2}, w_{p3}, \ldots, w_{px}), (w_{n1}, w_{n2}, w_{n3}, \ldots, w_{ny}))$, where $n = x+y$. The elements v_{pi} and w_{pi} indicate the positive attributes of the vectors V and W, respectively, while v_{qj} and w_{qj} indicate the negative attributes of them. In light of this enhancement, the previous definitions of vectors defined on only positive attributes can be expanded accordingly.

In the new setting a vector V is said to be **greater than or equal to (i.e., it precedes)** vector W (denoted as $V \succeq W$) iff $v_{pi} \geq w_{pi}$, for $i = 1,2,3,\ldots,x$, and $v_{ni} \leq w_{ni}$, for $i = 1,2,3,\ldots,y$. In this case one may also say that vector W is **less than or equal to (i.e., it follows)** vector V (denoted as $W \preceq V$). As before, when any of these two ordering relationships can be defined between two vectors, we say that such vectors are **related**. Otherwise, they are called **unrelated**. Please note that now the symbols $\succeq$ and $\preceq$ are used instead of the previous $\geq$ and $\leq$ symbols, respectively.

For any pair of vectors V and W defined as above, if either their positive groups or negative groups are unrelated, then the vectors V and W are unrelated too. Furthermore, the vectors V and W are unrelated when they have different number of attributes in either their positive or negative group. It should also be noted here that if both the positive and negative groups of vector V are at the same time greater than or smaller than those of vector W, then the vectors V and W are still unrelated.

For some illustrative examples of the above, consider the four binary vectors defined as follows: V = ((010), (111)), W = ((110), (011)), P = ((100), (010)) and Q = ((011), (110)). Then according to the previous definitions, it follows that $W \succeq V$ and $Q \succeq V$. All other pairs are comprised of unrelated vectors.

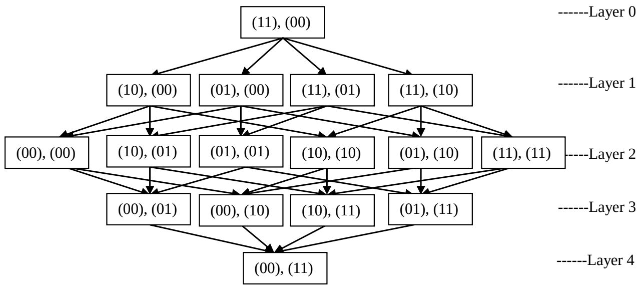

FIGURE 2.3: The complete poset when *n*=4, *x*=2, and *y*=2.

2.5 Graphical Representation for Datasets With Positive and Negative Attributes

As was the case with binary data that have only positive attributes, the enhanced type of binary data which are defined on both positive and negative binary attributes can be represented graphically in terms of posets as well. Figure 2.3 depicts the poset for the complete 4-attribute binary dataset which has two positive attributes and two negative attributes. It is an illustrative example of constructing a poset for such kind of 4-attribute binary dataset. Any 4-attribute binary dataset with two positive and two negative attributes can be defined in this format, but with less than 16 members is a subset of the one displayed in Figure 2.3. Then, such a binary dataset can be graphically represented accordingly, and this idea can be expanded to display any dataset (i.e., not only binary) with ordinal attributes in two dimensions, provided that the number of vectors is manageable.

The earlier definitions of lower units, upper zeros, border points of purely monotonic datasets still hold as long one defines the ordering relations between pairs of vectors in terms of the two groups of positive and negative attributes. Table 2.3 presents a simple 5-dimensional dataset with purely monotonic data which is defined on three positive attributes and two negative attributes. The same data are also depicted in Figure 2.4 in terms of a poset, and along with the corresponding lower unit and upper zero vectors.

Finally, Table 2.4 presents a dataset which is not purely monotonic and it is defined on three positive attributes and two negative attributes. The corresponding poset representation and a graphical depiction of its lower units and upper zeros are given in Figure 2.5.

TABLE 2.3: An example of a purely monotonic binary training dataset when positive/negative attributes are taken into consideration.

Positive vectors	((111),(00)), ((111),(01)), ((111),(10)), ((110),(10)), ((111),(11)) ((011),(10)), ((011),(11))
Negative vectors	((101),(00)), ((101),(01)), ((100),(00)), ((010),(10)), ((100),(01)) ((001),(01)), ((000),(10)), ((100),(11)), ((001),(11))

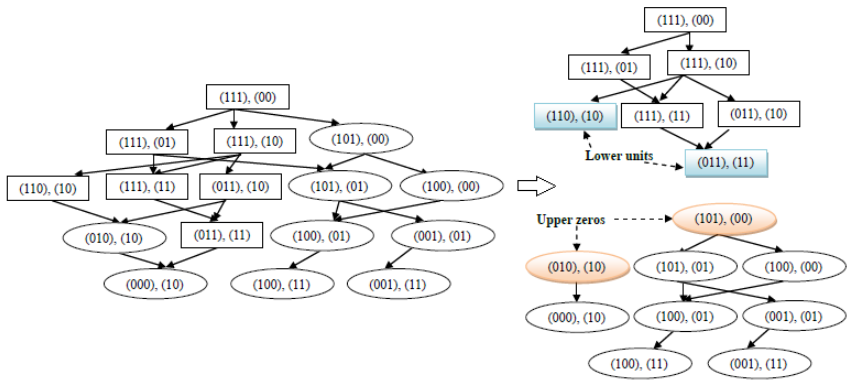

FIGURE 2.4: The poset for the dataset listed in Table 2.3 and its border points.

TABLE 2.4: An example of a non-purely monotonic binary training dataset when positive/negative attributes are taken into consideration.

Positive vectors	((111),(00)), ((111),(10)), ((110),(10)), ((101),(00)), ((010),(10)) ((000),(10)), ((111),(11)), ((011),(11)), ((101),(01))
Negative vectors	((111),(01)), ((100),(00)), ((011),(10)), ((001),(10)), ((100),(01)) ((100),(11)), ((001),(01)), ((001),(11))

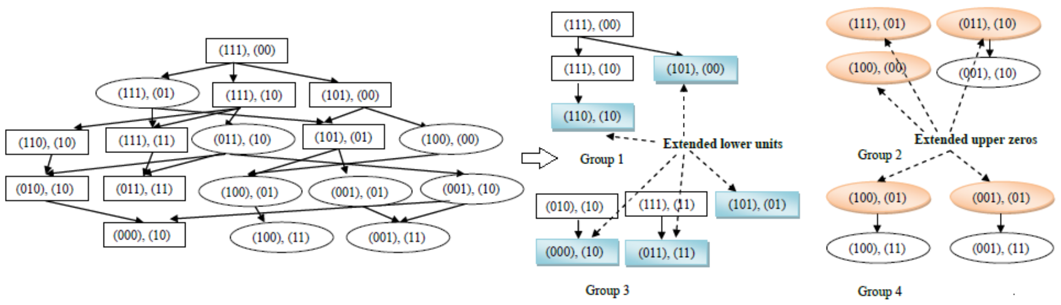

FIGURE 2.5: The poset for the dataset listed in Table 2.4 and its border points.

By following the approach described above, the dataset listed in Table 2.4 can be divided into four sub-groups, as shown in Figure 2.5. Based on these sub-groups, the extended lower units are {((110),(10)), ((101),(00)), ((000),(10)), ((011),(11)), ((101),(01))}, and the extended upper zeros are {((111),(01)), ((100),(00)), ((011),(10)), ((100),(01)), ((001),(01))}.

2.6 Types of Pairs of Vectors

The previous sections formally introduced some key relationships between any two vectors by mostly ignoring their class values. That is, two vectors are either related to each other or are unrelated. This section discusses how two vectors can be related by also considering their class values.

Let us assume that each vector in a dataset is classified to be either positive or negative. By considering such class values, there are five possible relationships between the two vectors in any pair of such vectors. These relationships are: a positive vector V precedes another positive vector W; a positive vector V precedes a negative vector W; a negative vector V precedes a positive vector W; a negative vector V precedes a negative vector W; or the vectors V and W are unrelated.

For any pair of vectors V and W, if the vectors are related and they have the same class value, they are said to comprise a pair which is in agreement with the monotonic property, or an **AMP pair**. In the situation when a negative vector V precedes a positive vector W (i.e., $V \succeq W$), the pair comprised in this order is in conflict with the monotonic property (or it is a **CMP pair**).

For a pair of vectors V and W where a positive vector V precedes a negative vector W, their class values do no conflict with the monotonic property. This indicates that if one has $V \succeq W$ and knows that vector V has been classified as positive, then the vector W can be either positive or negative without conflicting with the monotonic property. A similar observation follows if $V \succeq W$ and the vector W has been classified as negative. Then the vector V can be either positive or negative.

Given the above discussion, from now on two monotonically related vectors V and W of the same class value will be said to comprise a pair which is in agreement with the monotonic property of type 1, or an **AMP1 pair**. For another pair of vectors V and W, where $V \succeq W$ and the class value of V is positive while the class value of W is negative, it is called a pair in agreement with the monotonic property of type 2, or an **AMP2 pair**. If the vectors V and W are unrelated, then the class value of one vector has no effect on that of the other. In this case, they comprise a monotonically neutral pair (or an **MNP pair**). As it is shown in the experimental section, these types of pairs of vectors play a crucial role in determining how easily a dataset can be accurately analyzed by a large spectrum of classifiers.

More formally, suppose that given are two distinct vectors V and W defined on n attributes (binary or ordinal in general) and have the same positive and negative attributes. It is also assumed that there are only two classes; the positive and the negative. Next, all the possibilities of the relative relations between any two vectors V and W and their class values are formally introduced as follows:

Definition 2.18: *Two numeric vectors V and W, where $V \succeq W$, constitute a pair which is in agreement with the monotonic property of type 1, denoted as an **AMP1 pair**, iff they have the same class value.*

Definition 2.19: *Two numeric vectors V and W, where $V \succeq W$, constitute a pair which is in agreement with the monotonic property of type 2, denoted as an **AMP2 pair**, iff the class of vector V is positive while the class of vector W is negative.*

Definition 2.20: *Two numeric vectors V and W, where $V \succeq W$, constitute a pair which is in conflict with the monotonic property, denoted as a **CMP pair**, iff the class value of V is negative, while the class value of vector W is positive.*

Definition 2.21: *Two numeric vectors V and W form a monotonically neutral pair, denoted as an **MNP pair**, iff they are unrelated.*

Figure 2.6 depicts a collection of positive and a collection of negative training examples when $n = 4$, $x = 2$, and $y = 2$. This training dataset is comprised of four positive vectors (denoted with squared shapes) and three negative ones (denoted with oval shapes), and it is a subset of the dataset decipted in Figure 2.3. Then, according to the definitions given above, some examples of AMP1 (ordered) pairs are as follows:{((11),(00)), ((11),(01))}, {((11),(00)), ((11),(10))} and

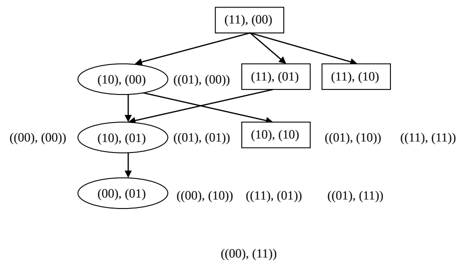

FIGURE 2.6: Some examples of different types of monotonically related pairs.

{((10),(00)), ((10,(01))}. Some examples of AMP2 (ordered) pairs are {((11),(01)), ((10),(01))} and {((11),(00)), ((10),(00))}. The only example of a CMP (ordered) pair is {((10),(00)), ((10),(10))}. Some examples of MNP pairs are {((11),(10)), ((10),(01))} and {((10),(00)), ((11),(01))}.

Chapter 3

The Experimental Design for Binary Datasets

This section explores the roles of some potentially important characteristics of numeric datasets, which could be used to predict how difficult it is to analyze a given dataset accurately. As it was stated earlier, the main research hypothesis is that such difficulty is primarily related to the monotonic characteristics of a dataset, even if monotonicity occurs partially (i.e., when there are CMP pairs in the datasets).

3.1 Design Issues of Experiments with Some Artificial Binary Datasets

The binary datasets used in this set of experiments were created in different sizes with the value of *n* (dimensions) ranging from 6 to 60. In each individual experiment, the training data and the testing data are binary datasets with the same number of dimensions (attributes).

As it was stated earlier, this study defines the "difficulty" or learnability of a dataset as how difficult it is to be accurately classified. This is indicated by the average accuracy when it is analyzed by multiple classifiers. The lower the value, the more difficult it is. Moreover, it is assumed that the false positive and the false negative errors are of the same penalty cost. The purpose of the experiments is to generate accurate regression models, which use the monotonic characteristics of the training datasets as the independent variables to formulate their difficulties.

In order to do that, a group of artificial binary datasets were studied in this family of experiments. This group was comprised of "easy" datasets, "difficult" datasets, and datasets with intermediate degrees of difficulty. Four kinds of binary datasets were therefore generated with different monotonic characteristics as follows:

1. Random datasets. Such datasets contain vectors with randomly assigned class values. In general, they are expected to have intermediate degrees of difficulty.

2. Datasets without CMP pairs. They are expected to be easy to classify accurately.

3. Datasets which have lots of CMP pairs. This is the opposite case to the one described above. Such datasets are expected to be difficult to classify accurately.

4. Pairs of datasets which have exactly the same border points but have very different numbers of AMP1 pairs. It is expected that the ones with more AMP1 pairs are easier to classify accurately.

3.2 Generating the Binary Experimental Datasets

3.2.1 Generating Random Datasets

The random datasets were generated in a straightforward way. Given are the number of attributes n and the number of selections K. Next, K vectors were selected with replacement with equal probability (i.e., equal to $1/2^n$). Any duplicate vectors were deleted to make sure that each vector appears only once in the generated dataset. The number of distinct vectors among the K ones is denoted as N (i.e., $K \geq N$). After that, every vector was assigned to class values 1 or 0 (for positive or negative, respectively) with probability equal to 0.50. The positive/negative attributes of these datasets are unknown, but they can be later determined by the way described in Section 2.4.1. It should be stated here that in this way datasets may not be generated completely randomly as some bias may be present [42].

3.2.2 Generating Datasets Which do not Contain Any CMP Pairs

In order to generate such datasets, one should first decide on the numbers of positive attributes (x) and negative attributes (y), such that $x + y = n$ and $x, y \geq 0$. The vectors in these datasets have n attributes while the first x attributes are considered as positive, and the next y attributes as negative.

After that, a complete (i.e, one of size 2^n) n-attribute binary dataset is represented as a 2-D poset, and it is divided into three groups: the positive group, the negative group and the unlabeled group. This is done by determining the values of K_1, K_2 and K_3, such that $K_1 + K_2 + K_3 = n + 1$ and $K_1, K_2, K_3 \geq 0$. The top K_1 layers comprise the positive group; the vectors located in these layers are labeled as positive. The bottom K_3 layers comprise the negative group; all the vectors located in these layers are labeled as negative. The vectors located in the middle layers are unlabeled.

The following describes how to generate an n-attribute binary dataset with N vectors for given K_1, K_2, K_3, x, and y values. It makes sense that by increasing the value of K_1 and decreasing the value of K_3, there should be more positive vectors in the generated dataset, and vice versa.

1. Create a complete n-attribute binary dataset D. This dataset should contain 2^n different binary vectors.

2. Randomly select a vector V from D without replacement. Determine the layer i in which it is located by using $i = x - P_1 + y - P_2$, where P_1 is the number of the positive attributes in V that are set to value "1" and P_2 is the number of the negative attributes in V that are set to value "0", then do the following:

   (a) If $i \leq K_1$, it is a positive vector.

   (b) If $i > K_1 + K_2$, it is a negative vector.

   (c) If $K_1 < i \leq K_1 + K_2$, then it can be randomly assigned to either the positive or the negative class with probability equal to 0.50. However, one needs to check if this vector violates monotonicity. For example, if at first a vector is randomly assigned to the negative class, and it precedes some positive vectors generated during previous iterations, then it should be changed to positive to maintain monotonicity.

3. Repeat Step 2) N times to generate N distinct vectors.

3.2.3 Generating Datasets Which Contain Many CMP Pairs

This case is the opposite of the previous one. The challenge is, given the same parameters n, x, y, and N as defined in the previous section, how to assign the class values to these N binary vectors such that the generated dataset has the maximum (or at least a very high) number of CMP pairs?

Please recall that two monotonically related positive-negative vectors comprise either an AMP2 pair or a CMP pair. However, according to the way one constructs the 2-D poset, the number of

AMP2 pairs should be always larger than that of the CMP pairs. Therefore, the number of CMP pairs should be just less than the number of AMP pairs.

One solution to achieve this goal is to first consider the complete n-attribute binary dataset in its 2-D poset, and label the vectors according to the layers they belong to. More specifically, the vectors in the same layer have the same class value, and their class values are different than those of vectors located at adjacent layers. That is, the top layer, which is comprised of the single vector that has all positive attributes set to "1" and all negative attributes set to "0", is assigned to the positive class. The second layer is assigned to negative, the third layer to positive, the fourth layer to negative, and so on.

Some experiments were performed to evaluate this method, and Table 3.1 shows some of the experimental results. One can observe from this table that by applying this method, an n-attribute complete binary dataset can have a very similar number of AMP2 and CMP pairs.

Therefore, by using given values for n, x and y, the aim is to have a random dataset with N vectors (where $N \leq 2^n$) such that it has a very large (but not necessarily the maximum) number of CMP pairs. A way to achieve this goal is to first consider the complete case (that is, the case with all 2^n vectors). Next, each vector is assigned to a class value in the way mentioned above. Finally, randomly select N distinct vectors from this complete set. This is how the third group of datasets were generated for this computational study.

It should be stated here that depending on different selections of vectors, an attribute may be recognized as a positive one in some derived datasets, but be considered as a negative one in some other derived datasets, according to the way described in Section 2.4.1.

3.2.4 Generating Datasets Which Have the Same Border Points but Very Different Monotonic Characteristics

The fourth group of datasets studies the roles CMP and AMP1 pairs play in the learnability of datasets. This group of datasets considered triplets of datasets as follows. First, a dataset was generated randomly as described in Section 3.2.1. Such a dataset may or may not have CMP pairs.

TABLE 3.1: The number of AMP2 and CMP pairs in the complete n-attribute binary datasets that are generated by the approach dessribed in Section 3.2.3.

Number of Attributes	Number of AMP2 pairs	Number of CMP pairs	Ratio
n= 5	61	60	1.017
n= 6	182	182	= 1
n=10	14,762	14,762	= 1
n=15	3,587,227	3,587,226	≈ 1
n=20	871,696,100	871,696,100	= 1
n=25	2.118×10^{11}	2.118×10^{11}	= 1
n=30	5.147×10^{13}	5.147×10^{13}	= 1
n=40	3.039×10^{18}	3.039×10^{18}	= 1
n=50	1.794×10^{23}	1.794×10^{23}	= 1
n=60	1.060×10^{28}	1.060×10^{28}	= 1

Based on this dataset, two extreme cases are introduced. The first extreme case is to build a dataset which has the same (extended) border points, the same number of positive and negative vectors but has the highest possible number of AMP1 pairs. This is denoted as **Type I extreme case**. The second extreme case is similar to the first one but the interest now is to build a dataset with the smallest possible number of AMP1 pairs. This is denoted as **Type II extreme case**.

Suppose a given dataset is defined on n binary attributes with x positive attributes and y negative attributes, it has N_1 positive vectors, N_2 negative vectors, and the sizes of the LU and UZ sets are equal to S_1 and S_2, respectively. First, let us assume that it contains no CMP pairs, and its border points (i.e., all the LU and UZ vectors) can therefore be determined by using Algorithm 1. Based on this information, two datasets are generated in order to represent the previous two extreme cases as far as the number of AMP1 pairs is concerned.

To begin with, one needs to first remove all the vectors from the dataset except the border points, and the same number of vectors will be regenerated to be *covered* by these border points. For the case of the LUs, a vector is said to be "covered" by a lower unit if and only if it precedes this lower unit. For the case of the UZs, a vector is said to be "covered" by an upper zero if and only if it follows this upper zero. For instance, if the vector ((010), (11)) is an LU, then it covers the vector ((110), (11)).

Algorithm 3: Generate an extreme case for a purely monotonic dataset D.

```
Input  : A purely monotonic dataset $D$.
Output: $D\_Extreme$. /* The extreme case of $D$ */
1  $Pos \leftarrow \phi$, $Neg \leftarrow \phi$, $D\_Extreme \leftarrow \phi$, $LU \leftarrow$ Lower units of $D$, $UZ \leftarrow$ Upper zeros of $D$;
2  $N_1 \leftarrow$ Number of positive vectors in $D$, $N_2 \leftarrow$ Number of negative vectors in $D$;
3  $S_1 \leftarrow$ Number of vectors in $LU$, $S_2 \leftarrow$ Number of vectors in $UZ$;
4  $n \leftarrow$ Number of attributes in $D$; $E^n \leftarrow$ Complete $n$-attribute binary dataset;
5  for Every vector $V \in E^n$ except the LU and UZ do
6      if $V$ is covered by some members in $LU$ then
7          $Pos \leftarrow Pos \cup V$;
8      end
9      if $V$ is covered by some members in $UZ$ then
10         $Neg \leftarrow Neg \cup V$;
11     end
12 end
13 Sort the vectors in $Pos$ in descending order by the number of $LU$ members each vector is
   covered;
14 $D\_Extreme \leftarrow D\_Extreme \cup \{$The first $N_1 - S_1$ vectors in $Pos\} \cup LU$;
15 Sort the vectors in $Neg$ in descending order by the number of $UZ$ members each vector is
   covered;
16 $D\_Extreme \leftarrow D\_Extreme \cup \{$The first $N_2 - S_2$ vectors in $Neg\} \cup UZ$;
17 Return $D\_Extreme$ ; /* This is for Type I extreme case. */
```

Every vector except the border points in E^n (i.e., the binary space of dimension n) is examined to see by how many LUs it is covered. Next,these vectors are ranked in *descending* order. Finally, the top $N_1 - S_1$ vectors together with the LUs, are introduced as the new positive vectors.

The negative examples are obtained in an analogous manner. All the vectors except the border points in E^n are ranked in descending order by how many UZs they are covered by. The top $N_2 - S_2$ vectors and all the UZs are introduced as the new negative examples. This is how the Type I extreme case is generated from a random dataset which has no CMP pairs. This new dataset has many AMP1 pairs because of the way the new positive and negative points are determined.

The Type II extreme case of a given dataset with no CMP pairs can be derived in a similar manner as the previous one. However, in this case the vectors are organized in *ascending* order by how many border points they are covered by. Now this dataset has positive and negative vectors which are covered by the least number of border points in the LU and UZ sets. This is how two new

Algorithm 4: Generate an extreme case for a dataset D which has some CMP pairs.

```
Input   : A dataset $D$.
Output: $D\_Extreme$. /* The extreme case of $D$, which now has some CMP pairs */
1  $CMP\_Vectors \leftarrow \phi, Monotonic\_Vectors \leftarrow \phi$;
2  for Every vector $V_i \in D$ do
3      if $V_i$ is found in CMP pairs then
4          $CMP\_Vectors \leftarrow CMP\_Vectors \cup V_i$;
5      end
6  end
7  $Monotonic\_Vectors \leftarrow D - CMP\_Vectors$;
8  Use Algorithm 3 (choose different sorting orders to get different extreme types) to derive a
   dataset $D_1$ from $Monotonic\_Vectors$;
9  $D\_Extreme \leftarrow CMP\_Vectors \cup D_1$;
10 Return $D\_Extreme$ ;
```

datasets are derived from a single random dataset without CMP pairs. The previous two approaches are summarized as Algorithms 3. In this algorithm, if one sorts out the vectors in descending order (as is currently shown in Algorithms 3), then the Type I extreme case dataset will be derived as dataset $D_{extreme}$. If however, the vectors are sorted out instead in ascending order, then the dataset $D_{extreme}$ will represent the Type II extreme case of the the original dataset D.

Next, let us assume that a given dataset D contains some CMP pairs. By removing all its vectors that are observed in the CMP pairs, it becomes a dataset with less number of observations and which has no CMP pairs. This is exactly the same type of dataset discussed in the previous case. Thus, from this reduced dataset one can derive two new datasets by implementing Algorithm 3 (with different sorting orders). The new generated datasets, together with the vectors that were removed earlier, form the two extreme cases of the dataset D. This approach is described as Algorithm 4.

3.3 Classifiers from WEKA

The classifiers used in these experiments are: a decision tree (J48) [6], a Bayes network [12], a naive Bayes [29], a logistic regression [18], an RBF network [25], a Kstar [27], an LWL [10], an LBK [10], an AdaBoost [21], a Multi-Boost [8], a VFI (Voting Feature Intervals) [15], an

ADTree (Alternating Decision Trees) [20], a BFTree (Best First Tree) [34], a random Forest [4], and SMO [39].

In the experiments, WEKA [46] was implemented as the classification tool. It was used to implement the previous set of classifiers, the parameters of these classifiers are kept at default values as set by WEKA. The experiments used a 10-fold cross validation approach in order to decrease variance. Moreover, each dataset was analyzed by all the classifiers, and the accuracies of the inferred models were recorded. The average accuracy of these models was used as a measure to express their difficulty in learning (i.e., their learnability value).

Chapter 4

The Experimental Study

4.1 Parameters Used to Describe the Monotone Structure of a Dataset

In order to further explore the relationships between the monotonic characteristics of the datasets and their difficulties, linear regression models were generated using these characteristics as the independent variables, and the difficulties of the datasets as the dependent value.

As discussed above, the difficulty of a dataset can be derived as the average accuracy when it is classified by the classifiers listed in Section 3.3. Moreover, the monotonic characteristics described in Table 4.1 were considered to be strong indicators of the datasets difficulty. In this study, the monotonic characteristics P_1 to P_6 were used as the independent (explanatory) variables to build the models. The reason why P_7 is excluded is because $P_7 = 1 - P_2 - P_5$.

The effectiveness of the chosen monotonic characteristics is in part indicated in Figure 4.1. This figure shows some experimental results performed on binary datasets which have 8 attributes, and each of them contains between 120 and 190 vectors. As can be observed in this figure, it seems that with an increasing number of CMP pairs, the difficulty of accurately analyzing datasets also increases. It also shows that when more unique vectors are observed in CMP pairs, the average accuracy of the derived classification models is going down. Furthermore, given two datasets, the one with more extended border points seems to be more difficult to accurately classify.

At this point one may question what happens if a function is used to transform the data in a different form and possibly dimensionality. This happens, for instance, when kernel functions are used in classification [2, 33]. If the data transformations used can change the ordering (e.g., greater than or equal than) between pairs of vectors, then the values of the six key parameters denoted as $P_1, P_2, \ldots, P_6$ may change as well. Depending on the way these six parameters change,

TABLE 4.1: The seven monotonic characteristics of the numeric datasets.

$P_1 = \frac{\text{Number of unique vectors found in CMP pairs}}{\text{Number of all vectors in the training dataset}}$	The ratio of the unique vectors that appear in the CMP pairs.
$P_2 = \frac{\text{Number of CMP pairs}}{\text{Number of all possible pairs}}$	The ratio of all possible pairs that are CMP pairs.
$P_3 = \frac{\text{Number of CMP pairs}}{\text{Number of all possible positive-negative pairs}}$	The ratio of the positive-negative pairs that are CMP pairs.
$P_4 = \frac{\text{Number of extended border points}}{\text{Number of vectors in the training dataset}}$	The ratio of the vectors that are extended border points.
$P_5 = \frac{\text{Number of all AMP1 pairs}}{\text{Number of all possible pairs}}$	The ratio of all possible pairs that are AMP1 pairs.
$P_6 = \frac{\text{Number of monotonically related pairs of vectors}}{\text{Number of all possible pairs}}$	The ratio of the pairs that are monotonically related.
$P_7 = \frac{\text{Number of all AMP2 pairs}}{\text{Number of all possible pairs}}$	The ratio of all possible pairs that are AMP2 pairs.

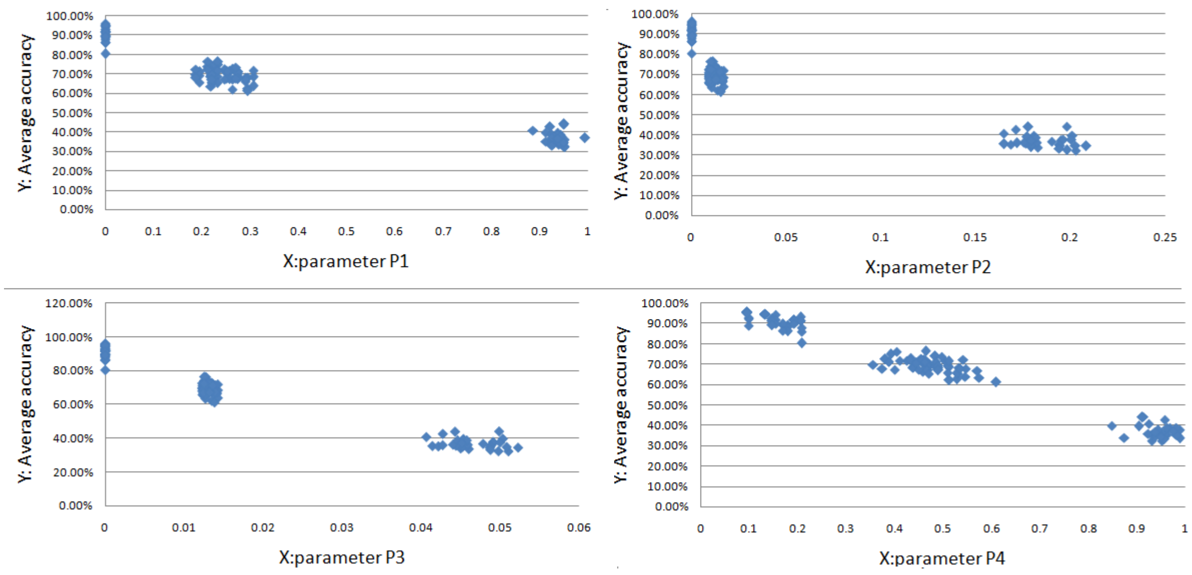

FIGURE 4.1: Relationships between average accuracy and some monotonic features for binary datasets when $n = 8$. They may have different positive/negative attributes.

the transformed dataset may become easier or more difficult to be analyzed. However, if the data transformation cannot affect the ordering of pairs of vectors (and assuming the class values are not affected either), then the learnabilities of the original and transformed datasets will remain identical.

The experiments were performed as follows. First a list of n-attribute numeric datasets were collected as the experimental datasets. Their vectors had been classified as either “postive” or "negative." Furthermore, they should present various levels of difficulties, that is, some of the datasets can be classified with high accuracies, while some others should be classified with low accuracies.

Next, the collected experimental datasets were split into two groups as the training and the testing data. Each dataset in the training group was analyzed in two aspects. First it was analyzed by the classifiers mentioned in Section 3.3. The average of the classification accuracies was recorded as its level of difficulty in learning (i.e., its learnability value). Next, its monotonic characteristics were calculated by using the formulas listed in Table 4.1.

After that, a group of linear regression models were generated using the PROC REG function in SAS [37]. The quality of the models was evaluated by their R-Square values and their performance. The models with higher R-Square values are usually more accurate in predicting the difficulty of the testing datasets. The performance of the models can be assessed as follows. First the monotonic characteristics of the testing datasets and their difficulties were calculated in the same way as that of the training datasets. Next, such monotonic characteristics were used in the generated regression models to predict the difficulties of the corresponding testing datasets. The difference between the predicted difficulties and the real difficulties are what is of interest. If the deviations were small, a regression model was believed to be an accurate one. Therefore, the mean of the deviations is an important evaluator as well.

Furthermore, the variance of the deviations is another issue. Two regression models *A* and *B* may have similar means of deviations. However, the deviations produced by model *A* may have a large variance while the ones produced by model *B* are more stable. That is, model *A* may make some predictions very accurately but be very inacurate in some other predictions. Meanwhile, model *B* does all the predictions with similar deviations. In such cases one would prefer model *B* since it has less risk of generating meaningless predictions (i.e., predictions that are very different from the actual values). Therefore, in this study, the deviations were analyzed by their mean and their 95% confidence intervals.

The next two sections provide the details of the experiments performed on some binary and continuous numeric datasets, respectively. The experimental results would be provided together with the analysis.

4.2 Experiments with Binary Datasets

4.2.1 Experiments on Artificial Binary Datasets

The monotonic characteristics of the datasets are the monotonic relationships between its pairs of vectors. If a dataset has too few pairs of vectors that are monotonically related, then the values of its monotonic characteristics P_1 to P_6 will all be close to 0, and therefore no useful information can be provided when generating regression models.

Therefore, the immediate problem is, what is the precondition under which the proposed approach can generate accurate regression models? A set of experiments were designed to explore this issue. The experimental datasets used in this set of experiments are some artificial binary datasets with dimensions n = 10, 14, 18 and 22. Each dataset had from 400 to 1,500 vectors and each vector was labeled as either "positive" or "negative."

At the beginning, the experimental datasets were categorized by their dimensions, four groups of datasets were therefore generated. Furthermore, the datasets in each group were further split into several experimental groups by their monotonic characteristic P_6 (the ratios of the pairs that are monotonically related), and each experimental group had around 300 datasets. A list of regression models were generated from these experimental groups, and their R-Square values are what is of interest. Figure 4.2 shows some of the experimental results.

As one can observe from this figure, the R-Square values of the models were consistently above 0.92 when the experimental datasets have more than 6% pairs of vectors that are monotonically related. In other words, it seems that under this simple condition the quality of the models can be somehow ensured. Therefore, this study only focuses on numeric datasets that have more than 6% monotonically related pairs.

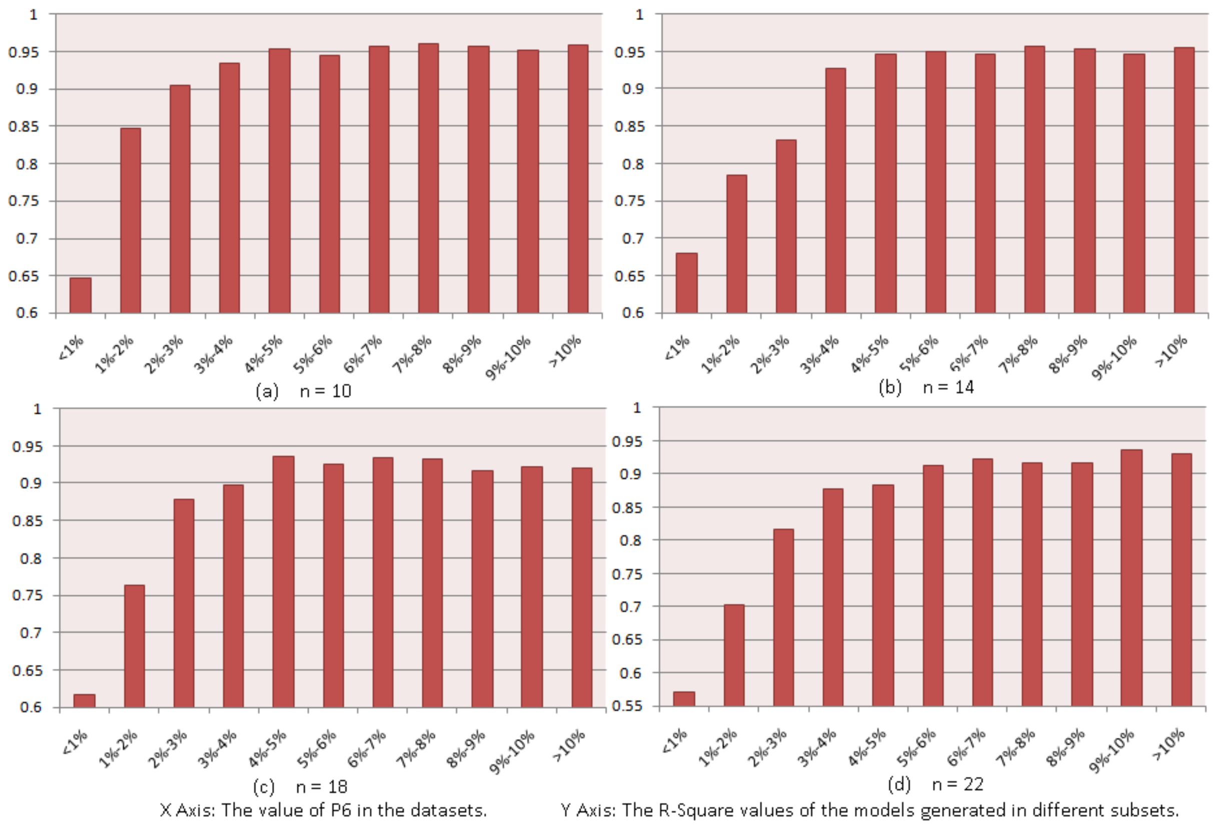

FIGURE 4.2: The R-Square values of the models generated from artificial binary datasets with different dimensions.

The above four groups of datasets for *n*=10, 14, 18 and 22 served as pilot tests. Next, artificial datasets were generated according to the procedures described in Section 3.2. Now the dimensions ranged from 8 to 60. In these experiments, the datasets were grouped by their dimensions, and each group contained about 450 datasets, where 350 of them were randomly selected for training and the remaining 100 were used for testing. Furthermore, according to the way they were generated, the difficulty of these datasets would also vary significantly. The quality of the regression models would be evaluated by their R-Square values and also their performance in predicting the difficulties of the testing datasets. Tables 4.2 and 4.3 list some experimental results on these datasets.

As one can observe from this table, when there are sufficient number of vectors that are monotonically related in a dataset, the difficulty of this dataset can be accurately predicted by analyzing its monotonic characteristics. The R-Square values for all the models listed in this table are greater

TABLE 4.2: The regression models generated for different dimensions of artificial binary datasets.

Dimensions	Regression models
N= 8	Y = 0.532 + 9.345×P1 - 22.55×P2 - 0.312×P3 - 0.236×P4 + 1.734×P5 + 0.209×P6
N= 10	Y = 0.737 + 10.19×P1 - 44.48×P2 - 0.103×P3 - 0.260×P4 + 1.104×P5 - 0.013×P6
N= 13	Y = 0.891 + 10.43×P1 - 41.27×P2 - 0.387×P3 - 0.330×P4 + 0.307×P5 + 0.151×P6
N= 17	Y = 0.746 + 14.42×P1 - 71.86×P2 - 0.189×P3 - 0.253×P4 + 0.019×P5 + 0.628×P6
N= 20	Y = 0.686 + 26.76×P1 - 87.84×P2 - 0.126×P3 - 0.118×P4 - 1.017×P5 + 0.879×P6
N= 25	Y = 0.857 + 26.78×P1 - 56.75×P2 - 0.092×P3 - 0.384×P4 + 1.147×P5 - 0.201×P6
N= 30	Y = 0.789 + 18.16×P1 - 46.25×P2 - 0.170×P3 - 0.216×P4 + 0.084×P5 + 0.133×P6
N= 35	Y = 0.746 + 11.64×P1 - 71.38×P2 - 0.189×P3 - 0.253×P4 + 0.019×P5 + 0.628×P6
N= 40	Y = 0.686 + 23.86×P1 - 78.85×P2 - 0.126×P3 - 0.118×P4 - 1.017×P5 + 0.879×P6
N= 45	Y = 0.857 + 22.35×P1 - 69.11×P2 - 0.092×P3 - 0.384×P4 + 1.147×P5 - 0.201×P6
N= 50	Y = 0.789 + 18.03×P1 - 76.25×P2 - 0.164×P3 - 0.216×P4 + 0.084×P5 + 0.133×P6
N= 60	Y = 0.789 + 16.63×P1 - 59.64×P2 - 0.170×P3 - 0.143×P4 - 0.043×P5 + 0.167×P6

TABLE 4.3: Details of the regression models listed in Table 4.2.

Dimension	Number of vectors in each experimental dataset	Difficulties of the training datasets	R-Square value of the models	Mean of the deviation	95% confidence intervals of the deviations
n= 8	120 to 190	53.85% to 96.73%	0.986	1.806%	1.565% to 2.047%
n=10	400 to 600	52.66% to 93.97%	0.928	2.513%	1.710% to 2.118%
n=13	800 to 1,200	55.87% to 91.57%	0.937	2.415%	1.732% to 2.937%
n=17	1,200 to 1,600	56.83% to 93.66%	0.927	2.208%	1.833% to 2.583%
n=20	1,200 to 1,600	53.12% to 96.78%	0.943	2.376%	1.973% to 2.779%
n=25	1,200 to 1,600	54.51% to 95.33%	0.928	2.632%	2.136% to 3.128%
n=30	1,200 to 1,600	56.78% to 96.46%	0.942	2.517%	1.968% to 3.055%
n=35	1,200 to 1,600	56.37% to 93.60%	0.925	2.226%	1.879% to 2.573%
n=40	1,200 to 1,600	57.47% to 93.27%	0.932	2.562%	2.086% to 3.308%
n=45	1,200 to 1,600	55.87% to 95.96%	0.937	2.396%	1.832% to 2.960%
n=50	1,200 to 1,600	55.29% to 94.45%	0.939	2.452%	1.973% to 2.931%
n=60	1,200 to 1,600	53.58% to 93.32%	0.942	2.523%	2.074% to 3.031%
Average			0.939	2.387%	1.846% to 2.928%

than 0.925, and some can be as high as 0.986. Therefore, these models present very strong relationships between the monotonic characteristics of the training datasets and their difficulties.

The performance of the models are encouraging in predicting the difficulties of the testing datasets. The mean of all the deviations is 2.387%, and in 95% of the cases, the difference between the predicted difficulties and the real difficulties is less than 2.928%. This observation indicates that the generated regression models are very accurate.

In Table 4.3 and consecutive tables some of the classifiers performed quite poorly. This is indicated by the low values of the range of classification difficulties (which can be as low as in the 50s percentage points). However, the inferred regression models could predict low or high accuracies (e.g., learnability levels) of the classifiers used quite accurately as indicated by the high R-Square values of these regression models. For stability reasons we dropped the top two best and bottom two worse classification accuracy values in order to block out any outlier behaviors by some of the classifiers used in this study.

A related question at this point is which classifiers to include and which not to include. This question cannot be answered in a generally accepted manner. This study included a wide range of well-known classifiers from Weka. One may always argue why a given classifier was included or not included in this group. In the future new significantly accurate classifiers may be introduced. In that case, one may include them in a similar study. This point reveals that there is a need to have a "representative" group of classifiers that could be used to standardize the quantification of the proposed learnability metric.

In traditional applications, in order to evaluate how difficult it is to accurately classify an n-attribute binary dataset, such a dataset has to be analyzed by a set of classifiers. The times spent for each classification are different depending on the classification algorithms. For some algorithms, such as the Decision Tree (J48) and the Bayes Analyze, the classification can be done quickly. However, if one uses SVM or KStart algorithms, they usually need long processing time. Furthermore, in order to reduce the variance, one may need to implement the 10 or 20 cross-validation scheme, which makes the analysis time even longer. Therefore, it is quite time consuming to get the actual average accuracy from a set of classifiers. However, by analyzing the monotonic characteristics, one can easily predict the difficulty of such datasets accurately. It only takes O(N^2) time to get all the monotonic characteristics and not much time to use the appropriate regression model, where N is the number of the observations in the dataset.

4.2.2 Experiments on Some Real-Life Binary Datasets

In order to further validate the effectiveness of the models, this section uses the same approach to analyze the difficulties of some real-life binary datasets. These are: the *Spect* [44], the *Adult*, the *Mushrooms*, the *W1a*, and the *Covtype* [31] datasets.

The experimental datasets in this section were generated as follows. The *Spect* dataset has 22 binary attributes and about 17% of its pairs of vectors are monotonically related. In this set of experiments, 200 experimental datasets were generated by randomly selecting various subsets of vectors from *Spect*. Moreover, there are 123 binary attributes in the *Adult* dataset and it has about 32,000 vectors. In this study, three groups of subsets were generated from this dataset as additional experimental datasets. To be more specific, the datasets in the first group contain only 20-attribute vectors, the datasets in the second and the third groups have 40 and 60 attributes, respectively. They are denoted as experimental groups *Adult_20*, *Adult_40*, and *Adult_60*. Furthermore, in order to ensure the precondition that all the experimental datasets should have more than 6% monotonically related pairs, such experimental datasets were generated in the way described in Algorithm 5.

For instance, in generating the *Adult_60* experimental datasets, 60 attributes were randomly selected from the entire 123 ones, all duplicated 60-attribute vectors were removed, and 10 vectors were randomly selected without replacement. Next, all the remaining 60-attribute vectors were compared to these 10 vectors, the vectors that are monotonically related to at least one of them were selected to form a sample space. A total of 6,000 vectors were therefore selected. After that, 500 groups of vectors were randomly selected from the sample space, and each group had about

1,200 to 1,600 vectors. The vectors in these groups were labeled in the way described in Section 3.2 to generate 500 experimental datasets with different levels of difficulties.

Furthermore, the *Mushrooms* dataset has 112 binary attributes and 8,124 vectors, the *W1a* dataset has 300 binary attribute and 50,000 vectors, and the *Covtype* dataset has 38 binary datasets and 580,000 vectors. Algorithm 5 was again applied to each of these datasets, and eight experimental groups were generated and were named as *Mushrooms_20*, *Mushrooms_40*, *Mushrooms_60*, *W1a_20*, *W1a_40*, *W1a_60*, *Covtype_20* and *Covtype_38*. The details of such datasets are provided in Table 4.4, their derived regression models are listed in Table 4.5.

Algorithm 5: Generate experimental datasets which have adequate number of monotonically related pairs

```
Input  : A list of $n$-attribute vectors $Vectors$, $K$, $N$, and $M$
Output: $M$ number of $n$-attribute binary datasets that have many pairs of vectors to be
         monotonically related.
1  $Sample\_Space \leftarrow \emptyset$;
2  Randomly select $K$ vectors from $Vectors$ without replacement, they form a subset called
   $Initial\_K$.
3  while $Sample\_Space$ has less than $N$ vectors and the $Vectors$ is not empty do
4      Randomly select a vector $V$ from $Vectors$;
5      if $V$ is monotonically related to at least one vector $V_k \in Initial\_K$ then
6          $Sample\_Space \leftarrow Sample\_Space \cup V$
7      end
8  end
9  Generate $M$ binary datasets from $Sample\_Space$ in the way described in Section 3.2.
10 Return the generated binary datasets;
```

Table 4.6 shows the experimental results on these real-life binary datasets. As one can observe from this table, the monotonic characteristics of the datasets are strong indicators of their difficulties. This is supported by the fact that the average R-Square value of the models is 0.954, and when such models were used to predict the difficulties of the testing datasets, in about 95% of the cases, the differences between the predicted difficulties and the real difficulties were no more than 2.345%.

TABLE 4.4: The information of the datasets listed in Table 4.5.

Datasets	Number of vectors in experimental datasets	Number of training datasets	Difficulties of the training datasets	R-Square value of the models
Spect	140 to 200	150	67.56% to 94.47%	0.964
Adult_20	1,000 to 1,200	400	56.69% to 91.28%	0.958
Adult_40	1,400 to 1,600	400	61.28% to 92.68%	0.954
Adult_60	1,400 to 1,600	400	60.25% to 93.47%	0.945
Mushrooms_20	1,000 to 1,200	400	59.47% to 93.54%	0.948
Mushrooms_40	1,400 to 1,600	400	55.71% to 94.75%	0.953
Mushrooms_60	1,400 to 1,600	400	57.88% to 93.56%	0.929
W1a_20	1,000 to 1,200	400	53.58% to 96.87%	0.945
W1a_40	1,400 to 1,600	400	60.59% to 92.57%	0.977
W1a_60	1,400 to 1,600	400	54.58% to 94.15%	0.951
Covtype_20	1,000 to 1,200	400	57.97% to 95.66%	0.955
Covtype_38	1,400 to 1,600	400	56.12% to 93.86%	0.934
Average				0.954

TABLE 4.5: The regression models generated for some real-life binary datasets.

Datasets	Regression models
Spect	Y = 0.637 + 6.35×P1 - 12.45×P2 - 11.35×P3 - 0.436×P4 + 1.104×P5 + 0.217×P6
Adult_20	Y = 0.576 + 4.24×P1 - 15.34×P2 - 10.86×P3 - 0.265×P4 + 1.186×P5 + 0.331×P6
Adult_40	Y = 0.587 + 8.43×P1 - 17.38×P2 - 11.19×P3 - 0.372×P4 + 1.123×P5 + 0.356×P6
Adult_60	Y = 0.557 + 5.39×P1 - 11.98×P2 - 12.32×P3 - 0.351×P4 + 1.245×P5 + 0.415×P6
Mushrooms_20	Y = 0.734 + 9.31×P1 - 25.32×P2 - 17.24×P3 - 0.783×P4 + 0.897×P5 + 0.478×P6
Mushrooms_40	Y = 0.786 + 7.32×P1 - 21.44×P2 - 16.98×P3 - 0.673×P4 + 0.942×P5 + 0.375×P6
Mushrooms_60	Y = 0.772 + 6.29×P1 - 27.43×P2 - 17.38×P3 - 0.683×P4 + 1.106×P5 + 0.416×P6
W1a_20	Y = 0.685 + 5.59×P1 - 19.32×P2 - 19.28×P3 - 0.356×P4 + 1.033×P5 + 0.176×P6
W1a_40	Y = 0.678 + 7.25×P1 - 18.24×P2 - 18.35×P3 - 0.376×P4 + 1.105×P5 + 0.156×P6
W1a_60	Y = 0.691 + 6.67×P1 - 19.11×P2 - 21.31×P3 - 0.386×P4 + 1.042×P5 + 0.166×P6
Covtype_20	Y = 0.721 + 6.92×P1 - 7.472×P2 - 14.61×P3 - 0.537×P4 + 1.053×P5 + 0.205×P6
Covtype_38	Y = 0.732 + 8.35×P1 - 8.272×P2 - 13.88×P3 - 0.561×P4 + 0.964×P5 + 0.198×P6

TABLE 4.6: Experimental results from the real-life binary datasets listed in Table 4.5.

Datasets	Number of vectors in experimental datasets	Number of testing datasets	Mean of the deviation	95% confidence intervals of the deviations
Spect	140 to 200	50	1.105%	0.859% to 1.351%
Adult_20	1,000 to 1,200	100	1.828%	1.496% to 2.160%
Adult_40	1,400 to 1,600	100	2.017%	1.658% to 2.376%
Adult_60	1,400 to 1,600	100	2.146%	1.775% to 2.517%
Mushrooms_20	1,000 to 1,200	100	1.745%	1.451% to 2.039%
Mushrooms_40	1,400 to 1,600	100	2.074%	1.683% to 2.465%
Mushrooms_60	1,400 to 1,600	100	2.341%	1.985% to 2.697%
W1a_20	1,000 to 1,200	100	1.659%	1.357% to 1.961%
W1a_40	1,400 to 1,600	100	1.939%	1.429% to 2.449%
W1a_60	1,400 to 1,600	100	2.142%	1.432% to 2.652%
Covtype_20	1,000 to 1,200	100	1.868%	1.697% to 2.159%
Covtype_38	1,400 to 1,600	100	2.147%	1.828% to 2.366%
Average			1.906%	1.467% to 2.345%

4.3 Experiments with Some Continuous Datasets

The above experimental results are very encouraging. However, can these results be generalized more? After all, the above experiments only analyzed binary datasets. In these cases the population space is small and the vectors are simple (they are comprised by 0s and 1s). What about the case when datasets are comprised by vectors with continuous attributes? Therefore, a new family of experiments were designed and implemented to test the robustness of the proposed monotonicity based approach by analyzing some continuous datasets.

4.3.1 Generating Experimental Datasets

In this family of experiments, the experimental data were generated from nine continuous datasets selected from the UCI Machine Learning Repository. These datasets are the *Abalone*, the *Yeast*, the *Ecoli*, the *Diabetes*, the *Blood test*, the *Liver*, the *Iris*, the *Strength*, and the *Strike* [44], and each has from 600 to 4,800 vectors. They are considered as the **target** datasets in these experiments.

It was observed that the vectors in some datasets are assigned to more than two categories, the *Abalone*, the *Yeast* and the *Iris* are some of the examples. However, our research focuses on two-class classifications, Therefore, for a dataset which contains more than two categories, a set of two-class subsets were generated by randomly selecting any two classes of vectors from the original dataset. For instance, there are three categories of vectors in the *Iris* dataset: the Setosa vectors, the Versicolour vectors and the Virginica vectors. A dataset D_1 with only two classes can be comprised by vectors from the categories Setosa and Versicolour, the categories Versicolour and Virginca, or the categories Setosa and Virginca.

Furthermore, in order to get enough experimental datasets, a new dataset can be created by randomly selecting a portion of the vectors from D_1 without replacement. Different samplings may generate experimental datasets with different monotonic characteristics, and one can use this approach to create sufficient observations.

Similar to the binary case, the experimental data in this set of experiments should also be comprised of datasets with different levels of difficulty. The difficulties of the experimental datasets can be somehow impacted during their generations by including different number of vectors from different categories. For example, in generating the *Blood* datasets, if 80% of the vectors are chosen from class 1(people donated blood in March 2007), and the rest are chosen from class 0 (people did not donate blood in March 2007), then for most of the classifiers, the accuracies of the derived classification models should be higher than 80%. Since they are believed to be more sophisticated than the most intuitive guess: "all of them are class 1 vectors." By carefully selecting vectors from different categories, one can generate training datasets that have different levels of difficulty.

Moreover, it is also noticed that the dimension of the target datasets are varied from $n = 4$ to $n = 7$. Therefore, for each target dataset which has more than 4 attributes, several groups of experimental datasets can be generated from them with different dimensions. For instance, the *Abalone* dataset has 7 continuous attributes denoted as $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. Therefore, based on which group of 4-attribute datasets can be generated by randomly selecting four of its attributes, such as $\{A_1, A_2, A_3, A_4\}$, or $\{A_3, A_4, A_6, A_7\}$, the selected attributes, combined with the class values, formed some of the desired datasets. This 4-attribute experimental group is denoted as *Abalone_4*.

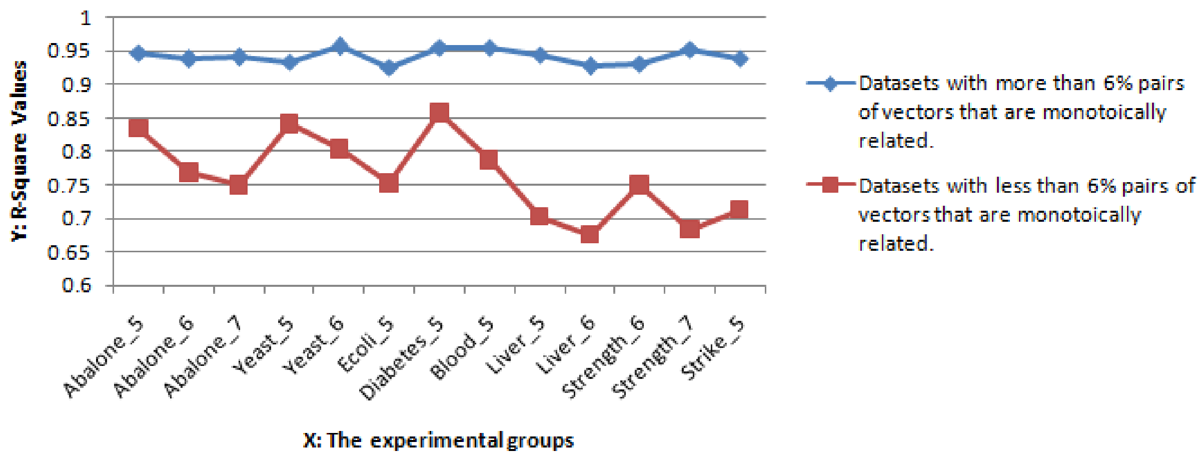

FIGURE 4.3: The R-Square values of the models generated from continuous experimental groups with different levels of monotonicity.

Furthermore, another group of 5-attribute datasets, called *Abalone_5*, can be generated in a similar way by randomly selecting 5 attributes, and so did the groups of *Abalone_6* and *Abalone_7*.

According to the previous experimental results, the quality of the generated regression models was ensured when the experimental datasets have sufficient pairs of vectors that are monotonically related. Some experiments were performed to find out the appropriate value of this criterion. In these experiments, 13 experimental groups were selected, and the datasets in an experimental group were further split into two categories. These are, the ones that have more than 6% pairs of vectors that are monotonically related, and the ones that have less than 6% pairs of vectors that are monotonically related. Next, regression models were generated by analyzing datasets in each of the categories and their R-Square values were recorded.

As one can observe from Figure 4.3, the quality of the models are unpredictable and inaccurate when they were generated from datasets with less than 6% monotonically related pairs. However, when the training datasets have more than 6% monotonically related pairs, the R-Square values of the models are consistently above 0.93. This indicates that the generated models are reliable. Therefore, this study only concentrated on datasets that have more than 6% of pairs that are monotonically related. Some groups of datasets, such as *Yeast_7*, *Ecoli_6*, and *Ecoli_7*, because they have too few pairs of monotonically related vectors, were not analyzed in this study.

TABLE 4.7: Regression models generate from some real-life continuous datasets.

Datasets	Regression models
Abalone_4	Y = 0.637 + 7.65×P1 - 11.76×P2 - 11.63×P3 - 0.526×P4 + 0.905×P5 + 0.194×P6
Abalone_5	Y = 0.628 + 7.22×P1 - 9.436×P2 - 16.14×P3 - 0.428×P4 + 1.625×P5 + 0.252×P6
Abalone_6	Y = 0.665 + 7.14×P1 - 14.25×P2 - 14.42×P3 - 0.129×P4 + 1.425×P5 + 0.284×P6
Abalone_7	Y = 0.632 + 6.67×P1 - 13.14×P2 - 17.53×P3 - 0.421×P4 + 1.351×P5 + 0.145×P6
Yeast_4	Y = 0.715 + 5.67×P1 - 14.63×P2 - 12.62×P3 - 0.355×P4 + 1.254×P5 + 0.169×P6
Yeast_5	Y = 0.709 + 6.12×P1 - 13.47×P2 - 13.73×P3 - 0.625×P4 + 1.154×P5 + 0.205×P6
Yeast_6	Y = 0.712 + 5.75×P1 - 15.48×P2 - 11.74×P3 - 0.278×P4 + 1.186×P5 + 0.311×P6
Ecoli_4	Y = 0.612 + 8.34×P1 - 12.74×P2 - 13.63×P3 - 0.259×P4 + 1.171×P5 + 0.265×P6
Ecoli_5	Y = 0.617 + 8.42×P1 - 15.75×P2 - 12.45×P3 - 0.341×P4 + 1.059×P5 + 0.258×P6
Diabetes_4	Y = 0.676 + 7.46×P1 - 13.54×P2 - 15.62×P3 - 0.453×P4 + 1.185×P5 + 0.165×P6
Diabetes_5	Y = 0.678 + 7.42×P1 - 12.55×P2 - 16.65×P3 - 0.486×P4 + 1.326×P5 + 0.185×P6
Blood_4	Y = 0.682 + 8.32×P1 - 12.63×P2 - 16.35×P3 - 0.253×P4 + 1.205×P5 + 0.215×P6
Blood_5	Y = 0.674 + 8.34×P1 - 11.78×P2 - 13.84×P3 - 0.354×P4 + 1.247×P5 + 0.236×P6
Liver_4	Y = 0.667 + 7.53×P1 - 16.74×P2 - 17.74×P3 - 0.452×P4 + 1.104×P5 + 0.298×P6
Liver_5	Y = 0.658 + 7.34×P1 - 14.54×P2 - 18.73×P3 - 0.354×P4 + 1.115×P5 + 0.258×P6
Liver_6	Y = 0.662 + 9.23×P1 - 17.36×P2 - 14.34×P3 - 0.159×P4 + 1.157×P5 + 0.275×P6
Iris	Y = 0.876 + 3.11×P1 - 13.63×P2 - 17.35×P3 - 0.428×P4 + 1.095×P5 + 0.265×P6
Strength_4	Y = 0.634 + 5.36×P1 - 17.76×P2 - 22.37×P3 - 0.327×P4 + 0.985×P5 + 0.106×P6
Strength_5	Y = 0.631 + 3.46×P1 - 16.73×P2 - 19.56×P3 - 0.452×P4 + 0.968×P5 + 0.257×P6
Strength_6	Y = 0.627 + 6.75×P1 - 13.47×P2 - 21.96×P3 - 0.356×P4 + 1.015×P5 + 0.162×P6
Strength_7	Y = 0.633 + 9.42×P1 - 12.57×P2 - 17.78×P3 - 0.251×P4 + 1.077×P5 + 0.168×P6
Strike_4	Y = 0.675 + 6.32×P1 - 18.36×P2 - 15.15×P3 - 0.352×P4 + 1.256×P5 + 0.325×P6
Strike_5	Y = 0.667 + 7.11×P1 - 13.55×P2 - 13.36×P3 - 0.174×P4 + 1.174×P5 + 0.286×P6

By using the approaches described above, each experimental group (for instance, *Abalone_5*, *Iris*, or *Yeast_6*) were generated and each contains from 300 to 550 2-class continuous experimental datasets. Furthermore, such datasets present different levels of difficulty.

4.3.2 Experiments When the Training and Testing Datasets Were Originated from the Same Target Dataset

This section explores the power of monotonicity by studying "relevant" datasets. That is, the training and testing datasets used in this set of experiments were originated from the same target dataset. In studying these training datasets, one can use the same strategy as described in the binary case to generate regression models, and next use them to evaluate the difficulty of the testing datasets.

TABLE 4.8: Experimental results from real-life continuous datasets listed in Table 4.7.

Datasets	Number of training datasets	Number of testing datasets	Difficulties of the training datasets	R-Square value of the models	Mean of the deviation	95% confidence intervals of the deviations
Abalone_4	450	100	57.69% to 93.58%	0.952	1.105%	0.859% to 1.351%
Abalone_5	450	100	56.97% to 96.25%	0.948	1.228%	0.725% to 1.731%
Abalone_6	450	100	58.25% to 95.48%	0.938	1.267%	0.736% to 1.798%
Abalone_7	450	100	60.33% to 92.47%	0.941	1.209%	0.817% to 1.601%
Yeast_4	400	100	58.44% to 91.23%	0.925	1.458%	1.012% to 2.104%
Yeast_5	400	100	58.36% to 94.33%	0.933	1.325%	0.917% to 1.733%
Yeast_6	400	100	57.37% to 95.21%	0.957	1.439%	1.005% to 1.873%
Ecoli_4	120	80	58.26% to 94.76%	0.948	1.539%	1.115% to 1.963%
Ecoli_5	120	80	59.85% to 93.87%	0.925	1.542%	1.136% to 1.948%
Diabetes_4	120	80	64.57% to 92.59%	0.948	1.878%	1.451% to 2.305%
Diabetes_5	120	80	63.14% to 94.88%	0.955	1.698%	1.208% to 2.188%
Blood_4	200	100	58.36% to 93.26%	0.954	1.528%	1.215% to 1.814%
Blood_5	200	100	61.56% to 94.37%	0.956	1.564%	1.103% to 2.205%
Liver_4	120	80	57.24% to 95.22%	0.941	1.458%	1.015% to 1.901%
Liver_5	120	80	56.87% to 94.18%	0.944	1.465%	1.069% to 1.871%
Liver_6	120	80	57.13% to 96.24%	0.927	1.598%	1.157% to 2.039%
Iris	120	80	90.68% to 100%	0.835	2.328%	0.885% to 3.771%
Strength_4	300	100	59.56% to 93.69%	0.966	1.287%	0.763% to 1.811%
Strength_5	300	100	61.27% to 94.85%	0.958	1.316%	0.982% to 1.650%
Strength_6	300	100	60.34% to 93.56%	0.951	1.613%	1.298% to 1.928%
Strength_7	300	100	58.47% to 94.78%	0.953	1.625%	1.157% to 2.093%
Strike_4	200	100	61.58% to 95.19%	0.945	1.898%	1.426% to 2.370%
Strike_5	200	100	59.87% to 94.32%	0.940	1.877%	1.397% to 2.357%
Average				0.944	1.306%	1.024% to 1.588%

Some details of these regression models are provided in Tables 4.7 and 4.8. As one can observe from these tables, the training datasets represent diverse levels of difficulty, and the average R-Square values of the generated regression models is equal to 0.944.

Moreover, Table 4.8 lists the deviations when such models were used to predict the difficulties of the testing datasets. According to this table, the mean of all the deviations is equal to 1.306%. Furthermore, the same confidence interval scheme was implemented as in the binary case to test their variance. As it is indicated from this table, for most of the predictions, their deviations are not larger than 1.588%.

Let us use the *Abalone_4* experimental datasets as an example here. In these experiments, a total of 550 4-attribute subsets were generated from the target dataset *Abalone* with different levels of difficulty.

Among these subsets, 450 of them were randomly chosen and were used to generate a regression model. That model was next applied to analyze the remaining 100 testing datasets. In these 100 predictions, the mean of the deviations was equal to 1.105% with 95% confidence interval (0.859%, 1.351%). Statistically speaking, 95% of the predictions differ from the actual values by no more than 1.351%. This provides strong evidence that the generated regression model is of high quality.

4.3.3 Evaluating the effect of monotonic characteristics over different target datasets

The training and testing data in this family of tests were somehow "irrelevant." In other words, the training and the testing data used in this set of experiments were originated from different target datasets. More specifically, these experiments were implemented in two scenarios. In the first scenario, the training data were comprised of datasets of the same dimensions and were originated from all target datasets.

For example, there are nine groups of 4-attribute datasets in this study, with a total number of 2,850 datasets. Among these datasets, 2,030 of them were randomly selected and aggregated to form a composite training dataset. A regression model was trained from such training data, and was later used to analyze the difficulties of the remaining datasets in each group. The derived deviations were also recorded to assess its quality.

Table **??** provides the derived regression models and the experimental results. As one can observe from this table, the R-Square values of the regression models are all above 0.924, and these models can predict the difficulties of the testing datasets with very little deviations.

For instance, when predicting the difficulties of the testing datasets generated from the *Ecoli_4* datasets, the mean of the 80 deviations was 1.587% and in 95% of the cases the deviations were less than 2.158%.

TABLE 4.9: Some characteristics of the regression models generated independently of the testing data. The actual regression models are shown in Table 4.10.

Training Datasets	R-Square	Testing group	Testing size	Means of the deviations	95% confidence intervals of the deviations
Comprised by all 4-attribute datasets except the ones being tested	0.927	Abalone_4	550	2.824%	2.320% to 3.320%
	0.910	Ecoli_4	200	1.732%	1.455% to 2.010%
	0.919	Diabetes_4	200	2.770%	2.355% to 3.184%
	0.923	Blood_4	300	3.772%	3.446% to 3.996%
	0.920	Liver_4	200	3.384%	2.937% to 3.831%
	0.914	Strike_4	300	2.458%	2.147% to 2.769%
Average	0.911			2.733%	2.189% to 3.277%
Comprised by all 5-attribute datasets except the ones being tested	0.931	Abalone_5	550	2.715%	2.426% to 3.004%
	0.914	Ecoli_5	200	1.862%	1.275% to 2.449%
	0.912	Diabetes_5	200	2.854%	2.463% to 3.245%
	0.930	Blood_5	300	3.428%	2.816% to 4.040%
	0.916	Liver_5	200	3.294%	2.864% to 3.724%
	0.934	Strike_5	300	2.366%	1.852% to 2.880%
Average	0.922			2.719%	2.241% to 3.197%
Comprised by all 6-attribute datasets except the ones being tested	0.889	Abalone_6	550	2.953%	2.498% to 3.408%
	0.893	Yeast_6	500	3.524%	2.886% to 4.126%
	0.907	Liver_6	200	3.462%	2.773% to 4.151%
	0.912	Strength_6	200	3.354%	2.563% to 4.145%
Average	0.900			3.323%	2.625% to 4.021%

In general, the deviations produced by these regression models are somewhat greater than the ones listed in Table 4.8, but most of them are still less than 2.4%. This observation indicates that the generated regression models are reliable and effective.

In the second scenario, the training data were totally irrelevant (independent) of the testing data. That is, the training datasets were selected from different experimental groups than the testing datasets. For example, there are four groups of 6-attribute datasets, these are: *Abalone_6*, *Yeast_6*, *Liver_6*, and the *Strength_6*. Suppose the current task is to evaluate the difficulties of the *Liver_6* datasets. Then the training data would be comprised by all the *Abalone_6*, *Yeast_6*, and the *Strength_6* datasets. While all the Liver_6 datasets were used for testing.

In these sets of experiments, the testing datasets were quite different from the training datasets. For instance, the training data may use different measuring units from the testing data, such as inches used in training datasets but kilograms used in the testing ones. It is also possible that in the

TABLE 4.10: Regression models for datasets listed in Table 4.9.

Testing dataset	Regression models
Abalone_4	Y = 0.672 + 11.25×P1 - 15.62×P2 - 11.63×P3 - 0.526×P4 + 0.877×P5 + 0.184×P6
Diabetes_4	Y = 0.592 + 8.596×P1 - 9.582×P2 - 12.24×P3 - 0.526×P4 + 0.943×P5 + 0.185×P6
Blood_4	Y = 0.635 + 12.52×P1 - 12.56×P2 - 12.68×P3 - 0.549×P4 + 1.105×P5 + 0.179×P6
Liver_4	Y = 0.663 + 7.418×P1 - 9.581×P2 - 11.55×P3 - 0.675×P4 + 1.026×P5 + 0.194×P6
Iris	Y = 0.878 + 7.584×P1 - 12.25×P2 - 11.42×P3 - 0.425×P4 + 0.936×P5 + 0.187×P6
Strength_4	Y = 0.654 + 8.593×P1 - 11.86×P2 - 10.35×P3 - 0.488×P4 + 0.852×P5 + 0.193×P6
Strike_4	Y = 0.625 + 9.524×P1 - 12.73×P2 - 10.89×P3 - 0.782×P4 + 0.871×P5 + 0.252×P6
Abalone_5	Y = 0.636 + 7.685×P1 - 9.526×P2 - 11.69×P3 - 0.457×P4 + 0.845×P5 + 0.204×P6
Diabetes_5	Y = 0.587 + 9.563×P1 - 11.62×P2 - 10.34×P3 - 0.257×P4 + 1.025×P5 + 0.196×P6
Blood_5	Y = 0.686 + 9.475×P1 - 10.85×P2 - 9.546×P3 - 0.527×P4 + 1.089×P5 + 0.221×P6
Liver_5	Y = 0.637 + 10.52×P1 - 8.574×P2 - 12.65×P3 - 0.274×P4 + 0.890×P5 + 0.204×P6
Strength_5	Y = 0.685 + 10.75×P1 - 7.563×P2 - 11.57×P3 - 0.341×P4 + 0.857×P5 + 0.187×P6
Strike_5	Y = 0.638 + 11.53×P1 - 8.524×P2 - 12.54×P3 - 0.367×P4 + 0.957×P5 + 0.186×P6
Abalone_6	Y = 0.629 + 11.48×P1 - 12.01×P2 - 10.76×P3 - 0.428×P4 + 0.942×P5 + 0.198×P6
Yeast_6	Y = 0.653 + 12.56×P1 - 11.89×P2 - 12.45×P3 - 0.547×P4 + 1.024×P5 + 0.204×P6
Liver_6	Y = 0.661 + 10.75×P1 - 10.35×P2 - 10.34×P3 - 0.251×P4 + 1.063×P5 + 0.214×P6
Strength_6	Y = 0.639 + 9.586×P1 - 10.54×P2 - 11.77×P3 - 0.674×P4 + 1.124×P5 + 0.189×P6

training datasets all the numeric values are in float point format but in the testing datasets all the values are integer. Furthermore, it is possible that the training and testing datasets were generated from totally different application domains, such as the case between the *Blood* datasets for training and the *Iris* datasets for testing. However, according to the experimental results listed in Table 4.9 (by using the regression models listed in Table 4.10), the difficulties of the testing datasets can still be accurately predicted by regression models generated from learning some irrelevant datasets of the same dimension. This is supported by the fact that in analyzing 4-attribute datasets, most of the predictions can have deviations within 3.277%. In analyzing 5-attribute datasets, the predicted deviations are in the range of (2.241% to 3.197%). In assessing the difficulties of the 6-attribute datasets, even though the training size is small, that is, too few experimental groups were selected for training, it is still ensured that 95% of the predictions are different than the real values by no more than 4%.

Based on the above observations, it seems that even though datasets may differ in many ways, they always exhibit similar learnability patterns when one considers their monotonic characteristics. The relationships between the difficulty of the datasets and their monotonic characteristics were similar for all the numeric datasets of the same dimension.

Under this conclusion, for any numeric datasets that have sufficient number of monotonically related pairs, one may accurately predict their difficulties without implementing any classifiers, but by only spending O(N^2) time to compute their monotonic characteristics and a fixed time to apply the appropriate regression model. Furthermore, although the regression models generated in these experiments were sufficiently accurate, the training size is still relatively small, and the regression models may not be the most accurate. It is believed that when more target datasets are collected for training, the generated regression models may become even more accurate.

Chapter 5

A Meta-Learning Approach

5.1 The Motivation of the Meta-Learning Approach

As one can observe from previous sections, it is found that when datasets are comprised of highly monotonic data, then they can be classified accurately by most methods while datasets that are not comprised of highly monotonic data, tend to be more difficult to be accurately analyzed by classifiers. In other words, datasets which have very few CMP pairs are “easy” ones, while datasets which have many CMP pairs are “difficult” ones.

The next question to ask is what happens if a given dataset does not exhibit strong monotonicity. Could the previous developments still be somehow applicable, perhaps after some data manipulations? Our research suggests that the answer to this very important question is quite often affirmative.

To see this consider the following metaphor. Figure 5.1, part (a), depicts a perfectly (increasing) monotone linear function in two dimensions (2-D). Part (b) of the same figure depicts a function which is not perfectly monotonic. However, the function in part (b) can be decomposed into a sequence of perfectly monotone functions. These are alternating increasing and decreasing monotone functions as shown in part(c) of Figure 5.1.

It turns out that any dataset, even ones that exhibit no monotonicity at all, can be decomposed into a family of strongly monotone subsets. This is analogous to the situation depicted in parts (b) and (c) in Figure 5.1. Therefore, the central ideas explored in this part of research are how to decompose a non-monotonic dataset into a family of strongly monotonic subsets and how to derive classification models from the smaller, but strongly monotonic, subsets. The second task is how to determine a procedure for combining the individual classification models and build a single model for the original dataset. The final task is to determine under what conditions the proposed

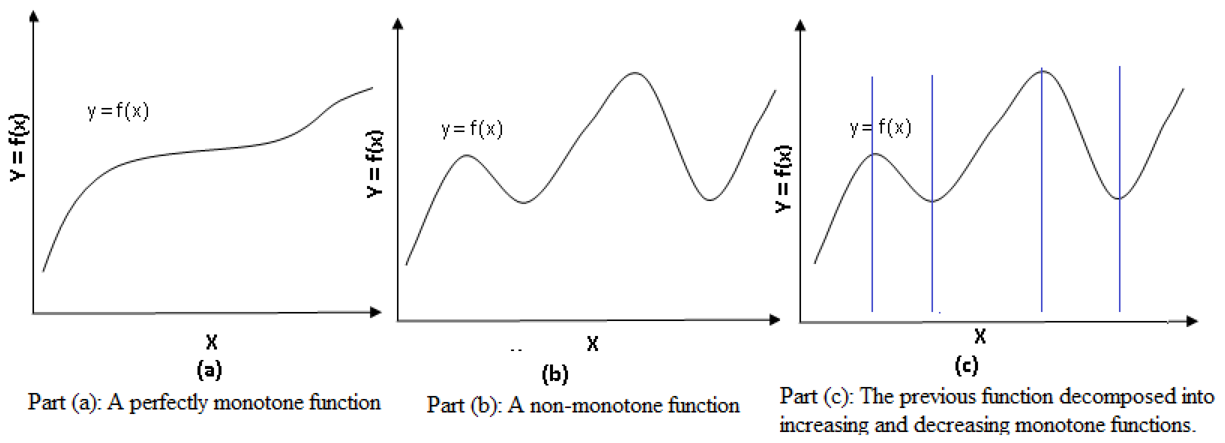

FIGURE 5.1: Different cases of scenarios of the monotonicity observed in functions.

decomposition approach is beneficial. For instance, does the number of derived monotone subsets, their relative sizes, and so on, play a detrimental role in determining whether the proposed approach might contribute to more accurate classifications?

5.2 Data Pre-processing

The ultimate goal of this approach is to always train the classification models on “easy” datasets. Therefore, the following P_{key} parameter becomes the most important criterion in doing the data manipulation (it is also the same as parameter P_3 in Table 4.1). As it is also indicated in Figure 4.2, in order to make this monotonic property to be meaningful, at least 4% of all possible distinct pairs of vectors should be monotonically related.

$$P_{key} = \frac{\text{Number of CMP pairs}}{\text{Number of distinct positive-negative pairs of vectors}}$$

In order to further explore the role of the single parameter P_{key}, six numeric datasets collected from the UCI Machine Learning Repository are considered. These datasets were used in an extensive pilot study, they have 600 to 4,800 vectors (data points) and come from a wide spectrum of application domains. The chosen datasets are the *Abalone*, the *Yeast*, the *Blood donation*, the *Car evaluation*, the *Auto_MPG*, and the *Mammo*. In the experiments, each derived experimental dataset had more than 4% pairs of vectors that were monotonically related. Thus, the monotonic properties in them would be meaningful and potentially significant.

The previous six datasets were used to build numerous two-class datasets for testing purposes in this pilot study. If a dataset was defined on more than two classes, two-class datasets were created as follows. First, two classes were selected randomly among the multiple classes to form two-class experimental datasets. Arbitrarily, one class was considered as the positive observations while the other was treated as the negative one. For instance, the Abalone dataset is originally defined on 29 classes. Thus the testing dataset with code name *Aba_5_7* indicates that class 5 is considered as the positive data while class 7 as the negative data.

A different way to form positive and negative data was employed by aggregating classes together. For instance, the dataset with code name *Aba_5Î1_7Î2* indicates that the data collected from classes 5 and 11 were aggregated to form the positive data while data collected from classes 7 and 12 were aggregated to form the negative data.

Some datasets were originally defined on two classes. Then, testing datasets were generated by randomly selecting a predetermined portion of data from the original dataset. For instance, from the two-class *Blood donation* dataset, random selections of 65% of the original data were used without replacement to form such testing datasets. These datasets were denoted as *Blood_1*, *Blood_2*, ..., *Blood_50*. Besides the *Blood donation* dataset, the datasets derived from the *Car evaluation*, the *Auto_MPG*, the *Mammo* and the *Yeast* datasets were treated in a similar manner.

The *Car evaluation* dataset has some ordinal attributes. Thus, equivalent numerical values were used for such attribute values to reflect the appropriate order. For instance, attribute values equal to {very high, high, med, low} were replaced by the numbers {4, 3, 2, 1}, respectively.

An effort was also made to create datasets of various representative degrees of difficulty. To see how this was done, one needs to consider two extreme cases: One with a balanced two-class dataset (i.e., one with almost equal number of positive and negative data points) and one with highly skewed data (for instance, by far more positive data points than negative ones). Then, the question is: which of these two scenarios is more likely to have many CMP pairs? The answer seems to be under the first scenario, and the computational studies described next seem to support

TABLE 5.1: Some Characteristics of the Experimental Datasets.

Original datasets	The number of experimental datasets	Range of the average accuracy when they were classified by multiple classifiers
Abalone	400	53.62% to 92.54%
Blood	600	57.23% to 92.37%
Yeast	400	55.02% to 91.83%
Car	400	80.42% to 92.53%
Auto_MPG	300	51.54% to 91.29%
Mammo	400	56.85% to 92.47%

this argument. Eventually, testing datasets of various representative P_{key} values (that reflect the percentage of vectors involved in CMP pairs) were generated (see also Figure 5.2).

The testing classifiers and classification tools used in this part of the study were the same as the ones described in Section 3.3. We used the previous experimental environment to analyze these two-class datasets, too.

5.3 The Pilot Study

The computational experiments were run in two steps. First, the experimental datasets were analyzed by the classifiers listed in the previous section. The average classification accuracies derived in this way were considered to be their classification difficulties (i.e., the learnability values). Next, at the second step, the CMP pairs were identified and the value of the P_{key} parameter was computed.

It should be stated here that the above approach is a heuristic one. If a different set of training data collected from the same orgininal dataset is analyzed this way, then it is possible that some attributes may be classified differently as positive and negative. However, the more representative a training dataset is to the original dataset, the more likely is that the attributes will be classified more accurately.

Table 5.1 lists some details of the experimental datasets used in this pilot study. Next, the difficulties of some of the experimental datasets, together with their key monotonic property denoted by the parameter P_{key}, were plotted together in 2-D graphs as shown in Figure 5.2. Each graph

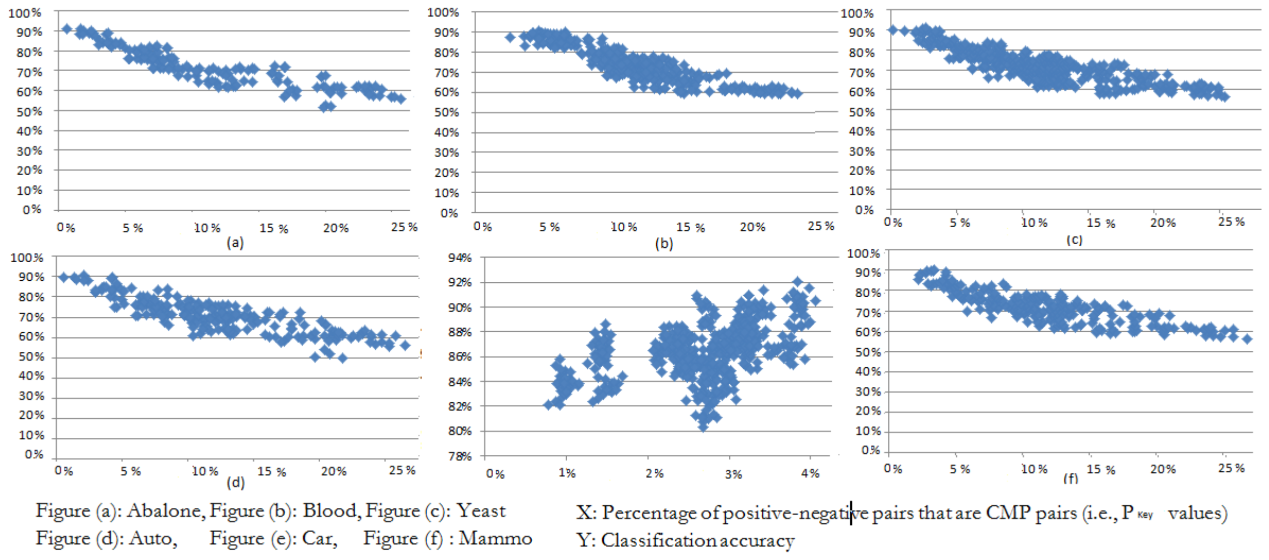

FIGURE 5.2: The relationship between the difficulty of the datasets and P_{key} values.

presents the experimental results observed by analyzing a group of datasets as described in the previous section.

As one can observe from this figure, when the number of CMP pairs increases, the datasets become more difficult to be accurately classified. This is quite clear by observing the plots of the results by analyzing the *Abalone*, the *Blood donation*, the *Yeast*, the *Auto_MPG*, and the *Mammo* based datasets. In these plots one can also observe that there are results for a wide spectrum of values on the horizontal axis. The classification accuracies for these datasets range from rather low values to very high ones. On the other hand, when datasets based on the *Car evaluation* dataset are considered, then almost always the classification accuracies are high (more than 80%). This means that these subsets are always easy ones. As one can observe from this plot, the *Car evaluation* based experimental datasets all have very few CMP pairs (less than 4%). This is another aspect that supports the previously detected tendency that easy datasets tend to have very few CMP pairs.

5.4 The Proposed Approach to Improve Classifications

The previous pilot computational results suggest that datasets with few CMP pairs are easier to analyze than those with more CMP pairs. Therefore, a reasonable idea is to try to partition a dataset into two groups of positive and negative subsets such that pairs of positive-negative subsets have no or very few CMP pairs. This approach is described formally as Algorithm 6. Before this approach is used, one first needs to determine the positive and the negative attributes as described in the Section 2.4.1. The proposed approach is comprised of two main steps that are iterated until the original dataset is completely partitioned. For a given dataset D this is described as follows:

1. Find out all positive vectors in D which either precede or are unrelated to all negative vectors. Next, they are removed from D to form a positive subset. If no such positive vectors exist, then we create an empty positive subset.

2. From the remaining vectors, find out all negative vectors which either precede or are unrelated to the rest of the positive vectors. Next, they are removed from D to form a negative subset. If no such positive vectors exist, then we create an empty negative subset.

3. Repeat steps 1 and 2 until the training dataset D becomes empty.

This approach is of O(N^2) time complexity, where N is the number of vectors in the training dataset D. Moreover, at the end of this approach the training dataset D has been partitioned into two groups of 1-class subsets. The first group of such 1-class subsets is comprised of all the positive subsets while the other group is comprised of all the negative ones. Depending on the dataset used, some of the previous subsets may end up to be empty. The positive subsets will be denoted as $Gp_1, Gp_2, \ldots Gp_m$, while the negative ones will be denoted as $Gn_1, Gn_2, \ldots Gn_m$.

Furthermore, according to the way such subsets are generated, for every pair of positive subsets Gp_i and Gp_j, where $j > i$, the vectors in Gp_i always either precede or are unrelated to all the vectors in Gp_j. A similar relationship exists for any pair of negative subsets Gn_i and Gn_j, where $j > i$. Also, for a pair of subsets Gp_i and Gn_i the vectors in Gp_i always either precede or are unrelated to all the vectors in Gn_i.

5.5 A Monotonicity-Based Classification Approach

The previous section described how a dataset can be decomposed into two groups of one-class subsets. These subsets are alternating, for instance, a positive subset is followed by a negative one,

Algorithm 6: A monotonic-based approach to partition a dataset

```
Input  : Dataset $D$
Output: $PositiveSets, NegativeSets$ /*PositiveSets and NegativeSets are the sets of the
        positive and negative subsets, respectively*/
1  $PositiveSets \leftarrow \phi, NegativeSets \leftarrow \phi$;
2  $E^+ \leftarrow$ all positive vectors in dataset $D$;
3  $E^- \leftarrow$ all negative vectors in dataset $D$;
4  while $E^+ \neq \phi$ or $E^- \neq \phi$ do
5      $Subset^+ \leftarrow \phi, Subset^- \leftarrow \phi$;
6      for each $e_i^+ \in E^+$ do
7          if There is no $e_j^- \in E^-$ such that $e_j^- \succ e_i^+$ then
8              $Subset^+ \leftarrow Subset^+ \cup e_i^+$;
9              Remove $e_i^+$ from $E^+$;
10         end
11     end
12     for each $e_i^- \in E^-$ do
13         if There is no $e_j^+ \in E^+$ such that $e_j^+ \succ e_i^-$ then
14             $Subset^- \leftarrow Subset^- \cup e_i^-$;
15             Remove $e_i^-$ from $E^-$;
16         end
17     end
18     $PositiveSets \leftarrow PositiveSets \cup Subset^+$ ;
19     $NegativeSets \leftarrow NegativeSets \cup Subset^-$;
20 end
21 Return PositiveSets, NegativeSets;
```

which is followed by a positive subset, and so on. Also, any pair of positive and negative subsets can form a two-class experimental dataset which contains no CMP pairs (by switching the class values of the vectors, if necessary). Therefore, according to the previous analyses, classification models derived from such experimental datasets should be more accurate in classifying their data points rather than the model derived from the original (and larger) dataset. However, a new critical question needs to be considered at this point. Given a new vector (data point) of unknown class value, which model(s) should classify it? There are many models now derived from the new generated subsets and the original dataset.

In order to answer this question, one first needs to examine the nature of the vector to be classified and somehow determine which model (or models) is (are) the most appropriate to classify it. For this, one needs to determine which subset(s) of training data (derived after the previous decomposition approach has been applied) is (are) the most closely related to this new vector. Such notion of "the most closely related" will be treated under the scope of monotonicity. In the following considerations a new vector is assigned to one of three mutually exclusive and exhaustive categories. These three categories are denoted as Type I, Type II and Type III category of testing vectors. Algorithm 7 describes how this categorization can be done.

Type I vectors

A new vector V is placed in this category if and only if it can be ***covered*** by a single subset G (which is either a positive or a negative one). In this study, the subset G is said to **cover** V if and only if G contains two vectors P and Q such that $P \succeq V \succeq Q$. According to the way these subsets are generated, at most one subset can be found to satisfy this criterion.

Type II vectors

Suppose that a new vector V is not of Type I as defined above. However, if two vectors P and Q can be identified such that $P \in G_i$ and $Q \in G_j$, where G_i and G_j are two subsets with the same class value where $j > i$, and $P \succeq V \succeq Q$, then the vector V is called to be of Type II.

Type III vectors

If a new testing vector V is neither a Type I nor a Type II testing vector, then it is called to be a Type III testing vector.

The previous three categories for grouping a new testing vector are used to design the classification process for assigning a class value to it. This is explained in the following subsections.

5.5.1 How to Classify Type I Testing Vectors

If a testing vector V has been assigned to the Type I category, that means it is covered by a single subset G, where G can be either a positive subset or a negative one.

First, suppose that G is a positive subset. Then, when one of the negative subsets is considered along with this positive subset G, one can form a two-class experimental dataset which possesses

Algorithm 7: Identify the types of the testing vectors.

```
Input  : Positive training subsets $G_p$, negative training subsets $G_n$, and a testing vector $V$
Output: Type of $V$
1  for each subset $G$, $G \in G_p$ or $G \in G_n$ do
2      if There are two vectors $P$ and $Q$ in $G$, such that $P \succeq V \succeq Q$ then
3          Return $V$ as a Type I vector;
4      end
5  end
6  if There exist two subsets $G_i$, $G_j$, and either $G_i$, $G_j \in G_p$ or $G_i$, $G_j \in G_n$. If one can find a
   vector $P \in G_i$ and a vector $Q \in G_j$ such that $P \succeq V \succeq Q$ then
7      Return $V$ as a Type II vector;
8  else
9      Return $V$ as a Type III vector;
10 end
```

two important properties: (a)Its positive subset has two positive vectors P and Q and $P \succeq V \succeq Q$, and (b) it contains no CMP pairs. Thus, according to previous discussions, the classification model derived from this experimental dataset would be more accurate in classifying the data point V.

Hence, the next step is to consider all such combinations (G, G_{ni}) for $i = 1,2,3,\ldots,m$, and apply the original classification algorithm (i.e., Decision Tree, ANN, SVM, etc. classification approach) and derive m classification models. The class value of the testing vector V can be determined by taking the majority vote among these m models. The fact that the testing vector V can be placed between two training vectors (i.e., P and Q) which both are positive does not necessarily mean that the vector V should also be classified as positive by each one of the m models.

The second possibility is when the vector V is located between two negative training vectors. Now the subset G is comprised of negative training data. Then it should be treated in an analogous manner. In this case there are m two-class experimental datasets denoted as (G_{pi}, G) for $i = 1,2,3,\ldots,m$. The rest of the steps are the same as in the previous case. This process is formally described as Algorithm 8.

5.5.2 How to Classify Type II Testing Vectors

The methodology for classifying Type II testing vectors is similar to that for classifying Type I vectors. That is, a given base classifier is applied to analyze a group of training subsets and the

Algorithm 8: The approach to classify type I testing vectors.

```
Input  : Two groups of one-class training subset $G_p$ and $G_n$, testing vector $V$, classification
         algorithm $C$
Output: Class value of $V$
1  Classifier_List ← $\phi$ ;
2  Find out one-class training subset $G$, such that there exist two vectors $P$ and $Q$ in $G$, and
   $P \succeq V \succeq Q$;
3  if $G \in G_p$ then
4      for each one-class subset $G_{ni} \in G_n$ do
5          $S = G \cup G_{ni}$ ;
6          Classifier_List ← Apply $C$ to analyze $S$;
7      end
8  else
9      for each one-class subset $G_{pi} \in G_p$ do
10         $S = G \cup G_{pi}$ ;
11         Classifier_List ← Apply $C$ to analyze $S$;
12     end
13 end
14 Use all the classifiers in the Classifier_List to classify $V$ and perform majority voting on the
   classification results;
15 Return the result of the majority voting;
```

derived classification models are used to classify the testing vector V. By performing the majority vote on the classification results, the class value of V can be therefore determined.

The next question is, how to generate such classification models? A Type II testing vector cannot be covered by a single one-class subset. However, it can always be covered by the combination of multiple same-class subsets. In other words, one can always find two subsets which have the same class value (G_{pi} and G_{pj}, or G_{ni} and G_{nj}, for example), such that there exist a vector $P \in G_{pi}(G_{ni})$ and a vector $Q \in G_{pj}(G_{nj})$ where $i < j$ and $P \succ V \succ Q$.

Therefore, based on the values of i and j, one can form a training subset S where $S = G_{pi} \cup G_{ni} \cup G_{p(i+1)} \cup G_{n(i+1)} \cup \cdots \cup G_{pj} \cup G_{nj}$, and it is easy to find out that S covers V. In this study, the desired training subset S should be comprised by the subsets where $j - i$ is minimized. In other words, the goal is to find out the fewest subsets whose combination can cover the vector V.

By applying the base classifier C to analyze S, a classification model can be therefore generated. Furthermore, more sub classification models can be generated by using the same classifier C to analyze the combination of S and the rest of the training data. For instance, a training subset S^0 can be generated as $S^0 = S \cup G_{pk} \cup G_{nk}$, where $k < i$, another subset S^1 can be generated as $S^1 = S \cup G_{pk} \cup G_{nk} \cup G_{p(k+1)} \cup G_{n(k+1)}$, where $k > j$, and so on. There are 2^{num} number of such combinations where $num = 2 \times (m - j + i)$. For reason of simplicity, the other training subsets were generated by combining pairs of unused positive-negative training subsets with S, one at a time. That is, the training subsets $S_k = S \cup G_{pk} \cup G_{nk}$, where $0 \leq k < i$ or $j < k \leq m$. A total of $m - j + i - 0 + 1$ (including the set S itself) number of training subsets were used to generate the desired learners.

To summarize, a Type II testing vector V should be classified as follows:

1) Find out the training subset S, $S = G_{pi} \cup G_{ni} \cup G_{p(i+1)} \cup G_{n(i+1)} \cup \cdots \cup G_{pj} \cup G_{nj}$, where $0 \leq i < j \leq m$. S should cover vector V and the value of $j - i$ is minimized.

2) For all other pairs of positive-negative subsets Gp_k and Gn_k, they are combined with S, one at a time, to form a group of training subsets. That is, $S_k = S \cup Gp_k \cup Gn_k$ for all $0 \leq k < i$ or $j < k \leq m$.

3) Analyze such S_k and S subsets, compare their P_{key} parameter values to that of the entire training data, and select only the ones which have smaller P_{key} values. According to the key observation described in Section 2, they are more likely to result in more accurate classifiers.

4) Use the base classifier on all selected training subsets. The derived classification models are used to classify the testing vector V. The class value of V is determined by the result of the majority vote.

It should be noted that the training datasets generated in this way may contain some CMP pairs. That is why they should be compared to the original datasets in terms of their P_{key} values. The motivation for having step 3, is to have subsets that are more likely to lead to more accurate classifiers than the original dataset as a whole. The entire process is formally illustrated in Algoirthm 9.

Algorithm 9: The approach to classify Type II testing vectors.

```
Input : Two groups of one-class training subsets $G_p$ and $G_n$, testing vector $V$, classification
        algorithm $C$, the value of monotonic property $P_{key}$ in the original training dataset,
        denoted as $P_{key-original}$.
Output: Class value of $V$
1  Determine $i, j$ to form a subset $S$, $S \leftarrow G_{pi} \cup G_{ni} \cup G_{p(i+1)} \cup G_{n(i+1)} \cdots \cup G_{pj} \cup G_{nj}$, such that $S$
   covers $V$ and $j - i$ is minimized;
2  Classifier_List ← the classifier learned from applying $C$ to analyze $S$ ;
3  for each $k$,$1 \leq k < i$ or $j < k \leq m$ do
4      $S_k = S \cup G_{pk} \cup G_{nk}$;
5      if The value of monotonic property $P_{key}$ derived from $S_k$ is smaller than $P_{key-original}$ then
6          Classifier_List ← Classifier_List ∪ the classifier learned from applying $C$ to analyze
           $S_k$;
7      end
8  end
9  Use all the classifiers in Classifier_List to classify $V$, perform majority voting on the
   classification results;
10 Return majority voting result;
```

5.5.3 Minimum Size of the Derived Training Subsets

The two families of subsets derived as described earlier, play a key role in the quest for improving classification accuracy. However, it is possible occasionally such subsets to be of too small size. If subsets of too small size are used, then the benefits of using purely monotonic subsets for inferring highly accurate classification models from training data may be canceled. This is possible because too small subsets may not be as representative as larger subsets. Such problems happen with most methods that infer models from data. In this study a limit was used for the minimum size of such subsets. This limit was determined empirically, and it was set at 30. That is, if a subset had less than 30 members then it was ignored.

Chapter 6

Experiments For Meta-Learning Approach

6.1 Some Preliminaries on the Experiments

Some experiments were designed and performed to test the performance of the proposed approach. The experimental datasets used in these experiments were as follows: a) All the experimental datasets mentioned in Section 2.2.1; a total of 2,500 datasets were used. b) 1,800 massive datasets. The latter ones were generated to simulate some difficult datasets. They were created by randomly selecting datasets from part a). Next, the vectors in these selected datasets were randomly assigned to class values “positive” or “negative” with probability equal to 0.50. They are grouped by their dimensions into three groups denoted as *Mass_5,Mass_6 and Mass_7* in this study.

Moreover, the proposed approach is a meta-learning approach. Thus, any classification algorithm can be used as the base classifier *C* in Algorithms 8 and 9. Four widely used classification algorithms were used as the base classifiers in this study. These are: a *decision Tree*, a *support vector machine* (SVM), an *ADTree*, and an *artificial neural network*(ANN). Moreover, 10 rounds of cross validations were implemented in the classifications to decrease the variance.

It is noticed that the ratio of the testing and training vectors is an important factor that impacts the classification accuracy. Generally speaking, with more vectors selected for training, the generated classification model is expected to be more complete and therefore more accurate. Currently, the setting of this ratio is empirical. In order to find out a reasonable ratio, several experimental datasets were randomly selected and analyzed to study the impact of this factor.

In this study, each selected dataset was partitioned into a group of training and testing subsets. The ratios of training size over testing size for each dataset in each partition were varied from 1% to 99%. In each scenario, the datasets were first analyzed by simply implementing the *dicision tree*. After that, they were analyzed by the proposed approach using the *decision tree* as the base clas-

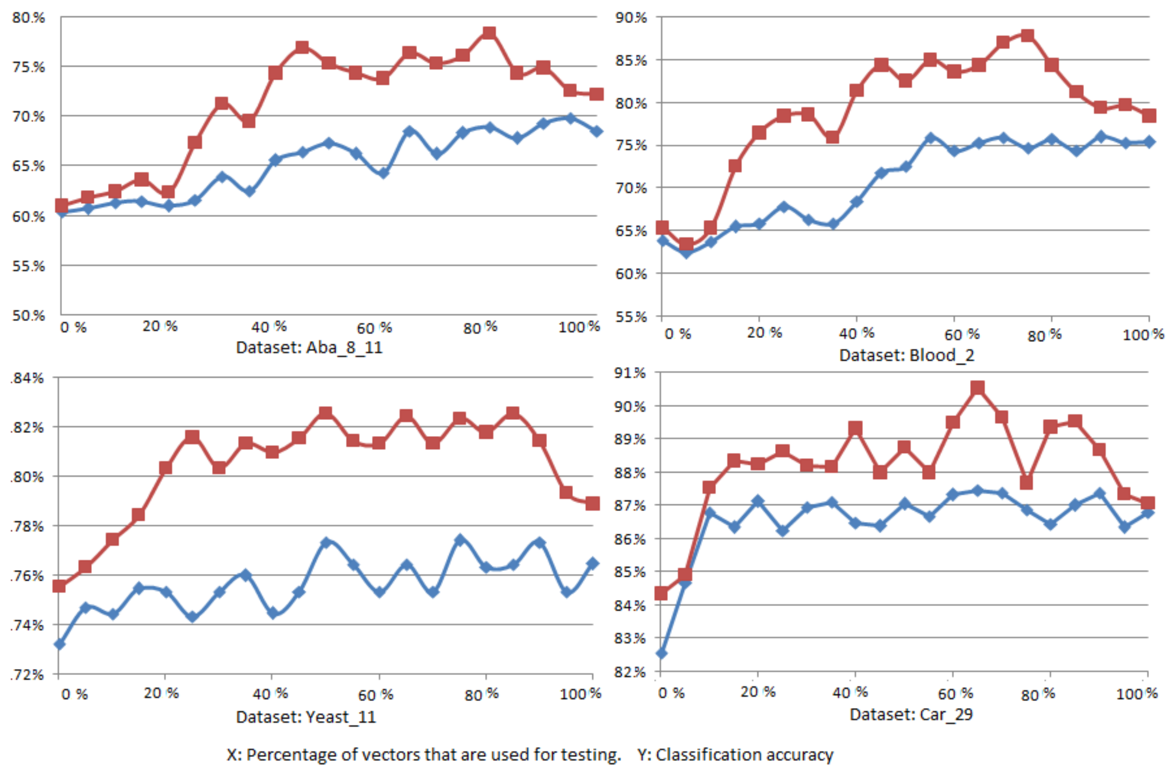

FIGURE 6.1: Experimental results with different ratios of training and testing vectors.

sifier. In these classifications, a 10-cross validation approach is applied to reduce the classification variance. The experimental results are plotted in Figure 6.1.

The graphs listed in Figure 6.1 are organized as follows. In each graph the X-coordinate indicates the percentage of vectors that were selected for training, and the Y-coordinate indicates the classification accuracies when analyzing testing vectors under various cases. The datasets used in this figure are: *Aba*_5_11, *Yea*_1_4, *Blood*_2, and *Car*_17. As one can observe from Figure 6.1, the classification improvements become stable when the ratios of the training data reside in the range of (20%, 85%). Therefore, in these experiments, 60% of the vectors were randomly selected for training and the rest were used for testing.

6.2 The Experimental Results

In order to better evaluate the effectiveness of the proposed approach, its performance was compared to some other meta-learning approaches, such as the *Bagging* [5],*Boosting* [16], and *HBA* [35]. In this study, each experimental dataset was analyzed under the following scenarios:

1. It was analyzed by using a base classification algorithm C. The corresponding classification accuracy is denoted as ACC_{c-name}.

2. It was analyzed by the proposed approach using C as the base classifier. The corresponding classification accuracy is denoted as $ACC_{Proposed}$.

3. It was analyzed by the *Bagging* approach using C as the base classifier. The corresponding classification accuracy is denoted as $ACC_{Bagging}$.

4. It was analyzed by the *Boosting* approach using C as the base classifier. The corresponding classification accuracy is denoted as $ACC_{Boosting}$.

5. It was analyzed by the *HBA* approach using C as the base classifier. The corresponding classification accuracy is denoted as ACC_{HBA}.

6. It was analyzed by using C as the base classifier, its Type I and Type II vectors were analyzed by the proposed approach, and its Type III vectors were classified by the HBA approach. The corresponding classification accuracy is denoted as $ACC_{Combined}$.

7. Its relative maximum possible improvement rate (denoted as $Improvement_{RMP1}$) was calculated. This parameter indicates the relative maximum improvement that the proposed approach achieves over the maximum possible improvement. For example, as shown in Table 6.2, when one analyzes the *Blood_42* dataset using *SVM* as the base classifier, the proposed approach can obtain an improvement of 79.2% - 63.5% = 15.7% with respect to the classification accuracy compared with the base classifier. However, it is noticed that the maximum possible improvement that one may obtain by simply applying *SVM* to classify *Blood_42* is 100%-63.5% = 36.5%. Therefore, the relative maximum possible improvement rate by using the proposed approach should be 15.7% / 36.5% = 43.0%. This is a more descriptive way to assess the impact of the proposed approach.

8. In a similar manner the relative maximum possible improvement was calculated when the ACC_{c-name} is compared to the $ACC_{Combined}$ value. This relative improvement value is denoted as $Improvement_{RMP2}$ in Tables 6.1 to 6.8.

Some experimental results are provided in Tables 6.1 to 6.8, while the datasets *Abalone*, *Blood*,*Yeast*, *Car*, *Auto_MPG*, *Mammo*, together with the 5, 6, and 7-attribute artificial datasets were used as the training data, respectively. Each of these tables is organized in the following format. The first row of the table lists the names of the experimental datasets. From the second row to the bottom of the table, every eight rows comprise a block that displays the experimental results of analyzing the corresponding experimental dataset under different classification scenarios. There are four such blocks in each table which are separated by a narrow blank gap, and each of them uses a different classification algorithm as the base classifier.

According to the experimental results, when one compares the results obtained by using the base classifier alone with the results obtained by using the proposed approach, the proposed approach always outperforms the stand-alone classifiers. When one examines the results for the *Abalone*, *Blood donation*, *Yeast*, *Auto_MPG* and *Mammo* datasets, the proposed method can frequently achieve more than 10% improvement in classification accuracy. Such improvements can be as high as 21%. An example is the case of analyzing the *Blood_2* dataset by using *SVM* as the base classifier (Table 6.2). In terms of the relative maximum possible improvement measures, the values of their $Improvement_{RMP1}$ are mostly distributed in the range of (10%, 55%), while their $Improvement_{RMP2}$ values are mostly ranged in (20%, 75%). In the case of analyzing the *Car* datasets, the obtained classification improvements were less significant. However, the improvements are always present and frequently are more than 2%, in terms of the relative maximum possible improvement measures, their $Improvement_{RMP1}$ values are consistently more than 10% and their $Improvement_{RMP2}$ values are consistenly more than 20%, as are the cases of classifying other datastes. Why sometimes the improvements are significant and some times are not, is discussed in the next paragraph.

TABLE 6.1: Some experimental results when analyzed the *Abalone* datasets.

Datasets	Aba_7_11	Aba_58_9	Aba_67_1114	Aba_612_8	Aba_511_9
$ACC_{DecisionTree}$	77.6%	64.6%	90.8%	70.8%	67.4%
$ACC_{Proposed}$	83.3%	77.0%	92.8%	79.1%	75.3%
$ACC_{Bagging}$	76.2%	66.2%	90.8%	73.2%	68.7%
$ACC_{Boosting}$	77.1%	63.7%	91.2%	71.4%	68.7%
ACC_{HBA}	84.3%	80.7%	93.2%	82.7%	78.1%
$ACC_{Combined}$	87.1%	83.4%	94.1%	86.8%	84.7%
$Improvement_{RMP1}$	29.4%	35.0%	21.7%	28.4%	24.2%
$Improvement_{RMP2}$	42.4%	53.1%	35.8%	54.7%	53.0%
ACC_{SVM}	77.3%	65.7%	87.7%	71.4%	70.2%
$ACC_{Proposed}$	80.6%	80.7%	90.3%	77.4%	76.3%
$ACC_{Bagging}$	76.2%	66.4%	87.7%	72.1%	69.7%
$ACC_{Boosting}$	78.3%	65.7%	87.7%	72.9%	70.9%
ACC_{HBA}	82.7%	82.1%	92.3%	80.2%	79.4%
$ACC_{Combined}$	85.6%	86.4%	94.8%	85.5%	83.4%
$Improvement_{RMP1}$	14.5%	43.7%	21.1%	20.9%	20.4%
$Improvement_{RMP2}$	36.6%	61.8%	45.8%	56.8%	49.7%
ACC_{ADTree}	76.1%	64.4%	90.4%	66.4%	67.0%
$ACC_{Proposed}$	79.9%	75.4%	92.6%	74.8%	74.5%
$ACC_{Bagging}$	77.4%	65.7%	90.4%	67.2%	69.5%
$ACC_{Boosting}$	75.8%	66.2%	90.4%	69.1%	67.9%
ACC_{HBA}	82.5%	79.7%	93.2%	78.7%	78.4%
$ACC_{Combined}$	86.4%	85.3%	94.9%	84.8%	85.8%
$Improvement_{RMP1}$	15.9%	30.9%	22.9%	25.0%	22.7%
$Improvement_{RMP2}$	43.1%	58.7%	46.9%	54.8%	57.0%
ACC_{ANN}	74.8%	64.4%	89.0%	73.2%	68.9%
$ACC_{Proposed}$	79.8%	76.4%	91.6%	78.5%	76.4%
$ACC_{Bagging}$	73.9%	64.7%	89.0%	75.8%	70.1%
$ACC_{Boosting}$	74.8%	65.7%	89.0%	74.9%	72.7%
ACC_{HBA}	82.9%	79.3%	93.8%	80.0%	79.4%
$ACC_{Combined}$	87.6%	84.4%	95.9%	84.8%	86.2%
$Improvement_{RMP1}$	19.8%	33.7%	23.6%	19.8%	24.1%
$Improvement_{RMP2}$	50.8%	56.2%	62.7%	43.3%	55.6%

TABLE 6.2: Some experimental results when analyzed the *Blood donation* datasets.

Datasets	Blood_2	Blood_13	Blood_35	Blood_42	Blood_47
$ACC_{DecisionTree}$	64.6%	72.5%	65.7%	68.2%	80.9%
$ACC_{Proposed}$	83.1%	79.2%	74.1%	81.3%	81.7%
$ACC_{Bagging}$	67.2%	73.5%	67.2%	69.5%	80.9%
$ACC_{Boosting}$	65.3%	72.8%	67.2%	68.9%	80.9%
ACC_{HBA}	87.6%	82.2%	78.7%	82.7%	84.6%
$ACC_{Combined}$	91.3%	85.8%	83.4%	85.3%	85.9%
$Improvement_{RMP1}$	52.2%	24.4%	24.5%	41.2%	4.19%
$Improvement_{RMP2}$	75.4%	48.4%	51.6%	53.8%	26.2%
ACC_{SVM}	61.4%	65.7%	63.9%	63.5%	77.7%
$ACC_{Proposed}$	82.7%	77.7%	70.0%	79.2%	82.3%
$ACC_{Bagging}$	60.8%	68.1%	63.9%	65.7%	75.9%
$ACC_{Boosting}$	61.7%	67.4%	63.9%	64.2%	76.8%
ACC_{HBA}	81.6%	82.9%	73.7%	81.9%	83.4%
$ACC_{Combined}$	83.3%	87.3%	79.5%	86.2%	87.3%
$Improvement_{RMP1}$	55.1%	35.0%	16.9%	43.0%	20.6%
$Improvement_{RMP2}$	49.4%	50.2%	35.9%	56.5%	35.9%
ACC_{ADTree}	67.0%	74.5%	68.0%	68.3%	80.2%
$ACC_{Proposed}$	85.1%	81.0%	76.4%	82.4%	84.1%
$ACC_{Bagging}$	68.1%	74.5%	69.7%	69.5%	80.2%
$ACC_{Boosting}$	67.7%	74.5%	66.8%	68.3%	80.7%
ACC_{HBA}	86.9%	82.4%	79.2%	81.7%	85.6%
$ACC_{Combined}$	92.7%	86.5%	85.8%	84.4%	87.4%
$Improvement_{RMP1}$	54.8%	25.5%	26.3%	44.5%	19.7%
$Improvement_{RMP2}$	77.9%	47.1%	55.6%	50.8%	36.4%
ACC_{ANN}	63.3%	71.7%	67.9%	67.0%	80.2%
$ACC_{Proposed}$	82.1%	78.6%	74.8%	78.1%	83.2%
$ACC_{Bagging}$	62.8%	72.6%	67.9%	66.4%	80.7%
$ACC_{Boosting}$	64.7%	71.7%	66.8%	68.7%	80.9%
ACC_{HBA}	84.3%	80.5%	77.7%	82.5%	85.2%
$ACC_{Combined}$	90.7%	85.7%	84.6%	86.2%	86.6%
$Improvement_{RMP1}$	51.2%	24.4%	21.5%	33.6%	15.2%
$Improvement_{RMP2}$	74.7%	49.5%	52.0%	58.2%	32.3%

TABLE 6.3: Some experimental results when analyzed the *Yeast* datasets.

Datasets	Yea_3	Yea_17	Yea_21	Yea_28	Yea_49
$ACC_{DecisionTree}$	64.1%	68.0%	73.6%	78.7%	83.6%
$ACC_{Proposed}$	69.4%	76.1%	79.6%	85.4%	85.3%
$ACC_{Bagging}$	65.7%	67.6%	73.5%	79.8%	83.6%
$ACC_{Boosting}$	64.1%	68.7%	74.4%	78.8%	83.6%
ACC_{HBA}	72.3%	79.9%	82.6%	87.5%	87.4%
$ACC_{Combined}$	78.5%	83.3%	85.5%	88.6%	89.5%
$Improvement_{RMP1}$	14.7%	25.3%	22.7%	31.5%	10.3%
$Improvement_{RMP2}$	40.1%	47.8%	45.1%	46.5%	36.0%
ACC_{SVM}	63.9%	67.1%	71.8%	78.5%	82.9%
$ACC_{Proposed}$	69.8%	75.7%	78.6%	84.4%	85.6%
$ACC_{Bagging}$	64.1%	68.2%	70.6%	79.6%	83.3%
$ACC_{Boosting}$	63.9%	67.6%	73.3%	76.8%	81.7%
ACC_{HBA}	72.8%	78.3%	81.8%	87.0%	87.2%
$ACC_{Combined}$	77.4%	80.4%	84.9%	87.8%	89.8%
$Improvement_{RMP1}$	16.3%	26.1%	24.1%	27.4%	15.8%
$Improvement_{RMP2}$	37.3%	40.4%	46.4%	43.3%	40.3%
ACC_{ADTree}	61.4%	69.2%	72.6%	79.5%	83.4%
$ACC_{Proposed}$	68.8%	75.4%	79.7%	82.9%	86.1%
$ACC_{Bagging}$	60.7%	69.2%	73.6%	79.8%	83.3%
$ACC_{Boosting}$	59.8%	68.6%	71.4%	79.5%	83.4%
ACC_{HBA}	71.9%	79.9%	82.1%	86.5%	89.4%
$ACC_{Combined}$	76.6%	84.9%	85.9%	87.6%	91.2%
$Improvement_{RMP1}$	19.2%	20.1%	25.9%	16.6%	16.2%
$Improvement_{RMP2}$	39.3%	50.9%	48.5%	39.5%	47.0%
ACC_{ANN}	62.0%	66.7%	73.4%	76.4%	82.5%
$ACC_{Proposed}$	68.6%	75.7%	79.5%	83.3%	84.8%
$ACC_{Bagging}$	63.4%	66.2%	71.2%	76.7%	81.2%
$ACC_{Boosting}$	62.0%	67.6%	74.8%	78.5%	80.7%
ACC_{HBA}	72.4%	78.3%	83.8%	87.0%	86.4%
$ACC_{Combined}$	76.6%	80.4%	86.9%	89.8%	88.5%
$Improvement_{RMP1}$	17.4%	27.1%	22.9%	29.2%	13.1%
$Improvement_{RMP2}$	38.4%	41.1%	50.7%	56.8%	34.3%

TABLE 6.4: Some experimental results when analyzed the *Car evaluation* datasets.

Datasets	Car_4	Car_21	Car_36	Car_39	Car_48
$ACC_{DecisionTree}$	89.2%	88.9%	86.4%	85.0%	84.3%
$ACC_{Proposed}$	91.0%	89.6%	88.2%	86.5%	87.3%
$ACC_{Bagging}$	89.2%	89.1%	86.4%	85.7%	83.9%
$ACC_{Boosting}$	89.2%	88.9%	86.4%	85.0%	84.3%
ACC_{HBA}	93.3%	91.5%	92.7%	88.5%	89.2%
$ACC_{Combined}$	94.6%	92.7%	93.6%	91.2%	92.5%
$Improvement_{RMP1}$	16.6%	6.30%	13.2%	10.0%	19.1%
$Improvement_{RMP2}$	50.0%	34.2%	52.9%	41.3%	52.2%
ACC_{SVM}	89.9%	88.5%	85.5%	85.0%	86.0%
$ACC_{Proposed}$	91.2%	89.3%	88.0%	87.7%	89.5%
$ACC_{Bagging}$	90.1%	87.8%	86.4%	85.7%	85.4%
$ACC_{Boosting}$	89.4%	88.5%	85.9%	85.0%	85.7%
ACC_{HBA}	93.3%	92.2%	91.7%	89.5%	90.0%
$ACC_{Combined}$	94.7%	93.5%	93.4%	90.8%	92.7%
$Improvement_{RMP1}$	12.9%	6.96%	17.2%	18.0%	25.1%
$Improvement_{RMP2}$	47.5%	43.5%	54.5%	38.7%	47.9%
ACC_{ADTree}	88.1%	87.7%	85.3%	79.3%	81.0%
$ACC_{Proposed}$	88.9%	89.0%	88.0%	81.1%	83.9%
$ACC_{Bagging}$	88.1%	88.2%	85.3%	80.7%	81.4%
$ACC_{Boosting}$	88.1%	87.7%	84.9%	80.7%	80.7%
ACC_{HBA}	92.3%	90.5%	89.7%	84.5%	85.2%
$ACC_{Combined}$	94.3%	94.7%	93.6%	87.5%	87.9%
$Improvement_{RMP1}$	6.72%	10.6%	18.4%	8.70%	15.3%
$Improvement_{RMP2}$	52.1%	56.9%	56.5%	39.6%	36.3%
ACC_{ANN}	88.9%	86.9%	85.0%	84.3%	82.7%
$ACC_{Proposed}$	89.7%	89.2%	87.0%	85.7%	85.7%
$ACC_{Bagging}$	88.6%	87.5%	86.4%	84.3%	82.7%
$ACC_{Boosting}$	89.4%	87.5%	86.4%	85.0%	82.7%
ACC_{HBA}	90.3%	90.5%	89.7%	87.4%	87.5%
$ACC_{Combined}$	91.7%	92.7%	91.6%	89.7%	88.3%
$Improvement_{RMP1}$	7.20%	17.6%	13.3%	8.91%	17.3%
$Improvement_{RMP2}$	25.2%	44.3%	44.0%	34.4%	32.3%

TABLE 6.5: Some experimental results when analyzed the *Auto_MPG* datasets.

Datasets	Auto_16	Auto_34	Auto_57	Auto_135	Auto_364
$ACC_{DecisionTree}$	67.5%	73.5%	75.6%	68.5%	82.5%
$ACC_{Proposed}$	78.4%	77.9%	79.8%	77.3%	86.6%
$ACC_{Bagging}$	68.7%	75.4%	76.7%	68.7%	83.3%
$ACC_{Boosting}$	69.4%	74.8%	74.3%	67.7%	83.6%
ACC_{HBA}	79.3%	77.6%	80.5%	77.5%	87.8%
$ACC_{Combined}$	82.2%	78.7%	81.7%	79.4%	89.8%
$Improvement_{RMP1}$	33.5%	16.6%	17.2%	27.9%	23.4%
$Improvement_{RMP2}$	45.2%	19.6%	25.0%	34.6%	41.7%
ACC_{SVM}	68.1%	74.8%	75.3%	68.2%	83.2%
$ACC_{Proposed}$	78.7%	78.5%	79.8%	78.8%	87.9%
$ACC_{Bagging}$	69.8%	75.4%	77.5%	69.3%	84.3%
$ACC_{Boosting}$	71.5%	75.9%	76.7%	68.4%	84.7%
ACC_{HBA}	79.8%	779.5%	79.5%	79.5%	88.9%
$ACC_{Combined}$	81.5%	80.1%	79.9%	79.9%	90.5%
$Improvement_{RMP1}$	33.2%	14.7%	18.2%	33.3%	28.0%
$Improvement_{RMP2}$	42.0%	21.0%	18.6%	36.8%	43.5%
ACC_{ADTree}	66.7%	73.7%	74.8%	67.5%	83.1%
$ACC_{Proposed}$	74.6%	77.5%	78.5%	76.9%	86.9%
$ACC_{Bagging}$	68.5%	73.5%	75.6%	67.5%	84.3%
$ACC_{Boosting}$	67.8%	74.8%	74.4%	68.9%	83.9%
ACC_{HBA}	76.3%	78.7%	80.6%	75.7%	88.9%
$ACC_{Combined}$	77.2%	79.4%	81.1%	77.8%	89.7%
$Improvement_{RMP1}$	23.7%	14.4%	14.7%	28.9%	22.5%
$Improvement_{RMP2}$	31.5%	21.7%	25.0%	31.7%	39.1%
ACC_{ANN}	67.4%	74.2%	76.2%	67.3%	82.7%
$ACC_{Proposed}$	78.5%	78.6%	81.5%	75.6%	86.2%
$ACC_{Bagging}$	68.7%	75.7%	77.6%	68.7%	83.3%
$ACC_{Boosting}$	69.5%	74.9%	75.7%	66.9%	82.8%
ACC_{HBA}	79.8%	78.9%	82.4%	75.9%	88.9%
$ACC_{Combined}$	81.5%	79.5%	83.3%	79.8%	90.1%
$Improvement_{RMP1}$	34.0%	17.1%	22.3%	25.4%	20.2%
$Improvement_{RMP2}$	43.3%	20.5%	29.8%	38.2%	42.8%

TABLE 6.6: Some experimental results when analyzed the *Mammo* datasets.

Datasets	Mammo_4	Mammo_28	Mammo_69	Mammo_81	Mammo_97
$ACC_{DecisionTree}$	74.5%	65.3%	72.6%	86.3%	78.6%
$ACC_{Proposed}$	81.2%	70.8%	79.6%	88.7%	84.5%
$ACC_{Bagging}$	76.5%	66.5%	73.3%	86.5%	82.4%
$ACC_{Boosting}$	75.3%	65.9%	72.8%	84.6%	83.1%
ACC_{HBA}	81.9%	71.2%	80.7%	87.8%	86.4%
$ACC_{Combined}$	82.3%	71.6%	81.4%	89.1%	87.7%
$Improvement_{RMP1}$	26.3%	15.9%	25.5%	17.5%	27.6%
$Improvement_{RMP2}$	30.6%	18.2%	32.1%	20.4%	42.5%
ACC_{SVM}	75.3%	68.5%	73.5%	87.1%	78.8%
$ACC_{Proposed}$	83.3%	73.4%	82.9%	87.6%	83.6%
$ACC_{Bagging}$	76.5%	70.1%	74.5%	87.1%	81.2%
$ACC_{Boosting}$	75.9%	69.8%	73.2%	87.1%	82.5%
ACC_{HBA}	84.6%	74.6%	83.5%	87.3%	85.4%
$ACC_{Combined}$	87.4%	74.9%	85.1%	87.9%	86.6%
$Improvement_{RMP1}$	32.4%	15.6%	35.4%	3.88%	22.6%
$Improvement_{RMP2}$	49.0%	20.3%	43.8%	6.2%	36.8%
ACC_{ADTree}	74.2%	67.7%	71.8%	86.9%	78.6%
$ACC_{Proposed}$	79.6%	73.5%	79.6%	88.1%	83.7%
$ACC_{Bagging}$	75.4%	69.8%	73.7%	86.9%	79.8%
$ACC_{Boosting}$	74.8%	68.5%	74.3%	87.1%	78.4%
ACC_{HBA}	80.8%	74.4%	80.9%	89.2%	83.9%
$ACC_{Combined}$	82.2%	74.7%	82.7%	89.5%	85.4%
$Improvement_{RMP1}$	20.9%	18.0%	27.7%	9.16%	23.8%
$Improvement_{RMP2}$	31.0%	21.7%	38.7%	19.8%	31.8%
ACC_{ANN}	73.6%	66.4%	72.2%	86.6%	77.9%
$ACC_{Proposed}$	77.6%	73.5%	78.7%	87.9%	83.6%
$ACC_{Bagging}$	74.7%	67.6%	73.6%	86.8%	78.7%
$ACC_{Boosting}$	73.6%	68.1%	74.3%	87.5%	77.9%
ACC_{HBA}	78.5%	73.2%	80.5%	88.9%	85.1%
$ACC_{Combined}$	79.8%	75.8%	81.8%	89.3%	85.8%
$Improvement_{RMP1}$	15.2%	21.1%	23.3%	9.70%	25.8%
$Improvement_{RMP2}$	23.5%	28.0%	34.5%	20.1%	35.7%

TABLE 6.7: Some experimental results when the 5-attribute artificial datasets were analyzed.

Datasets	Mass_5_7	Mass_5_15	Mass_5_38	Mass_5_54	Mass_5_66
$ACC_{DecisionTree}$	47.2%	51.6%	49.7%	46.8%	51.3%
$ACC_{Proposed}$	59.7%	63.1%	57.4%	57.8%	62.2%
$ACC_{Bagging}$	46.8%	53.7%	51.3%	50.1%	54.5%
$ACC_{Boosting}$	48.6%	52.1%	50.7%	49.8%	53.6%
ACC_{HBA}	51.4%	55.3%	53.8%	49.8%	56.6%
$ACC_{Combined}$	60.6%	64.4%	58.1%	58.5%	63.9%
$Improvement_{RMP1}$	23.7%	23.8%	15.3%	20.7%	22.4%
$Improvement_{RMP2}$	25.4%	26.4%	16.7%	22.0%	25.9%
ACC_{SVM}	47.8%	52.3%	48.2%	47.6%	50.9%
$ACC_{Proposed}$	58.6%	58.7%	59.1%	59.4%	61.8%
$ACC_{Bagging}$	46.1%	52.9%	49.3%	49.5%	53.7%
$ACC_{Boosting}$	47.8%	51.6%	51.4%	48.9%	53.5%
ACC_{HBA}	51.4%	57.3%	54.2%	52.1%	55.4%
$ACC_{Combined}$	59.8%	59.9%	60.3%	59.0%	62.9%
$Improvement_{RMP1}$	20.7%	13.4%	21.0%	22.5%	22.1%
$Improvement_{RMP2}$	23.0%	15.9%	23.4%	21.8%	24.4%
ACC_{ADTree}	48.1%	51.3%	49.2%	49.8%	51.6%
$ACC_{Proposed}$	58.7%	57.5%	57.2%	59.3%	61.8%
$ACC_{Bagging}$	46.8%	52.1%	48.6%	51.3%	54.3%
$ACC_{Boosting}$	47.2%	51.8%	50.9%	52.2%	55.1%
ACC_{HBA}	54.4%	55.3%	52.5%	57.6%	57.8%
$ACC_{Combined}$	59.6%	59.4%	59.6%	62.2%	62.1%
$Improvement_{RMP1}$	20.4%	12.7%	15.7%	18.9%	21.1%
$Improvement_{RMP2}$	22.2%	16.6%	20.5%	24.7%	21.7%
ACC_{ANN}	46.9%	50.7%	48.4%	48.5%	51.1%
$ACC_{Proposed}$	57.3%	58.8%	57.3%	60.2%	62.8%
$ACC_{Bagging}$	46.9%	50.7%	49.1%	53.4%	55.4%
$ACC_{Boosting}$	47.4%	52.2%	50.4%	54.9%	54.6%
ACC_{HBA}	51.4%	54.8%	53.8%	58.7%	57.8%
$ACC_{Combined}$	58.5%	59.1%	58.2%	62.1%	63.3%
$Improvement_{RMP1}$	19.6%	16.4%	17.2%	22.7%	23.9%
$Improvement_{RMP2}$	21.8%	17.0%	19.0%	26.4%	24.9%

TABLE 6.8: Some experimental results when the 6-attribute and 7-attribute artificial datasets were analyzed.

Datasets	Mass_6_14	Mass_6_36	Mass_6_41	Mass_7_28	Mass_7_33	Mass_7_53
$ACC_{DecisionTree}$	47.2%	52.4%	53.5%	48.6%	54.3%	55.2%
$ACC_{Proposed}$	56.5%	60.8%	59.8%	57.6%	63.3%	64.3%
$ACC_{Bagging}$	48.4%	54.7%	53.5%	52.2%	55.4%	54.9%
$ACC_{Boosting}$	47.0%	52.5%	54.3%	51.6%	54.9%	56.7%
ACC_{HBA}	50.4%	54.6%	57.6%	55.3%	59.8%	58.9%
$ACC_{Combined}$	56.1%	61.9%	60.1%	59.8%	65.2%	65.2%
$Improvement_{RMP1}$	19.7%	21.5%	13.5%	17.5%	19.7%	20.3%
$Improvement_{RMP2}$	21.8%	23.7%	14.2%	21.8%	23.9%	22.3%
ACC_{SVM}	47.8%	50.9%	52.8%	49.4%	53.8%	56.6%
$ACC_{Proposed}$	57.3%	61.8%	58.8%	60.4%	62.7%	64.3%
$ACC_{Bagging}$	48.4%	54.1%	51.5%	53.2%	55.4%	56.7%
$ACC_{Boosting}$	49.2%	52.6%	52.2%	54.4%	56.3%	58.1%
ACC_{HBA}	51.9%	54.7%	56.4%	58.7%	59.8%	62.2%
$ACC_{Combined}$	58.4%	62.1%	59.8%	62.2%	63.8%	65.0%
$Improvement_{RMP1}$	19.9%	21.6%	12.7%	21.7%	19.3%	17.7%
$Improvement_{RMP2}$	20.8%	22.7%	14.8%	25.3%	21.6%	19.4%
ACC_{ADTree}	48.8%	51.5%	54.1%	48.7%	54.9%	54.8%
$ACC_{Proposed}$	57.6%	60.3%	62.2%	58.5%	62.2%	63.7%
$ACC_{Bagging}$	50.6%	53.5%	54.3%	52.3%	56.5%	55.6%
$ACC_{Boosting}$	51.4%	54.3%	53.7%	53.1%	54.7%	57.5%
ACC_{HBA}	52.5%	55.8%	59.8%	57.8%	60.1%	59.8%
$ACC_{Combined}$	58.9%	61.1%	64.2%	60.5%	63.5%	65.2%
$Improvement_{RMP1}$	17.4%	19.1%	17.6%	19.1%	16.2%	19.7%
$Improvement_{RMP2}$	22.5.7%	20.5%	22.0%	23.0%	19.1%	23.0%
ACC_{ANN}	48.6%	51.1%	53.5%	49.4%	54.5%	53.9%
$ACC_{Proposed}$	58.2%	60.7%	60.3%	58.7%	64.1%	62.8%
$ACC_{Bagging}$	52.5%	54.4%	54.5%	52.1%	56.5%	54.5%
$ACC_{Boosting}$	53.7%	52.6%	55.6%	51.9%	54.6%	53.7%
ACC_{HBA}	51.4%	52.8%	59.8%	58.3%	58.9%	60.7%
$ACC_{Combined}$	61.7%	63.3%	62.3%	62.7%	64.2%	62.2%
$Improvement_{RMP1}$	21.5%	19.9%	14.6%	18.4%	21.0%	19.3%
$Improvement_{RMP2}$	24.4%	20.9%	18.9%	26.3%	21.3%	18.0%

In most of the scenarios, it is more difficult to improve an effective classification model (the one that can classify the target dataset with high accuracy) than one of poor quality models (the one that classifies the target dataset with low accuracy). The two relative maximum possible improvement rates ($Improvement_{RMP1}$ and $Improvement_{RMP2}$) provide better measures for expressing improvements. As one can observe from these tables, this is the reason why the proposed approach provides only little improvements when it analyzed the *Car* datasets. However, since the classification models used to classify the *Car* datasets are already accurate enough, little improvements in such cases can be very important. This is why it is very critical to observe that the values of $Improvement_{RMP1}$ in most of the cases are higher than 10%. When $Improvement_{RMP1}$ is considered, then these values are often times greater than 30%. This is another way, and perhaps more representative, to express the improvements achieved by the proposed approach. As it can be seen from the computational results, the proposed approach can be beneficial to a wide range of datasets of various degrees of classification difficulty.

Furthermore, the performance of the proposed approach dominates that of the Bagging and Boosting approaches. It is noticed that in these tests the performances of the Bagging and Boosting approaches were often unpredictable. That is, sometimes, these methods produced better results when compared to the stand-alone classifiers, while other times they produced inferior results. On the other hand, the proposed approach always outperformed the stand-alone classifiers as well as the Bagging and Boosting approaches.

When comparing the proposed approach to HBA, one may observe that the proposed approach is more effective than the HBA when analyzing difficult datasets (see also Tables 6.7 and 6.8). Even though in some cases the HBA is more effective than the proposed approach, the HBA approach takes longer processing time than the proposed one. Moreover, as it can be observed from Tables 6.1 to 6.8, the proposed approach can classify Type I and Type II testing vectors more accurately than the HBA. This is the motivation for combining the proposed approach with the HBA one. The combined approach classifies all Type I and Type II vectors by using the proposed monotonicity-based approach, while only the rest of the vectors (i.e., those of Type III) are classified by using the HBA approach. The corresponding classification results are given by the $ACC_{Combined}$ values. As one can observe from these tables, the values of $ACC_{Combined}$ is always higher than the values of ACC_{HBA} and any of the other methods tested and compared.

More importantly, when analyzed the artificial datasets (i.e., the datasets generated with class values randomly assigned to their vectors), the proposed approach outperformed any other tested approach in a profound way (see also Tables 6.7 and 6.8). When the stand-alone classifiers were used in these datasets, in many cases the classification accuracies were below 50%. Therefore, such classifications are of limited practical benefit. Even when the Bagging and Boosting approaches were used on these datasets, the classification accuracies were still too low as they cannot be more than 50% consistently. However, as one can see from the previous tables, by implementing the proposed approach, the classification accuracies were always greater than 50%, which makes the classifications meaningful. Furthermore, even when analyzing these very difficult datasets, the proposed approach can still obtain around 10% improvement in classification accuracy frequently. In terms of the previous relative measure for expressing classification improvements, the achieved improvements were often times higher than 15% for $Improvement_{RMP1}$ and 20% for $Improvement_{RMP2}$.

6.3 Analysis of the experimental results

Based on the above discussions, the proposed monotonicity-based approach is very effective in obtaining classification improvements. However, it is observed that the performance of this approach may vary. That is, for some experimental datasets, the implementation of this approach can lead to significant improvements, while for some other experimental datasets, only small improvements can be obtained.

This section will explore this phenomenon by performing an analysis on the previous experimental results. It explores the factors that may impact the performance of the proposed approach, and more importantly, how such factors impact the performance.

Tables 6.9 to 6.12 present the analysis of some of the datasets listed in Tables 6.1 to 6.8, while the classifiers *Decision Tree*, *Artificial Neural Network*, *ADTree* and *Support Vector Machine* are used as the base learner, respectively. Moreover,the following factors are what is of interest:

1. The average classification accuracy when the *Original_Classifier* is used to classify the entire testing dataset in 10-cross validations. This is denoted as F_1.

2. The average classification accuracy when the *Derived_Classifiers* are also used to facilitate the classifications on the testing dataset in 10-cross validations. This is denoted as F_2.

3. The average percentage of the testing vectors which are Type I vectors in the 10-cross validation. This is denoted as F_3.

4. The average classification accuracy when Type I vectors are classified by the *Original_Classifier* in 10-cross validations. This is denoted as F_4.

5. The average classification accuracy when Type I vectors are classified by the *Derived_Classifiers* in 10-cross validations. This is denoted as F_5.

6. The average percentage of the testing vectors which are Type II vectors in the 10-cross validation. This is denoted as F_6.

7. The average classification accuracy when Type II vectors are classified by the *Original_Classifier* in 10-cross validations. This is denoted as F_7.

8. The average classification accuracy when Type II vectors are classified by the *Derived_Classifiers* in 10-cross validations. This is denoted as F_8.

As one can observe from Table 6.9, for all these experimental datasets, their derived F_5 and F_8 values are much higher than that of F_4 and F_7. This observation supports the argument that the *Derived_Classifiers* can classify a portion of the testing vectors much more accurately than

TABLE 6.9: Some details of experiments when using *Decision Tree* (J48) as the base classifier.

Datasets	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Aba_7_11	77.6%	83.3%	28.6%	75.4%	98.8%	14.7%	72.4%	84.6%
Aba_58_9	64.6%	77.0%	36.7%	63.8%	97.4%	11.4%	69.4%	92.3%
Blood_2	64.6%	83.1%	47.4%	65.3%	98.2%	28.1%	67.3%	88.9%
Blood_42	68.2%	81.3%	27.6%	71.7%	99.3%	19.1%	66.9%	91.4%
Yea_21	73.6%	79.6%	22.8%	77.4%	97.7%	12.6%	75.7%	85.3%
Yea_49	83.6%	85.3%	11.6%	85.4%	98.5%	12.7%	86.6%	90.6%
Car_21	89.2%	91.0%	29.3%	89.5%	100%	17.6%	92.7%	98.2%
Car_48	84.3%	87.3%	27.4%	86.1%	94.7%	14.9%	84.4%	92.5%
Auto_34	73.5%	77.9%	36.2%	74.8%	88.6%	13.4%	72.9%	78.7%
Auto_135	68.5%	77.3%	38.7%	69.3%	90.7%	21.4%	67.6%	81.4%
Mammo_4	74.5%	81.2%	25.9%	74.8%	92.3%	16.7%	76.7%	89.7%
Mammo_69	72.6%	79.6%	22.5%	71.5%	94.4%	20.3%	73.5%	87.4%
Mess_5_15	51.6%	63.1%	23.5%	50.8%	71.1%	14.5%	51.7%	68.4%
Mess_5_38	49.7%	57.4%	31.2%	53.7%	76.7%	31.3%	49.3%	70.1%
Mess_6_14	46.8%	57.8%	28.9%	49.8%	75.4%	19.8%	44.1%	68.7%
Mess_6_36	51.3%	62.2%	25.3%	51.7%	83.4%	15.4%	50.8%	74.3%
Mess_7_28	48.6%	57.6%	20.1%	48.7%	78.5%	17.8%	47.9%	72.3%
Mess_7_33	54.3%	63.3%	19.8%	53.4%	79.5%	19.6%	52.6%	69.8%

TABLE 6.10: Some details of experiments when using *Artificial Neural Network* as the base classifier.

Datasets	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Aba_7_11	77.6%	83.8%	28.6%	75.5%	98.6%	14.7%	72.3%	84.7%
Aba_58_9	64.6%	77.7%	36.8%	63.1%	97.5%	11.4%	68.5%	92.5%
Blood_2	64.6%	83.6%	46.5%	65.5%	98.2%	28.1%	67.1%	88.1%
Blood_42	68.2%	81.6%	27.4%	70.6%	99.6%	19.1%	66.2%	91.8%
Yea_21	73.8%	79.1%	22.4%	75.2%	97.4%	12.6%	75.7%	86.2%
Yea_49	83.1%	85.8%	11.7%	84.9%	98.7%	12.7%	86.1%	91.4%
Car_21	89.2%	90.5%	29.3%	88.2%	100%	17.6%	91.9%	97.9%
Car_48	84.3%	87.5.3%	27.4%	85.9%	94.7%	14.9%	83.4%	91.9%
Auto_34	73.5%	76.9%	36.2%	73.8%	88.6%	13.4%	72.1%	78.1%
Auto_135	68.5%	77.4%	38.7%	69.6%	90.2%	21.4%	67.0%	81.8%
Mammo_4	74.5%	81.8%	25.9%	74.5%	92.6%	16.7%	76.8%	89.4%
Mammo_69	72.6%	79.4%	22.5%	71.7%	93.9%	20.3%	73.1%	87.5%
Mess_5_15	51.6%	62.8%	23.5%	52.1%	71.2%	14.5%	51.2%	69.2%
Mess_5_38	49.7%	57.9%	31.2%	52.8%	76.7%	31.3%	50.5%	71.4%
Mess_6_14	46.8%	57.1%	28.9%	50.6%	75.4%	19.8%	45.1%	68.9%
Mess_6_36	51.3%	62.8%	25.3%	51.9%	83.8%	15.4%	50.4%	74.5%
Mess_7_28	48.6%	58.1%	20.1%	49.1%	77.9%	17.8%	47.4%	72.4%
Mess_7_33	54.3%	63.8%	19.8%	53.8%	80.5%	19.6%	52.2%	69.1%

TABLE 6.11: Some details of experiments when using *ADTree* as the base classifier.

Datasets	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Aba_7_11	77.6%	82.3%	28.6%	75.4%	98.1%	14.7%	72.4%	84.7%
Aba_58_9	64.6%	77.5%	36.7%	63.8%	97.3%	11.4%	69.4%	92.8%
Blood_2	64.6%	83.7%	47.4%	65.3%	98.8%	28.1%	67.3%	88.2%
Blood_42	68.2%	81.7%	27.6%	71.7%	98.9%	19.1%	66.9%	90.4%
Yea_21	73.6%	79.1%	22.8%	77.4%	97.2%	12.6%	75.7%	85.8%
Yea_49	83.6%	85.9%	11.6%	85.4%	98.8%	12.7%	86.6%	90.8%
Car_21	89.2%	90.2%	29.3%	89.5%	100%	17.6%	92.7%	98.2%
Car_48	84.3%	86.9%	27.4%	86.1%	94.8%	14.9%	84.4%	92.8%
Auto_34	73.5%	77.2%	36.2%	74.8%	88.6%	13.4%	72.9%	79.2%
Auto_135	68.5%	77.8%	38.7%	69.3%	90.7%	21.4%	67.6%	81.8%
Mammo_4	74.5%	81.8%	25.9%	74.8%	92.8%	16.7%	76.7%	89.4%
Mammo_69	72.6%	78.5%	22.5%	71.5%	93.9%	20.3%	73.5%	88.4%
Mess_5_15	51.6%	63.5%	23.5%	50.8%	71.5%	14.5%	51.7%	68.5%
Mess_5_38	49.7%	57.7%	31.2%	53.7%	76.2%	31.3%	49.3%	71.2%
Mess_6_14	46.8%	56.6%	28.9%	49.8%	75.7%	19.8%	44.1%	69.5%
Mess_6_36	51.3%	62.5%	25.3%	51.7%	83.4%	15.4%	50.8%	74.8%
Mess_7_28	48.6%	57.7%	20.1%	48.7%	77.9%	17.8%	47.9%	72.8%
Mess_7_33	54.3%	63.9%	19.8%	53.4%	79.8%	19.6%	52.6%	69.2%

TABLE 6.12: Some details of experiments when using *Support Vector Machine* as the base classifier.

Datasets	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Aba_7_11	77.6%	82.9%	28.6%	75.4%	98.3%	14.7%	72.4%	85.1%
Aba_58_9	64.6%	75.8%	36.7%	63.8%	96.5%	11.4%	69.4%	92.6%
Blood_2	64.6%	81.8%	47.4%	65.3%	97.6%	28.1%	67.3%	88.1%
Blood_42	68.2%	81.4%	27.6%	71.7%	98.5%	19.1%	66.9%	90.6%
Yea_21	73.6%	80.4%	22.8%	77.4%	97.6%	12.6%	75.7%	85.9%
Yea_49	83.6%	84.6%	11.6%	85.4%	98.5%	12.7%	86.6%	90.1%
Car_21	89.2%	90.8%	29.3%	89.5%	100%	17.6%	92.7%	98.9%
Car_48	84.3%	87.6%	27.4%	86.1%	94.7%	14.9%	84.4%	91.5%
Auto_34	73.5%	78.5%	36.2%	74.8%	89.2%	13.4%	72.9%	77.1%
Auto_135	68.5%	78.1%	38.7%	69.3%	90.0%	21.4%	67.6%	82.5%
Mammo_4	74.5%	81.7%	25.9%	74.8%	91.5%	16.7%	76.7%	88.6%
Mammo_69	72.6%	79.2%	22.5%	71.5%	94.9%	20.3%	73.5%	87.9%
Mess_5_15	51.6%	63.6%	23.5%	50.8%	72.2%	14.5%	51.7%	68.9%
Mess_5_38	49.7%	56.9%	31.2%	53.7%	76.1%	31.3%	49.3%	71.3%
Mess_6_14	46.8%	57.3%	28.9%	49.8%	75.5%	19.8%	44.1%	69.2%
Mess_6_36	51.3%	61.8%	25.3%	51.7%	83.9%	15.4%	50.8%	75.4%
Mess_7_28	48.6%	57.4%	20.1%	48.7%	77.8%	17.8%	47.9%	72.9%
Mess_7_33	54.3%	63.6%	19.8%	53.4%	78.6%	19.6%	52.6%	68.9%

TABLE 6.13: The details of the linear regression models for the experimental results listed in Table 6.9.

Experimental Datasets	Coefficient of F_3	Coefficient of F_4	Coefficient of F_6	Coefficient of F_7	R-Square Value
Abalone	0.398	-1.754	2.214	-0.856	0.826
Blood	0.289	-0.854	1.135	-0.391	0.904
Yeast	2.313	-1.342	1.432	-2.682	0.823
Car	1.482	-1.439	0.894	-1.432	0.844
Auto_MPG	0.867	-2.134	1.295	-0.851	0.839
Mess_5	2.185	-0.943	2.439	-0.493	0.865
Mess_6	4.328	-3.593	2.118	-3.688	0.792
Mess_7	0.587	-1.175	0.556	-2.214	0.819

the *Original_Classifier*. This is exactly the reason why the proposed approach can consistently generate more accurate classifications.

For example, the dataset *Aba_58_9* has 36.7% of its testing vectors as Type I vectors, and 11.4% as Type II vectors. Furthermore, when classifying its Type I vectors by using the *Original_Classifier*, only 63.8% of them can be correctly classified. However, the *Derived_Classifiers* can accurately classify 97.4% of them. Meanwhile, by using the proposed approach, about 92.3% $-$ 69.4% = 22.9% of the Type II testing vectors can be more accurately classified by the *Derived_Classifiers*. Therefore, one can observe about 77.0%-64.6% = 12.4% overall classification improvement by implementing the proposed approach when analyzing the dataset *Aba_58_9*. In terms of the measure that expresses the relative improvement, the value is equal to 35%.

In order to better understand how such factors impact the effectiveness of the proposed approach, several linear regression models were created using the classification improvements (the difference between the factors F_1 and F_2) as the dependent variable, while the factors F_3, F_4, F_6,and F_7 were used as the independent variables. The R-Square values of the models and the coefficients of the independent variables is what is important. Furthermore, a statistical package by SAS was used to generate such linear regression models. Tables 6.13 to 6.16 list the regression models derived from analyzing the experimental results shown in Tables 6.9 to 6.12, respectively.

TABLE 6.14: The details of the linear regression models for the experimental results listed in Table 6.10.

Experimental Datasets	Coefficient of F_3	Coefficient of F_4	Coefficient of F_6	Coefficient of F_7	R-Square Value
Abalone	0.385	-1.635	1.562	-0.256	0.808
Blood	0.549	-1.024	1.822	-0.892	0.859
Yeast	2.951	-1.852	2.300	-2.100	0.910
Car	2.252	-1.153	1.210	-1.325	0.855
Auto_MPG	1.027	-2.634	1.821	-0.632	0.845
Mess_5	2.525	-0.258	2.025	-1.025	0.896
Mess_6	4.188	-3.521	2.952	-2.845	0.801
Mess_7	0.627	-4.263	1.310	-2.962	0.789

TABLE 6.15: The details of the linear regression models for the experimental results listed in Table 6.11.

Experimental Datasets	Coefficient of F_3	Coefficient of F_4	Coefficient of F_6	Coefficient of F_7	R-Square Value
Abalone	0.562	-1.582	2.695	-0.520	0.833
Blood	0.629	-0.364	1.522	-0.361	0.895
Yeast	1.553	-1.698	1.251	-1.252	0.862
Car	1.852	-1.469	1.105	-1.424	0.852
Auto_MPG	0.967	-2.524	1.365	-0.885	0.841
Mess_5	1.265	-0.625	2.025	-0.639	0.806
Mess_6	3.652	-2.365	2.362	-2.526	0.841
Mess_7	1.125	-1.962	1.258	-2.852	0.823

TABLE 6.16: The details of the linear regression models for the experimental results listed in Table 6.12.

Experimental Datasets	Coefficient of F_3	Coefficient of F_4	Coefficient of F_6	Coefficient of F_7	R-Square Value
Abalone	0.125	-1.785	2.365	-0.251	0.842
Blood	0.203	-0.254	1.125	-0.896	0.896
Yeast	2.962	-1.361	1.185	-2.155	0.864
Car	1.222	-1.98	0.362	-1.985	0.825
Auto_MPG	0.256	-2.234	1.510	-0.120	0.817
Mess_5	2.362	-0.693	2.124	-1.012	0.836
Mess_6	4.126	-3.123	2.352	-3.228	0.825
Mess_7	0.587	-1.645	0.450	-2.863	0.842

As one can observe from these tables, the R-Square values in most of the models are more than 0.80, which indicates a satisfactory goodness of fit. Furthermore, the coefficients of the independent factors are consistent across the tables, that is, the factors which have positive(negative) coefficients in one table also have positive(negative) coefficients in all other tables. This observation indicates that some factors have positive impact to the classification improvements, while some other factors impact the improvements in a negative way.

To be more specific, the factors which have positive coefficient are directly proportional to the classification improvements. The increase of their values can help to obtain more accurate classifications. These are the factors F_3 and F_6. In the opposite case, if the factors have negative coefficients in the models, then they are inversely proportional to the improvements. An increase of their values will result in less significant improvements when implementing the proposed approach. These are the factors F_4 and F_7.

In other words, when implementing the proposed approach to analyze a dataset, the more of its testing vectors happen to be Type I or Type II vectors, the higher improvements can be expected. This is supported by the fact that the Type I and Type II vectors can be much more accurately classified by the $Derived\_Classifiers$. Meanwhile, when analyzing easy datasets, that is, datasets which can be accurately classified by the $Original\_Classifier$, such as the ones with high F_4 and F_7 values, little improvements can be obtained. This happens because when the $Original\_Classifiers$ are effective enough, then there is no significant difference between the performance of the $Original\_Classifier$ and the $Derived\_Classifiers$. Similar conclusions are reached when the two relative measures for expressing accuracy are used.

Chapter 7

Conclusions

7.1 An Overview of this Research

Our research focused on the problems of how to evaluate the classification difficulty, or learnability, of numeric datasets by exploring some monotonicity-based characteristics of the data. More importantly, how to use such characteristics to improve the classification accuracy. In the experiments described in this dissertation, the difficulty of a dataset is indicated by the average accuracy when it is classified by a wide set of classifiers. It proposes that such difficulty can be accurately predicted by analyzing some monotonic characteristics of the dataset, and different types of numeric datasets may show similar relationships between their monotonic characteristics and their difficulty in learning. Furthermore, by further exploring the monotonic properties of the datasets, it shows a meta-learning approach to improve the classification accuracies on all numeric datasets.

The definition of monotonicity is first discussed in Section 2 and then further explored by considering the attributes with positive/negative effects. Two vectors are said to be monotonically related if and only if they are defined on the same positive/negative attributes and one precedes another. Furthermore, by considering their class values, a pair of related vectors may comprise an AMP1 pair, an AMP2 pair, or a CMP pair. After one considers all possible pairs of vectors, some special vectors, called border points or extended border points, can be determined accordingly. By analyzing such monotonically related vectors and the border points, the key monotonic features of datasets can be determined.

Several groups of experiments were designed and performed to explore the relationships between the difficulty of numeric datasets and their monotonic characteristics. The details are provided in Section 4. According to the experimental results, regression models generated by determining the monotonic characteristics of the numeric datasets can be used to evaluate their difficulties very accurately.

One of the main contributions of our study is to propose and demonstrate that the monotonic features of numeric datasets play an important role in determining their difficulty. Furthermore, by analyzing these monotonic features, one can generate regression models that can accurately predict such difficulty for numeric datasets provided that the datasets have enough number (i.e., more than 4%) of monotonically related vectors. In the experiments of analyzing continuous datasets, even though continuous datasets may differ in many ways, they still present similar key relationships between their monotonic features and their classification difficulties.

Based on the above observation, we proposed an approach for dividing a difficult to classify training dataset into a group of easily to classify subsets. As result, classifiers derived from the smaller datasets tend to be much more accurate than the classifier derived when the original dataset is used as a whole. When the classifiers from the smaller datasets are combined together, the combined classification system performs consistently better than the original classifier derived from the entire dataset.

The proposed approach has been compared against the stand-alone approach for a number of well-known classifiers. It has also been compared to Boosting and Bagging approaches as well as the HBA approach [35]. In all occasions, the new approach outperformed all the previous approaches on a wide range of tests. For the HBA case, a combined approach is proposed as well which is profoundly more powerful than any other approach. This is a meta-heuristic approach, as it can be used in conjunction with any known classifier, and offers an exciting potential to significantly improve classification results in many cases.

7.2 Significance of the Findings of this Research

Before this research if one wanted to determine how difficult or easy a dataset is to classification analysis (i.e., what is called here its learnability value), he/she would had to extensively analyze the dataset in terms of many classifiers and collect statistical information. Such an approach is time consuming and also might be biased as one does not know which classifiers to use for this examination. But even more important, one would have no clue why a given dataset is difficult or easy for classification analysis. Thus, one would have to repeat this tedious approach with any new dataset.

The main conclusion of this research, however, provides a satisfactory answer to this challenge. It demonstrates that the monotonicity properties of numeric datasets are the main factors which impact their classification difficulty. Moreover, a method is introduced which uses some key monotonic properties to build regression models, and data from the experimental results. The derived regression models are quite powerful in predicting the classification difficulty quite accurately for many datasets.

Comparing to the traditional approach, the proposed method takes much shorter time without significantly compromising on accuracy. Furthermore, it only uses the monotonic relationships between pairs of vectors in the target dataset, and it is not concerned on what classifier will be used. That is, the only required data are dataset specific.

The above discovery is great contribution. However, there is another major contribution as well. It uses these results to improve classification accuracy when a classifier is used. This was demonstrated in Section 6. Which provides a general method for improving classification accuracy in various application domains.

First of all, the proposed approach is a meta-learning approach, and thus any classification algorithm can be used as the base classifier. Next, it works with any numeric dataset regardless the application domain, measurement units and so on. Last but not least, this approach can improve the classifications on any dataset provided that all attributes can somehow be converted into ordinal ones.

The classification improvement gained by implementing the proposed approach is significant and stable, according to the experimental results. The relative maximum possible improvement was frequently above 25%, regardless the difficulty of the experimental dataset used for the test.

Moreover, there exist some very difficult datasets that most of the classifiers cannot classify them with more than 50% accuracy. In this particular scenario traditional methods are doing meaningless classifications since they perform no better than the most naive random guessing. However, by analyzing the monotonic properties of difficult datasets, the proposed approach can always classify them with more than 50% accuracy, which makes classification more meaningful.

In summary, this research addressed successfully the following important challenges that had defied explanation until now: a) What factors make a dataset easy or difficult? b) How these factors impact the data difficulty? c) How to evaluate the data difficulty by analyzing these factors? and d) How to use such factors to improve classification? The answers to these challenges not only solve an existing mystery in data analysis, but also forge a solid foundation for future studies on monotonicity, such as monotonicity in transfer learning and in cases under different misclassification costs for the various types of error that may occur during the classification process. Considering the bright prospects of using monotonicity in data analysis, the end of this research is just a new beginning.

References

[1] J. Abonyi, J. Roubos, and F. Szeifert. Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. *International Journal of Approximate Reasoning*, 32(1):1–21, 2003.

[2] G. Baudat and F. Anouar. Kernel-based methods and function approximation. *Proceedings of 2001 International Joint Conference on Neural Networks (IJCNN '01)*, 2:1244–1249, 2001.

[3] E. Boros, P. Hammer, and T. Ibaraki. Polynomial-time recognition of 2-monotonic positive boolean functions given by an oracle. *SIAM Journal of Computing*, 1997.

[4] L. Breiman. Random forests. *Machine Learning*, 45(4):5–32, 2001.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2), 1996.

[6] H. Buhrman and R. D. Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(9):2002–2034, 1999.

[7] J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[8] M. Camara, F. Gustin, and H. Gualous. Supercapacitors and battery power management for hybrid vehicle applications using multi boost and full bridge converters. *Power Electronics and Applications, 2007 European Conference*, 2(5):71–98, 2007.

[9] O. Chapelle. Training a support vector machine in the primal. *Neural Computation.*, 19(5):1115–1178, 2007.

[10] Y. Chung and S. Moon. Memory allocation with lazy fits. *ISMM '00 Proceedings of the 2nd international symposium on Memory management*, 36(1):189–227, 2000.

[11] R. Church. Numerical analysis of certain free distributive strctures. *Duke Math*, 6:732–734, 1940.

[12] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(3):393–405, 1990.

[13] S. Dash and R. Rengaswamy. Fuzzy-logicbased trend classification for fault diagnosis of chemical processes. *Computers and Chemical Engineering*, 27(3):347–362, 2003.

[14] R. Dedekind. Ueber zerlegungen von zahlen durch ihre grossten gemeinsamen teiler. *Festchrift Hoch. Braunschweigu.ges. Werke*, 2:103–148, 1897.

[15] G. Demirz and T. Altay. Classification by voting feature intervals. *Machine Learning: ECML-97*, 1224(1997):85–92, 1997.

[16] H. Drucker and R. Schapire. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(04), 1993.

[17] W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for rule-based data understanding. *Proceedings of the IEEE*, 92(5):771–805, 2004.

[18] B. Efron. The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Association*, 70(6):98–124, 1975.

[19] M. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, 1996.

[20] Y. Freund. The alternating decision tree learning algorithm. *In ICML '99: Proceedings of the 16th International Conference on Machine Learning*, 13(2):124–133, 1999.

[21] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *In Proceedings of EuroCOLT¡¯94: European Conference on Computational Learning Theory. LNCS*, 6(9):171–185, 1994.

[22] D. Gainanov. On one criterion of the optimality of an algorithm for evaluating monotonic boolean functions. *USSR Computational Mathematics and Mathematical Physics*, 24(4):176–181, 1985.

[23] G. Hansel. Sur le nombre des fonctions boolenes monotones den variables. *C. R. Acad. Sci. Pair.*, 262(20):1088–1090, 1966.

[24] T. Horv¡äath, G. Pass, F. Reichartz, and S. Wrobel. A logic-based approach to relation extraction from texts. *19th International Conference on Inductive Logic Programming*, 18(6):34–48, 2009.

[25] G. Huang, P. Saratchandran, and N. Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE Trans. on Neural Networks*, 16(1):57–67, 2005.

[26] A. Jarre and B. Paterson. Knowledge-based systems as decision support tools in an ecosystem approach to fisheries: Comparing a fuzzy-logic and a rule-based approach. *Progress In Oceanography*, 79(2):390–400, 2008.

[27] B. Kernighan and X. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49(2):291–308, 1970.

[28] B. Kovalerchuk, E. Triantaphyllou, A. S. Deshpande, and E. Vityaev. Monotone boolean function learning techniques integrated with user interaction. *Information Sciences*, 151, 1996.

[29] C. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. *Machine Learning*, 4(15):1398–1414, 1998.

[30] X. Liao, C. Li, and K. Wong. Criteria for exponential stability of cohen-grossberg neural networks. *Neural Networks*, 17(10):1401–1414, 2004.

[31] C. J. Lin. http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html. 2003.

[32] T. Makino and L. Ibaraki. The maximum latency and identification of positive boolean functions. *SIAM Journal on Computing.*, 26:1363–1383, 1997.

[33] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D.M. Hummels. On the training of radial basis function classifiers. *Neural Networks*, 5(4):595–603, 1992.

[34] R. A. Olshen. Classification and regression trees. *Wadsworth and Brooks Cole Advanced Books and Software, Pacific California*, 11(6):97–106, 1984.

[35] H.N.A Pham and E. Triantaphyllou. An application of a new meta-heuristic for optimizing the classification accuracy when analyzing some medical datasets. *Expert Systems with Applications*, 36(5), 2009.

[36] M. Saks and A. Wigderson. Probabilisitic boolean decision trees and the complexity of evluting game trees. *Proceedings of the 27th FOCS*, 7(3):29–38, 1986.

[37] Documentation of regression models, 2008. http://support.sas.com/documentation/.

[38] N. Sokolov. On the optimal evaluation of monotonic boolean functions. *Ussr Computational Mathematics and Mathematical Physics.*, 22(2):207–220, 1982.

[39] J. A. K. Suykens. Least squares support vector machine classifiers. *Neural Processing Letters.*, 9(3):293–300, 1999.

[40] T. Tersvirta, C. Lin, and J. Granger. Power of the neural network linearity test. *Journal of Time Series Analysis*, 14(2):209–220, 1996.

[41] V. Torvik and E. Triantaphyllou. Minimizing the average query complexity of learning monotone boolean functions. *INFORMS Journal of Computing*, 14(2):144–174, 2002.

[42] V. Torvik and E. Triantaphyllou. Guided inference of nested monotone boolean functions. *Information Sciences*, (151):171–200, 2003.

[43] E. Triantaphyllou. *Data Mining and Knowledge Discovery via Logic-Based Methods: Theory, Algorithms and Applications*. Springer, NewYork, NY, USA, 2010.

[44] 2012. http://archive.ics.uci.edu/ml/datasets.html.

[45] M. Ward. Note on the order of free distributive lattices. *Bull America Mathematics Society*, 52:423, 1946.

[46] Data mining with open source machine learning software, 2004. http://www.cs.waikato.ac.nz/ml/weka/.

[47] S. Xu and J. Lam. A new approach to exponential stability analysis of neural networks with time-varying delays. *Neural Networks*, 19(1):76–83, 2006.

Vita

Di Lin, a native of Fuzhou, Fujian, China, received his bachelor degree at Fuzhou University in 2003. Thereafter, he worked for APEX software company for four years in Fuzhou, Fujian, China. As his interest in computer science grew, he made the decision to enter graduate school in the school of electrical engineering and computer science, computer science and engineering division at Louisiana State University. After four and half years, he got his master degree in System Science in December 2011. Then he decided to start his career in industry while part time continue his PHD. He will receive his PH.D degree in Computer Science in August 2013.

During his part-time study period, Di Lin worked for Risk Management Solution as a software engineer for 13 months. Then he joined Acxiom as a senior software engineer, his work is closely related to his research, that is, deal with big data and focus on data analysis.