ENSEMBLE-CLASSIFIER APPROACH TO NOISE ELIMINATION: A CASE STUDY IN SOFTWARE QUALITY CLASSIFICATION

VEDANG H. JOSHI

ENSEMBLE-CLASSIFIER APPROACH TO NOISE ELIMINATION: A CASE STUDY IN SOFTWARE QUALITY CLASSIFICATION

by

Vedang H. Joshi

A Thesis Submitted to the Faculty of

The College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, Florida

August 2004

ENSEMBLE-CLASSIFIER APPROACH TO NOISE ELIMINATION: A CASE STUDY IN SOFTWARE QUALITY CLASSIFICATION

by

Vedang H. Joshi

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Taghi M. Khoshgoftaar, Department of Computer Science and Engineering, and has been approved by the members of his supervisory committee. It was submitted to the faculty of The College of Engineering and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

SUPERVISORY COMMITTEE:

Taghi M. Khoshgoftaar
Thesis Advisor

Borko Furht

Chairperson, Department of Computer Science and Engineering

Dean, College of Engineering

June 22, 2004
Vice President for Research and Graduate Studies Date

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Taghi M. Khoshgoftaar, for the valuable comments and the kind support he has offered in completing my thesis. Dr. Khoshgoftaar, a great researcher in his own right, has inspired me in more than many ways with some of his greatest qualities as a researcher. It has been a real privilege working under his guidance. Special thanks are due to Dr. Martin K. Solomon and Dr. Bassem AlHalabi for agreeing to serve on my thesis committee and reviewing my thesis.

I cannot sufficiently thank all my colleagues who made a wonderfully well-coordinated effort to carry out the experimental work presented in this thesis, in the most efficient manner. I would also like to thank Naeem Seliya and Kehan Gao for sharing their knowledge of LaTeX with me and meticulously reviewing my thesis.

Many thanks to current and former members of the Empirical Software Engineering Laboratory, in particular Naeem Seliya, Kehan Gao, Jayanth Rajeevalochanam, Laurent Nguyen, Angela Herzberg, Yi Liu, and my fellow graduate students - Pierre Rebours, Yunling Wang, Hua Lin, and Nawal Abraham - for being such a great team to work with.

ABSTRACT

Author:	Vedang H. Joshi
Title:	Ensemble-Classifier Approach to Noise Elimination: A Case Study in Software Quality Classification
Institution:	Florida Atlantic University
Thesis Advisor:	Dr. Taghi M. Khoshgoftaar
Degree:	Master of Science
Year:	2004

This thesis presents a noise handling technique that attempts to improve the quality of training data for classification purposes by eliminating instances that are likely to be noise. Our approach uses twenty five different classification techniques to create an ensemble of classifiers that acts as a noise filter on real-world software measurement datasets. Using a relatively large number of base-level classifiers for the ensemble-classifier filter facilitates in achieving the desired level of noise removal conservativeness with several possible levels of filtering. It also provides a higher degree of confidence in the noise elimination procedure as the results are less likely to get influenced by (possible) inappropriate learning bias of a few algorithms with twenty five base-level classifiers than with a relatively smaller number of base-level classifiers. Empirical case studies of two different high assurance software projects demonstrate the effectiveness of our noise elimination approach by the significant improvement achieved in classification accuracies at various levels of filtering.

To my loving parents

CONTENTS

TABLES

FIGURES

Chapter 1

INTRODUCTION

In today's IT-driven and highly competitive world, information holds the key to success for any organization. However, just having vast amount of data/information may not necessarily work to an organization's advantage if the quality of the data and the usefulness of the data, in turn, is questionable. Not only can data of poor quality, if not handled correctly, hinder an organization's success, but it could also have disastrous consequences. Therefore, it is imperative for an organization to ensure that the data is of good quality before performing any task that involves extensive data analysis and/or decision making based on data mining results using the data available.

Indeed, the quality of data is important from a data mining point of view. Inductive learning algorithms, the heart of data mining, aim to generalize the concepts learnt from a set of training instances so as to improve the classification accuracy on previously unseen observations (instances). The predictive accuracy of a classification technique is influenced by two (among others) major factors: (1) Quality of the training data, and (2) Appropriateness of the chosen algorithm for the given

data. Poor-quality (noisy) data, when used during training, can have undesirable consequences due to decision making based on incorrect results. Hence, using an appropriate noise handling procedure as a preamble to any data mining / KDD task is of paramount importance.

The problem of effectively dealing with data noise can be approached mainly in three different ways. To cope with noise, one can either use robust (noise-tolerant) algorithms, try to correct noisy instances, or filter out noisy instances from the dataset. In the first approach, a robust learning algorithm is employed in such a way that the classifier built will not be overfitted to the possibly noisy training instances. This simple but subtle approach of handling noise finds its roots in the Occam's razor principle applied to Inductive Learning [33]. Teng [100] has explored a different approach, called *polishing*, in which instead of removing the instances identified as being noisy, corrections are made to either one or more features (attributes) or the class label of the instances suspected of being noisy. The concept of *Polishing* takes advantage of the fact that different components in a dataset may not be totally independent except in the case of irrelevant attributes. The third approach, noise elimination, is a rather direct approach that attempts to improve the quality of input data for hypothesis formation by removing potentially noisy instances so that they do not influence the hypothesis constructed [31]. This is the approach we have adopted to handle potentially noisy datasets in the domain of Software Quality Classification.

The empirical study presented in this thesis investigates the use of a noise elimination procedure, based on ensemble-classifier approach, in the context of Software Quality Classification problem. A Software Quality Classification model can assist software quality improvement efforts by identifying software program modules that are likely to be *fault-prone (fp)* during operations. This facilitates cost-effective utilization of resources allocated for software testing, inspection, and quality enhancement. Software measurements are key in developing a Software Quality Estimation model because of the software engineering assumption that they hold the underlying information regarding software product quality.

Noise elimination with ensemble-classifier approach was deemed appropriate for our study. The basic assumption in our study is that if a large number of classifiers misclassify a given software module, then it is likely that it is a noisy instance in the dataset. More specifically, such a software module suggests that its software measurements and quality data do not adhere to (or represent) the underlying characteristics of the quality of the software product. Noisy instances in a poor-quality dataset may have either erroneous attribute values (attribute noise) or corrupted class labels (class noise). However, since machine learning algorithms usually treat noisy examples as being mislabeled [32], we feel that the noise identified by our approach could actually be either attribute noise or class noise.

Our study extends the ensemble-classifier approach, first introduced by Brodley and Friedl [9], by experimenting with relatively large number of base-classifiers

to explore different levels of filtering instead of just majority- and consensus-based filtering. In the ensemble-classifier approach presented in [9] for noise elimination, only three base-level classifiers were employed. A similar work with five different base-level classifiers was presented in [10]. In our opinion, the number of base-level classifiers can be a key factor when the ensemble-classifier approach is used in noise detection and elimination. In the context of [9], one can argue that it is quite possible that two out of the three different classifiers or even all the three classifiers could misclassify genuinely noise-free instance(s). Similar argument can be made for five base classifiers [10] for noise filtering.

Indeed, it may not be wise to form an opinion about an instance being noisy by considering only a small number of classifiers, because the appropriateness (or bias) of the chosen learning algorithms applied to a particular dataset also plays a significant role. It may well be that the few chosen classifiers don't have the appropriate bias to learn the concepts for the given domain.

Experimenting with a rather large number of classifiers can ensure that we are reducing the probability of throwing away good-quality data and raising the level of confidence in the identification of actual noisy instances. In our study, we used 25 different base-level classifiers from different computational categories, such as Bayesian, instance-based, rule-based, decision-tree based, pattern-based, and statistical techniques, etc., for our ensemble-classifier noise removal approach. Unlike Brodley and Friedl's approach [9] that only considers majority filtering (the least

conservative approach) and consensus filtering (the most conservative approach), our study examines the effects of different levels of noise filtering on the predictive accuracy of classifiers. By using 25 base-level classifiers, we were able to achieve various levels of filtering (levels of conservativeness) for noise removal from the software measurement data investigated.

To our knowledge, this work is one of the few studies that examines the effect of a noise handling technique on a real-world dataset with potential inherent noise. Many empirical investigations, such as [9, 100, 117], have evaluated different noise handling mechanisms on datasets in which noise is artificially injected, either in the class label or in the attribute values. In such cases, there is no way to ensure that the noise handling procedure improves the true classification accuracy. Whereas, with our approach, noise free evaluation dataset is available because of the way noise filtering is performed.

The effectiveness of our noise elimination approach is evident in the significant improvement achieved in classification accuracies at various levels of filtering for both the case-studies of high assurance software projects empirically investigated in our study. The results statistically confirmed our intuitive assumption that the classification performance would improve as more and more software modules likely to be noise are eliminated. This is evidenced by the significant performance difference between the datasets with different levels of noise filtering. This was also apparent as the NECM values decreased from the most conservative level to the

least conservative level of noise filtering.

A Z-test was performed to compare two different proportions – proportions of the modules identified as likely-noise by two different noise filtering approaches. First, we compared the proportion of the modules identified as noisy (and hence eliminated) by our approach (ensemble-classifier consensus filter with 25 base-level classifiers) to the proportion of the instances identified as noisy by ensemble-classifier consensus filter with only 5 base-level classifiers [10]: J48, IBk, SMO, JRIP, and LWLStump. The results revealed that ensemble-classifier consensus filter is, statistically speaking, much more conservative with twenty five base-level classifiers than with only five classifiers. Thus, experimenting with relatively large number of classifiers can provide us a flexibility to choose the amount of conservativeness desired for noise elimination.

This thesis begins with an introduction to software measurement and the metrics involved in this study. Chapter 3 describes the various methodologies involved. They include different noise handling procedures,, different classification techniques used as base-level classifiers, Expected Cost of Misclassification as a singular practical classification performance measure, Z-test for proportions, Two-way Analysis of Variance (ANOVA) model, and Multiple Pairwise Comparison. Chapter 4 describes the experiments conducted and the results obtained. Finally, Chapter 5 draws conclusions from this study, and indicates directions for future study.

Chapter 2

SOFTWARE METRICS

At the heart of every engineering activity, there is measurement. In fact, measurements pervade almost every aspect of our lives to such a great extent that it is literally impossible to make a meaningful progress without measurements. In this chapter, we present how measurements play a significant role in our life, with a particular reference to software engineering activities.

2.1 Measurements

Indeed, measurements are so widely used in our daily activities that they have become commonplace. From professional technologists to normal human beings, everyone uses measurements to gain better understanding of the environment, interact with the surroundings, and improve life by taking important decisions in an objective and scientific manner.

Fenton and Pfleeger [23] define measurement as a process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.

The corollary to DeMarco's rule in [23]: *You cannot control what you cannot measure* [18] incisively points out how vital measurements are in scientifically assessing our current situation, tracking progress, and evaluating effectiveness, etc.

From an engineer's point of view, the importance of measurement is threefold. First, it is essential for better understanding of the environment and assessing present conditions so that baseline goals can be established in terms of expected performance, productivity, etc., and realistic view of current situations and future possibilities can be attained. Secondly, with measurements come the knowledge of how the entities involved interact and the insight on how to make changes to processes and/or products that would help us reach our goals. Third, measurements encourage us to improve our processes and products. The focus of our study is software quality classification, and hence, we describe the importance of measurements with a specific reference to software related activities in the following subsection.

2.2 Software Measurements

Software measurement, once an obscure and esoteric specialty, has become essential to good software engineering [23]. Although not always acknowledged as essential to good software engineering, software metrics play an important role. By measuring characteristics of a software, developers can figure out whether the requirements are consistent and complete, whether the design conforms to the requirements, when the code can be tested, and the amount of resources required

during different phases of the software development process. Many effective project managers have successfully used various software metrics related to the process and the product to predict the project completion time and the amount of resources required for software development projects.

Software metrics can be classified mainly into three categories: process metrics, product metrics, and execution metrics. Process metrics quantify the software related activities associated with a time-scale. Product metrics quantify the attributes of the object or entity involved. Execution metrics measure the parameters involved during the execution of a program. Besides these three categories, some metrics are categorized as *Quality Metrics*, e.g., the number of faults in a module. Quality Metrics are very good indicators of the reliability of software. Practitioners and researchers alike usually try to predict the value of a quality metric in advance, using process, product, or execution metrics, to get an idea about the reliability of the software and guide the development efforts accordingly.

2.2.1 Product Metrics

Software product metrics can be categorized mainly into three groups: call graph metrics, control flow graph metrics, and statement metrics. Call graph metrics depict the relationship among procedures in terms of invocation. Control flow graph metrics indicate the flow of control from one statement to another. Statement metrics measure properties of program text without any inference on the meaning

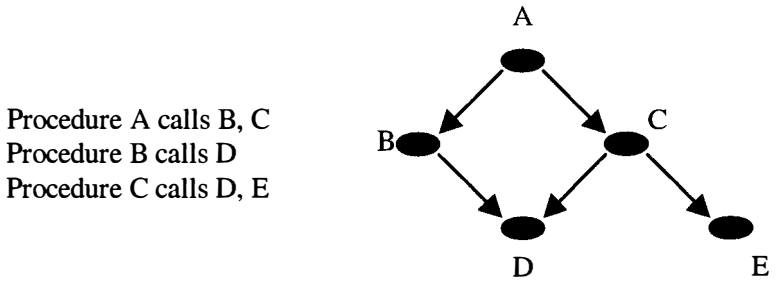

Figure 2.1: A Call Graph

of the text or the ordering of different components in a software module.

2.2.1.1 Call Graph Metrics

Call graph metrics are extracted from a very high level design of the software being developed, and hence, can be collected at a very early stage in the software life cycle. A directed call graph depicts how different procedures are invoked by examining the abstract model of the design. Figure 2.1 [23] shows one such example of a directed call graph. Number of distinct procedure calls, *CALUNQ*, and the number of second and following calls, *CAL2*, are some of the metrics that belong to this category.

2.2.1.2 Control Flow Graph Metrics

Control flow graph metrics graphically reveal the structural description of the algorithms in a given software module. They are concerned with the sequence in which instructions are executed in a program/software module, and are usually

modeled with *directed graphs*, where a node represents a program statement, and each directed edge (arc) represents the flow of control from one statement to another.

Number of nodes, number of arcs, in-degree of a node (the number of arcs arriving at the node), and the out-degree of a node (the number of arcs leaving the node) are some of the examples of control flow graph metrics. These metrics are available at a very early stage in the software development life cycle - right after detailed design is completed, even before the implementation phase begins.

2.2.2 Statement Metrics

Statement metrics are the measurements related to the property of text in a software module, with no inference on the meaning or ordering of program statements. The value of statement metrics essentially remains the same even if the order in which the program statements appear changes, and does not indicate what the program statements imply. Typical statement metrics used in practice include lines of code, number of executable statements, number of distinct include files, number of unique/total operators, and number of unique/total operands, etc.

2.2.3 Process Metrics

For successful completion of a software development project, one needs to give due attention to the associated process, and ideally, should not just focus on the product (software) being developed. Process metrics are the measurements related

to software development activities, and in many cases, are good indicators/predictors of the quality metric of interest.

For example, the experience of programmers is a very important factor. The more experienced the programmers, the less likely they are to introduce bugs in the software. The number of requirement errors found during inspection can be a good indicator of how effective the reviewing process actually is. Furthermore, the number of personnel working on the project within a period can give us insight into the resources needed for the development process.

2.2.4 Execution Metrics

Execution metrics are the metrics that measure the attributes concerning the execution of a software. The attributes measured could relate to the execution time under given conditions, e.g., *RESCPU*, *BUSCPU*, and *TANCPU*, or the consumption of resources, such as memory usage [44]. *RESCPU* is the execution time (microseconds) of an average transaction on a system serving consumers. *BUSCPU* is defined to be the execution time (microseconds) of an average transaction on a system serving businesses, whereas *TANCPU* is the execution time (microseconds) of an average transaction on a tandem system.

2.3 Software Metrics Used in This Study

For the case studies reported in this thesis, we used data from two C++ NASA projects, which are available through Metrics Data Program (MDP) website.

Table 2.1: Metric Description of JM1 and KC2 datasets

Metric Type	MetricNotation-JM1	Metric Notation-KC2
McCabe	Cyclomatic_Complexity	v(G)
	Essential_Complexity	ev(G)
	Design_Complexity	iv(G)
	Loc_Total	loc
Derived Halstead	Halstead_Length	N
	Halstead_Volume	V
	Halstead_Level	L
	Halstead_Difficulty	D
	Halstead_Content	I
	Halstead_Effort	E
	Halstead_Error_Est	B
	Halstead_Prog_Time	T
Line Count	Loc_Executable	loCode
	Loc_Comment	loComment
	Loc_Blank	loBlank
	Loc_Code_And_Comment	loCodeAndComment
Basic Halstead	Num_Unique_Operators	uniq_Op
	Num_Unique_Operands	uniq_Opnd
	Num_Operators	total_Op
	Num_Operands	total_Opnd
Branch	Branch_Count	branchCount

The two data sets are denoted by JM1 and KC2, the former being the larger (10,883 modules) of the two and the latter being the smaller (520 modules).

Both the JM1 and KC2 datasets contain 21 software metrics, which include the McCabe Metrics, the Halstead Metrics, the metrics of Line Count, as well as the metric of Branch Count. The metric description is listed in Table 2.1. Besides these twenty one metrics, KC2 has three quality metrics: *Error Rate* (number of defects in the module), *Defect* (whether or not the module has any defects), and *Defect Density*, whereas JM1 has two quality metrics: *Error Rate* and *Defect*. Out

of the available twenty one metrics, we used only the thirteen primitive metrics for our study. All the eight derived Halstead metrics were discarded. Also, we used only one quality metric, namely *Error Rate* (number of defects in the module), for the purpose of classification in our case studies. The class label *fp* (*fault-prone*) or *nfp* (*not-fault-prone*) was determined from the number of defects. An instance (module) was labelled *nfp* if it did not have any defect, and *fp* otherwise.

Chapter 3

METHODOLOGIES

This chapter aims to portray the theme of our study on noise handling in Software Quality Classification domain. Various aspects concerning data quality are introduced, and literature on different noise handling techniques reviewed. A conceptual framework for proposed noise handling technique is detailed, followed by detailed modeling methodology and a brief delineation of the different classification techniques used as a preamble to our approach.

3.1 Data Quality

Unfortunately, it is common for large datasets to have various kinds of errors, either random or systematic. According to [78, 87], unless an organization takes severe measures attempting to prevent data errors, the error rates involved in data entry and/or data acquisition typically range from 5 % or more.

3.1.1 What Affects Data Quality?

According to Tayi and Ballou [99], data quality is defined as fitness for use, which implies that quality is relative to the use of data. If the data have deficiencies,

generally known as noise, the data cannot land itself to use.

In practice, the quality, correctness, consistency, completeness, and reliability of a large dataset can be affected by several factors [22, 106, 107]. The dataset could have inconsistencies in terms of required format/syntax, semantics, or values, or it may have incorrect or missing values. The deficiencies could creep into the dataset for various reasons, such as poor interface design, data entry errors, failure of measurement device, lack of necessary information, subjectivity of the entity being measured, etc. No matter what the cause, data quality remains a prime concern in the fields involving extensive data analysis.

3.1.2 Aspects of Data Quality

There are several aspects to the quality of data, such as completeness, relevance, reliability, amount of data, consistency, correctness, timeliness, precision, unambiguity, accuracy, objectivity, conciseness, etc. [97, 105]. The list is not exhaustive, but is certainly representative of the type of attributes involved concerning the quality of data.

Reliability of data implies that the data stored is trustable, and can be taken as true information. Consistency of the data means that there is no contradiction between the data stored. When the data is objective, it means that the data does not depend on the judgement, interpretation, or evaluation of people. Informal definition for each of these attributes can be found in [6].

It should be noted that these aspects or dimensions of quality may actually be related to one another. For example, if the level of objectivity is low, i.e., if the data is subjective, then it does adversely affect the correctness and reliability of the data.

3.1.3 Why Is Data Quality Important?

Using the information at their disposal, organizations make well-informed decisions that improve their practice in order to attain their objectives and maintain a competitive edge in the market. The quality of data can have a great impact on the business decisions an organization may take. It is not difficult to realize what Redman [87] has so righteously noted - "decisions are no better than the data on which they are based". When the data suffers from poor quality, it may lose its usability, or can lead to incorrect decisions resulting in variety of losses.

Data with poor quality, when put to use, could very well translate into disastrous scenarios. The social and economic impact of poor-quality data could actually cost billions of dollars [61, 70, 79, 107]. One such example has been reported in [98], where hospital managers studied and used faulty information related to patients, and concluded that most of the patients suffered from the disease *hemorrhoids*. The managers allocated the hospital resources, such as number of beds, nurses, medical equipment, etc., accordingly to better serve the patients suffering from *hemorrhoids*.

This turned out to be a wrong and costly decision, not because the rationale in decision making was wrong, but because the information on which this decision was based was not completely accurate. The reason for inaccuracy in this case was poor interface design for data entry. At the check-in application, *hemorrhoids* was the default choice, and clerks selected it, because it was difficult to look for the correct choice. No matter what the cause was, the hospital finances suffered a great deal because of deficiency in data quality.

Indeed, data quality problems have become increasingly evident, especially in organizational databases. According to Tayi [98], 50 to 80% of computerized criminal records in the U.S. were found to be inaccurate, incomplete, or ambiguous. With the increasing use of computerized information to take important decisions that could even affect people's lives, the quality of such information/data has become important more than ever, and practitioners certainly realize it.

3.2 Coping with Noise

With any task that involves extensive data analysis and/or decision making based on available data, one needs to be vigilant of data quality issues. If the available data suffers from poor quality, i.e., has significant level of noise, appropriate noise handling procedures should be employed before the data is put to use.

For example, it is crucial in Knowledge Discovery in Database (KDD) processes to effectively handle noisy data for any data mining task and results to be

meaningful, applicable, and hence, valuable. Data Quality Mining, a deliberate application of data mining techniques for the purpose of data quality measurement and improvement, can supplement KDD, and contribute to improve the results of KDD projects, note Hipp et al. [38].

Over the past decade or so, researchers have proposed and studied various techniques, such as Data Quality Mining (DQM) techniques and statistical techniques to improve the quality of data. A brief survey of these techniques is delineated in the following subsections.

The problem of effectively dealing with noise can be approached mainly in three different ways. To cope with noise, one can either use robust algorithms, filter out noisy instances from the dataset, or try to correct noisy instances. A comparative study of three different noise handling techniques from each of this category has also been carried out, and is reported in [102].

3.2.1 Robust Algorithms

In the first approach, a robust learning algorithm is employed in such a way that the classifier built will not be overfitted to the training instances. This simple but subtle approach of handling noise finds its roots in the Occam's razor principle applied to Inductive Learning [33]. According to Li and Vitányi [65], the principle as originally stated - "Entities should not be multiplied beyond necessity" - could be interpreted as: "Among the theories that are consistent with the observed

phenomena, one should select the simplest theory".

Simply put, the Occam's razor principle, in the context of hypothesis formation, advocates the selection of the simplest hypothesis among all the hypotheses that best reflect the underlying concept of the training example set. According to Rissanen [89], the hypothesis thus selected would most likely be a generalization of the inherent concept(s), and could bring about significant improvement in the predictive accuracy of the algorithm.

This principle is fairly well-known in the inductive learning community, and has often been used not just to improve predictive accuracy of the algorithm, but also as a mechanism to handle noisy data by avoiding overfitting. Choosing the simplest structures to represent the underlying concept(s) and/or subconcept(s) over the complex ones (either by pruning in the case of tree-based algorithms or by truncating rules in the case of rule-based algorithms), perhaps at the expense of classification accuracy on the training example set in some cases, makes sure that the classifier does not become complex any more than necessary just to account for the noise. C4.5 [84] and CN2 [12] are examples of robust learning algorithms that come with in-built pruning mechanism.

Even though the principle has been successfully employed in various machine learning algorithms, there are some debatable issues which raise concerns, as Lavrač and Gamberger point out in [62]. Robust algorithms are appealing to practitioners, because they do not require any preprocessing of the data, but a classifier thus built,

i.e., the hypothesis thus formed, may have been influenced by the presence of noisy instances in the dataset, warn Gamberger et al. [31]. Also, the term *complexity* for a learning algorithm is a loosely defined term, and is subject to how one perceives *complexity*. That is why there is no single and universally agreed upon complexity measure to our avail. There are several different complexity measures available, such as Kolmogorov complexity based measure [65], Minimum Description Length (MDL) [89], and Tree size in the case of tree-based algorithms, etc.

Different complexity measures, such as Kolmogorov measure [65] and MDL [89], when used to select the simplest hypothesis, may in fact select different hypothesis for the same training dataset, caution Lavrač and Gamberger [62]. Therefore, choosing the most appropriate complexity measure for a given learning algorithm can be tricky at times.

The other issue of concern is that applying Occam's razor principle may not always yield the best predictive accuracy. In the empirical work [86] that threatens the validity of Occam's razor principle, boosting and bagging techniques (rather complex techniques) were found to yield better predictive accuracy.

Despite all these, Occam's razor principle is valid, and can be applied, provided certain conditions are fulfilled. Gamberger and Lavrač discuss the conditions required to be met for the principle's applicability, and present related theorems in an elaborative manner in [33].

3.2.2 Noise Elimination/Filtering

Many researchers [9, 31, 42] have explored the noise elimination approach to improve data quality by identifying and eliminating instances evaluated as being noisy according to certain criteria before applying the chosen algorithms. This is a rather direct approach which attempts to improve the quality of the input data for hypothesis formation so that the noisy instances do not influence the hypothesis constructed [31]. The concept of removing the instances suspected of being noisy resembles the approach used in robust regression and outlier detection techniques in statistics [91].

Brodley and Friedl [9] have introduced a method for identifying and eliminating mislabeled instances. It should be noted that although noise in training examples may be due to erroneous attribute values and erroneous class labels, machine learning algorithms usually treat noisy examples as being mislabeled [32]. This would mean that the method proposed by Brodley and Friedl [9] is not just applicable for removing instances with class noise, but is also applicable for eliminating instances that have corrupt attribute values.

The technique was inspired by a similar approach employed for removing outliers in regression analysis [108]. The fundamental concept behind the method is to use a number of learning algorithms that would filter out the instances likely to contain noise on the basis of misclassification by majority or consensus. The first step involves identification of the instances likely to contain noise. All the training

examples/instances are classified using m different learning algorithms (called the filter algorithms) by performing n-fold cross-validation. In the second step, all the training instances that are misclassified either by the majority or by all of the m base level classifiers (filter algorithms) are eliminated, and then input to the final learning algorithm(s). The results reported in [9] are quite promising, and have empirically substantiated that filtering can, in fact, improve classification accuracy for the datasets that suffer from poor-quality data.

It is obvious that the technique is a generalized method for noise removal, which can be used regardless of the learning algorithm(s) selected for filtering out noisy instances. The approach is distinct from other previous approaches in that it assumes that the data errors are independent of the particular model being fit to the data, and attempts to identify the datapoints that would be outliers in any model(s), explain Brodley and Friedl [9].

While noise elimination / instance selection has been shown to improve the performance of learning algorithms significantly [9, 10, 31, 32, 102], one has to be judicious in removing instances suspected of being noisy to balance the amount of noise removed from the data set and the amount of data retained for training. When only meager amount of data is available, this approach may or may not be feasible.

Since our study extends the approach proposed by Brodley and Friedl [9], the procedure involved was presented in an elaborate fashion. Various other instance

selection / noise elimination techniques have long been used to improve the performance of different learning algorithms, and instance-based techniques in particular, a brief survey of which is presented in the subsection that follows. A comprehensive literature survey can be found in [9].

3.2.3 Related Work

Wilson [109] used a 3-NN classifier to select instances that were then used to form a 1-NN classifier; the instances that were misclassified by the 3-NN classifier were eliminated from the instances that would be used to build the 1-NN classifier.

Extending the same approach, Tomek [103] experimented with several increasing values of nearest neighbors as a mechanism for elimination of instances. Wilson and Martinez [110, 111] have incorporated this approach into a suite of instance selection techniques for exemplar-based learning algorithms.

Aha, Kibler, and Albert [1] showed that if the instances are selected on the basis of their contribution towards the classification accuracy of an instance-based classifier, the accuracy of the resulting classifier can be improved.

A comprehensive overview of instance selection techniques for exemplar-based learning algorithms can be found in [111]. Applicability of the instance selection techniques is not limited to instance-based classifiers. It has also been applied to other types of classifiers. Winston [112] demonstrated the utility of selecting "near misses" when learning structural descriptions.

Skalak and Rissland [95] have proposed an instance selection mechanism that uses a case-based retrieval algorithm's taxonomy of cases for a decision tree algorithm. Lewis and Catlett [64] demonstrated that instances could also be selected by using an estimate of classification certainty.

Gamberger et al. [31, 32] have taken a different approach to dealing with noise. They have proposed a technique to identify and eliminate noisy examples from the training set by using a simple compression measure, namely MDL (Minimal Description Length). First, all the inconsistent examples from the training data are removed. Subsequently, the features are transformed into a binary feature set. An examination is carried out to see which set of examples, when removed, would reduce the complexity in terms of MDL so that the current set of instances would be consistent. By eliminating appropriate examples this way, a consistent and complete hypothesis can then be built from the set of remaining examples by using a learning algorithm, not necessarily a robust (noise-tolerant) one. Zhu et al. [117] have introduced a new strategy to identify and eliminate noisy instances on a partition-based scheme that is particularly useful when dealing with distributed and/or large datasets.

To address the problems of inaccuracies in feature measurements, Zhao and Nishida [116] have adopted fuzzy logic approach to represent and calculate inaccuracies in the training data. Noise in the attribute values are identified using qualitative correlations among different attributes. For example, when n-1 out of n symptoms

indicate that a patient has a particular disease, then it is quite possible that the n^{th} symptom was incorrectly measured or entered [9]. Zhao and Nishida's [116] method dynamically determines fuzzy intervals for inaccurate data, and calls for domain knowledge to be able to divide the features into sets whose members are qualitatively dependent.

A study by Marcus and Maletic [69] demonstrates that Association Rule Mining can be useful in identifying not only interesting patterns in various fields of interest, but also patterns that uncover errors in the data sets. In their study [69], Marcus and Maletic investigated the use of Ordinal Association rules to identify potential errors in the dataset with reasonably low computational complexity and high efficiency.

Guyon et al. [35] have described a method for data cleaning by discovering meaningless or garbage patterns likely to be noise. In their paper on Data Quality Mining, Hipp et al. [38] have also explored the use of association rules as a means to detect, quantify, explain, and correct data quality deficiencies.

3.2.4 Polishing

Teng [100, 101, 102] has explored a different approach, an approach he calls *polishing*, in which instead of removing the instances identified as being noisy, corrections are made to either one or more features or the class label of the instances suspected of being noisy. If employed correctly, this approach could approximate a

noise-free condition preserving maximal information.

The concept of *Polishing*, introduced in [100, 101, 102], takes advantage of the fact that different components in a data set may not be totally independent except in the case of irrelevant attributes. The interdependence between different components in a data set is the driving factor of the whole process. Conceptually, this technique of coming up with possible correction values using interdependence between different components in the dataset sounds very similar to using association rules.

As Teng describes in [100, 101, 102], the basic algorithm for polishing has two stages, namely, prediction and adjustment. In the prediction phase, attributes suspected of being corrupted are identified, and appropriate replacement values are suggested for each of these attributes. The replacement values are obtained by swapping the role of the target class and the attribute of interest for each of the attributes in the dataset. When a predicted value of the attribute is different from its original value in the dataset, the predicted value is a potential correction for that particular attribute. Adjustment phase consists of selectively correcting values of the suspected attributes of the identified noisy instances in the dataset. A detailed description of the procedure can be found in [100].

This procedure has a distinctive advantage over other noise handling procedures in the situations where the size of the dataset is small, making it unfeasible to toss out the noisy instances from the dataset, or where data recollection is costly

or practically impossible. Also, correcting noisy instances rather than eliminating has been shown to give better results in some cases [20].

While *polishing* has its merits in identifying and correcting noisy instances, it is no silver bullet. One still needs to be wary of the fact that the method comes with some limitations and certain degree of risk associated with it.

The major detrimental aspect of *polishing* is that it is applicable to datasets with nominal attributes only, which would mean that for datasets with real/numeric attributes, one cannot really correct the noisy instances without having to perform discretization. As Teng [100, 101, 102] points out, another limitation of the procedure is the time complexity involved. Not only does one need to build significant number of classifiers (models) with each of the attribute swapped with the respective class label, but one also needs to run down a potentially very long list of suggested changes in order to come up with appropriate replacement value(s) for respective attribute(s). The amount of time thus consumed may, in many cases, turn out to be a restricting factor.

In attempts to correct the noisy data, one may unintentionally introduce further noise, cautions Teng [100]. The task of correcting noisy instances can indeed be a precarious one. The suggested values from the prediction phase are not full proof, and can be bizarre because of the unfortunate fact that the process itself is based on imperfect data. The suggested correction values can contain errors also if there is a little or no degree of interdependence between different features and class

labels as in the case of irrelevant attributes.

In this approach, the adjustment procedure tries to change the attribute values even when there is noise in the class label and not in the attribute values of the instance. While the suggested corrections may be appropriate for the noisy class label, in effect, they serve nothing more than to introduce more noise. This effect, according to Teng [100], becomes prominent in data sets where a small number of attributes are highly predictive of the class label, as then only a few changes to these attributes would be enough to fit the altered class value.

There is still some room for improvement in the way *Polishing* is implemented [100, 101, 102] for data correction. *Polishing* fails to treat an instance fairly in the case when both the class label and some of the attribute values of the instance have corrupted values, because in the current implementation, the attribute values for a noisy instance are changed first if possible, and if that fails, the class label is changed appropriately. Teng [100] also points out that a more stringent criterion needs to be adopted to subject more noisy instances to data correction.

3.3 Our Approach to Noise Handling

There are many similarities between our approach to noise handling and the approach that Brodley and Friedl [9] have employed. But there are some major differences too. In essence, one can say that our study leverages the work done by

Brodley and Friedl [9] in order to effectively handle the noise. Because of the limitations and risks associated with *polishing*, noise elimination with ensemble classifier approach was deemed appropriate for our study. Also, the size of the dataset was not compellingly small to choose *polishing* over *noise elimination.*

Similar to [31, 32], we first remove inconsistencies from the dataset, i.e., the instances for which we found inconsistent class labels. This was achieved by clustering the instances according to their feature values, and then removing the instances for which the class labels were different, but the attribute values were identical.

Subsequently, an approach similar to the ensemble classifier approach [9] was used to identify and eliminate possibly noisy instances. Ensemble classifiers combine the outputs of a set of base-level classifiers [4, 36, 114].

In their ensemble classifier approach towards noise elimination, Brodley and Friedl [9] have reported results with only three different base-level classifiers, and Brodley and Utgoff [10] have carried out similar experiments with five different base level classifiers. In our opinion, the number of base level classifiers can be a key factor when ensemble classifier approach is used in noise detection and elimination. Especially, with regards to [9], one can argue that it is quite possible that two out of the three different classifiers or even all the three classifiers could misclassify genuinely noise-free instance(s). On the same line of argument, we can say that even the use of five base level classifiers may not be conservative enough.

Indeed, it would be naive to form an opinion about a module being noisy by considering only a small number of classifiers, because the appropriateness (bias) of the chosen learning algorithm applied to a particular dataset also plays a significant role. It may well be that the few chosen classifiers don't have appropriate bias to learn the concepts for a given domain. It is well known in the inductive learning community that classifiers do not perform consistently well across different domains. For example, a classifier which performs well in Software Quality Classification domain may not fare as well in a Medical Diagnosis domain. Experimenting with a rather large number of classifiers can ensure that we are reducing the probability of throwing away good data and raising the level of confidence in the identification of noisy modules.

In our study, we used 25 different base-level classifiers from different categories, such as bayesian, instance-based, rule-based, decision-tree based, pattern-based, and statistical techniques, etc., for our ensemble classifier approach towards noise elimination.

Unlike Brodley and Friedl's [9] approach that only considers majority filtering and consensus filtering, the former being the least conservative, and the latter being the most conservative, our study attempts to examine the effects of several different levels of conservative approach to noise elimination on predictive accuracy of classifiers. We experimented with four different levels of filtering. In our work, we decided to eliminate the instances misclassified by 23 or more classifiers (the most

conservative approach, i.e., misclassification by over 90% classifiers), 20 or more classifiers (misclassification by over 80% classifiers), 17 or more classifiers (misclassification by over 68% classifiers), and 13 or more classifiers (the majority filtering approach - the least conservative one).

Most importantly, all the empirical work towards noise detection and handling we have come across [9, 10, 100, 101, 102] have a flaw in that the quality of evaluation set is ignored. Without having a noise-free evaluation set, the predictive accuracies reported for different classifiers with different noise handling techniques may not be the true indicators of how the noise handling technique(s) fared. It is certainly not fair to any algorithm or any noise handling technique when the results are compared in terms of predictive accuracy of a classifier without ensuring that the evaluation set used was in fact noise free. In order to address this issue, we perform filtering on the dataset before generating impartial splits for training and evaluation set. The filtering is performed on the basis of the misclassification by ensemble of classifiers with cross-validation. Thus, our approach cleans not only the training set but also the evaluation set. This would give us a better idea of classifiers' predictive accuracy and efficacy of our noise handling approach than would any other existing approach.

Other than the factors mentioned above, our approach, theoretically speaking, is similar to Brodley and Friedl's [9] approach.

3.3.1 Handling Exceptions

Danyluk and Provost [17] note that learning from noisy data is difficult because it is hard to distinguish between instances that are noisy and instances that are exceptions to the general rule, especially if the noise is systematic. Brodley and Friedl also indicated in their paper [9] that one has to be cautious not to unknowingly remove exceptions from the dataset while trying to eliminate noisy instances. Several researchers [35, 76, 77, 96] have done work on how to distinguish exceptions from noise.

While it is indicated that further research is required to address this issue with ensemble-classifier approach in [9], we think that our ensemble-classifier approach, especially the approach with the most conservative level of filtering, does counteract the problem to a certain degree.

It is true that not all the classifiers can capture the atypicality of the instances that are exceptions to the general case. However, with our most conservative approach, it is likely that at least three of the twenty five classifiers would have the appropriate bias that could allow them to correctly classify exceptions or the instances that are "*hard-to-classify*". This would mean that our most conservative approach, where all the instances misclassified by 23 or more classifiers are eliminated, is the least likely of all the four different levels of filtering approaches to take exceptions for noise and eliminate those instances inadvertently. However, it should

be noted that this study was not particularly aimed at addressing this issue, because from induction point of view, exceptions have the same effect on the induction process as erroneous examples themselves [62].

3.3.2 Distinction of Our Approach

To our knowledge, our study is one of the few studies that examine the effect of a noise handling technique on a real-world data with inherent noise. Many empirical studies have been carried out evaluating different noise handling mechanisms on the datasets in which noise is artificially injected [9, 10, 100, 101, 102, 117], either in the class label or in the feature values. One potential problem with this approach is that a naturally occurring noisy class label may get changed to correct value, or irrelevant attributes might get their values changed. The effect of amount of noise handled (corrected/removed) on classification accuracy may not give the right idea in this situation.

Noise free evaluation dataset is available because of the way filtering is performed in our noise handling procedure.

Number of classifiers is rather large, and different learning algorithms from different categories have been chosen to form a set of base-level classifiers. This enables us to use different levels of filtering to eliminate noisy instances, avoids results from being influenced by inappropriate bias of a few classifiers, and raises the confidence level in the process of tossing out the instances suspected of being

noisy.

3.4 Classification Modeling

We deployed the proposed noise elimination technique on software quality data, as the domain of interest is Software Quality Classification. Twenty five different classification techniques[1] were used for the purpose of Software Quality Classification, a proven technique in achieving better software quality control [21, 74, 75, 94].

Typically, a two-group classification, in which software modules are classified as either *fault-prone*(fp) or *not fault-prone*(nfp), is employed for software quality classification. In the context of two-group classification, two types of misclassification can take place - false positive (Type I) and false negative (Type II). Type I error occurs when a *not fault-prone* module is misclassified as *fault-prone*, and Type II error occurs when a *fault-prone* module is misclassified as *not fault-prone*. All the different notations used in this study have been tabulated in Table 3.1.

3.4.1 Objective for Classification Models

It is well-known in the Software Quality Engineering community that there is a significant disparity between the costs of the two types of misclassification. For software development projects, Type II errors are invariably more severe in terms of the cost involved. As opposed to extra reviews involved when software modules are

[1] Each classification technique is briefly described in Section 3.6.

Table 3.1: Notations

Symbol	*Description*
ECM	Expected Cost of Misclassification
NECM	Normalized Expected Cost of Misclassification
fp	A fault-prone or high risk module
nfp	A not fault-prone or low risk module
C_I	Cost of Type I misclassification error
C_{II}	Cost of Type II misclassification error
π_{fp}	Prior probability of fp modules
π_{nfp}	Prior probability of nfp modules
p	The p-value for hypothesis testing
α	The significance level for hypothesis testing
CBR	Case-Based Reasoning [57, 63]
TD	The Treedisc classification tree algorithm [48]
LR	Logistic Regression [46]
LOC	Lines-of-Code
GP	Genetic Programming [3, 59]
ANN	Artificial Neural Network [67, 73]
LBOOST	LogitBoost [29]
RBM	Rule-Based Modeling [68]
BAG	Bagging [7]
RSET	Rough Sets [58]
MCOST	Meta Cost [19]
ABOOST	AdaBoost [27]
DTABLE	Decision Table [55]
ADT	Alternating Decision Trees [26]
SMO	Sequential Minimal Optimization [81]
IB1	1-Instance Based Learning
IBK	k-Instance Based Learning
PART	Partial Decision Trees [25]
ONER	OneR [40]
JRIP	Repeated Incremental Pruning to Produce Error Reduction Algorithm [13]
RDR	Ripple Down Rules [14, 15]
J48	Implementation of C4.5 algorithm [85]
NBAYES	Naive Bayes [24]
HPIPES	Hyper Pipes [80]
LWLS	LWL Stump [2]

misclassified according to Type I, Type II error involves inspection and correction after the software product becomes operational, which is obviously costlier, and can also hurt the organization in terms of its reputation and credibility.

While Type II errors (misclassifications) can cost more than Type I errors, the cost ratio $\frac{C_{II}}{C_I}$, ratio of the cost of misclassification of Type II to the cost of misclassification of Type I, is not constant, and varies depending on the quality improvement needs of the software development project and also the application domain and the nature of the system being developed. For example, for mission-critical and high-assurance systems, the cost-ratio $(\frac{C_{II}}{C_I})$ could be as high as 100, and on the contrary, for non-critical business applications, the cost-ratio $(\frac{C_{II}}{C_I})$ could be as low as 10.

Looking at the vast disparity between the costs of the two types of misclassification, one might think that building a classification model with the lowest Type II error would be a very good strategy since it would be able to detect as many *fault-prone (fp)* modules as possible. While this strategy is certainly appealing, one cannot just overlook Type I error. In practice, it is observed that as the Type I error increases, Type II error decreases, and vice versa [46, 48, 49, 50, 52]. Hence, a classification model with very low Type II error is likely to have Type I error of a very high magnitude, misclassifying many *not fault-prone(nfp)* modules as *fault-prone(fp)* modules in an attempt to correctly classify as many *fault-prone (fp)* modules as possible. This strategy may not be feasible when software development

organizations are confronted with limited quality enhancement resources.

A more pragmatic approach, one that is recommended by many software quality engineers, would be to select a classification model that offers balanced Type I and Type II errors, with Type II as low as possible. Adopting this strategy would mean that not only the Type II error rate is reasonably low, which ensures detection of significantly large number of *fault-prone (fp)* modules, but also the Type I error is reasonably low, which keeps the number of ineffective reviews and testing of the modules predicted to be *fault-prone (fp)* in check. Taking all these into consideration, we opted to select this model selection strategy for our case study with both the software systems (JM1 and KC2).

However, one should be aware that the appropriateness of a model-selection strategy also depends on the nature of the system being modeled and the amount of quality enhancement resources at the organization's disposal. If ensuring maximum software reliability is the prime concern, and there is no constraint on the resources to be expended for quality improvement, a classification model that offers the lowest Type II error, irrespective of Type I error, would be of interest. Such a strategy would be more appropriate for mission-critical software systems.

In summary, the classification modeling objectives for the two case studies presented in this study were: (1) to select appropriate classification model for each classification technique, (2) to use the predictions of the classification techniques to perform noise elimination using the proposed ensemble classifier approach, and (3)

to evaluate predictive performance of different classification techniques in terms of ECM (Expected Cost of Misclassification) values computed for different cost ratios on the data sets with different levels of noise.

3.4.2 Calibrating Classification Models

Every attempt was made to make the empirical investigation an impartial and unbiased one. We adopted common model-selection and model-evaluation strategy for all the twenty five classification techniques used as the base level classifiers in the noise elimination based on our ensemble-classifier approach.

The software quality modeling process involved building, selecting, and validating classification models for all the twenty five classification techniques on datasets with different level of noise. The process was carried out in the following steps:

1. *Preprocessing and Formatting Data:* Various classification modeling tools, such as SAS, WEKA, SMART, RBM, etc., were used to perform the necessary software quality classification. The input data to each of the tools had to be converted into the format acceptable by the tool.

2. *Building Models:* The datasets for both the software systems (JM1 and KC2) were proportionately split (before noise elimination and after noise elimination at various levels of noise filtering) into two halves to create training (fit) and evaluation (test) datasets. 10-fold cross validation was performed on the training dataset to build classification models with almost all the classification techniques, with a few

exceptions[2] of resubstitution (in the case of Rule-Based Modeling (RBM), Treedisc, Logistic Regression, Lines-of-Code, and GP) and n-fold cross-validation (in the case of Case Based Reasoning (CBR)).

For building classification models, various parameters specific to the modeling tool/classification technique and the cost of misclassification were varied. For example, when building a classification model for J48 (WEKA's implementation of the famous decision tree algorithm, C4.5), parameters, such as Pruning Confidence (C), Minimum number of instances per leaf node(M), etc. and cost of misclassification, were varied to build classification models. The model, which best satisfies our model-selection strategy, among all the possible choices, was selected to be applied to evaluation dataset. This procedure is very similar to the generalized classification rule presented in [47] to classify software modules as either *fp* or *nfp*.

It is worth mentioning here that the original unsplit dataset (the dataset with 8850 modules for JM1 and the dataset with 520 modules for KC2 system) was used to build a classification model each (mostly with 10-fold cross validation, as mentioned earlier) for all the twenty five classifiers to form the basis for the proposed noise filtering. A distinctive advantage of performing noise elimination this way is the availability of evaluation dataset, besides the training set, both with reduced level of noise, upon splitting the dataset after noise removal.

---

[2] Exceptions had to be made either because of infeasibility of cross-validation for the technique or because of limitation of the modeling tool used.

3. *Selecting and Evaluating Models:* The classification model selected based on the quality of fit for each classifier was applied to the evaluation set to assess the classifier's predictive performance as compared to its counterparts at a given level of noise filtering. The performance of each classifier was evaluated in terms of Expected Cost of Misclassification (ECM), a function of Type I and Type II errors. Using a singular measure for comparison makes the task of comparing the performance of twenty five different techniques on the datasets with different levels of noise much simpler.

3.4.3 Expected Cost of Misclassification

It was observed by our research group [48, 50, 52, 83, 90] that comparing different classification methods just based on the two misclassification rates is rather difficult, and that there is a need for a singular measure to make the task relatively easier.

One may argue that overall misclassification error is a possible alternative for the use as a singular measure that can facilitate comparison of performance of different classification techniques. While the argument is certainly valid, it should be noted that our study addresses software quality classification for two high-assurance software systems (JM1 and KC2), for which, as we mentioned earlier, there is likely to be a vast disparity between the costs of the two types of misclassification and also between the proportions of the two classes (*fp* and *nfp*). If we were to use overall

misclassification as a means to compare the performance of different classification techniques, it would mean that we would be neither discriminating between the two types of misclassification nor taking the difference in the class (group) population into consideration.

This led us to use ECM (Expected Cost of Misclassification) as a unified singular performance measure to compare the performance of different classification techniques on datasets with different levels of noise. The ECM measure (Equation (3.1)) takes prior probabilities of the two classes and the costs of misclassifications into account [43], and hence is considered a practically useful measure for evaluating the performance of different classification techniques in the context of Software Quality Classification [53]. Obviously, a preferred classification model is the one that yields a low Expected Cost of Misclassification.

Practically speaking, in many cases, it may not be possible for an organization to quantify the costs of the two types of misclassification. In order to overcome this problem, Normalized Expected Cost of Misclassification(NECM), which facilitates the use of the cost ratio, $\frac{C_{II}}{C_I}$, instead of individual misclassification costs by normalizing the value of ECM with respect to C_I (Equation (3.2)), is employed.

$$\text{ECM} = C_I Pr(fp|nfp)\pi_{nfp} + C_{II} Pr(nfp|fp)\pi_{fp} \tag{3.1}$$

$$\text{NECM} = \frac{\text{ECM}}{C_I} = Pr(fp|nfp)\pi_{nfp} + \frac{C_{II}}{C_I} Pr(nfp|fp)\pi_{fp} \tag{3.2}$$

The prior probabilities of the *fp* and *nfp* classes are given by, π_{fp} and π_{nfp} respectively. $Pr(fp|nfp)$ is the proportion of the *nfp* modules incorrectly classified as *fp*, and conversely, $Pr(nfp|fp)$ is the proportion of the *fp* modules incorrectly classified as *nfp*. The prior probabilities, i.e., π_{fp} and π_{nfp}, are estimated as the respective proportions in the given data set. We compared the performance of different classification techniques in terms of ECM [3] at different cost ratios, i.e., by varying $\frac{C_U}{C_I}$ in Equation (3.2).

In practice, the actual costs of misclassifications are unknown at the time of modeling. In order to make the empirical investigation more realistic and applicable, we explored a range of values (10, 20, 30, and 50) for the cost ratio $\frac{C_U}{C_I}$ to compute the ECM. Evaluating models across a range of cost ratios can also shed some light on the sensitivity (robustness) of a classification model with respect to the possible costs and effort values.

There is a noticeable difference between the model-selection and the model-evaluation approach we have adopted. In the model selection, we strive to achieve a preferred balance between the Type I and Type II errors (Type I and Type II errors as balanced as possible, with the lowest Type II), whereas the model evaluation approach is based on ECM. Looking at the practical usefulness of the ECM measure, one may wonder why it was not selected as the basis for model selection strategy.

[3] In this study, the notation ECM refers to the normalized expected cost of misclassification (NECM), and the two are used interchangeably.

One of the main reasons is the fact that it is very difficult to estimate the actual cost ratio $\frac{C_{II}}{C_I}$ at the time of modeling. Also, if the practical quality improvement objectives (See Section 3.4.1) are taken into account, it is not difficult to realize why the use of ECM as the basis for model selection may not be such a good idea even though it is a very good practical performance evaluation measure. For example, for a cost ratio of 100 (empirical upper bound for software systems), if a classification model demonstrates a very low Type II error rate and a high Type I error rate, its ECM value is likely to be very low, leading to conclusion that it be selected as the preferred model. However, as explained in Section 3.4.1, this may not be feasible when software organizations encounter the problem of limited resources for software quality improvement, which is often the case.

While our model-selection and model-evaluation approaches have been well justified, the reader should be aware of the underlying assumption that a model, having been selected according to the preferred balance criterion, would most likely (but not necessarily) generate balanced misclassification rates, when evaluated with the test dataset. Having a low value of ECM does not necessarily mean the misclassification error rates are balanced. However, tracking the stability and robustness of model performance (across training and evaluation sets) is out of scope for this study, and can be addressed in future work.

3.4.4 Two-way ANOVA: Randomized Complete Block Design

Analysis of Variance, commonly known as ANOVA, is a popular statistical technique for examining whether three or more independent groups or populations are significantly different from one another. We performed Two-way ANOVA: Randomized Complete Block Design to compare the performance of twenty five different classifiers on datasets with different levels of noise for two different software systems (JM1 and KC2) to observe if the different classification techniques and the different levels of noise filtering were significantly different from their respective counterparts.

Two-Way ANOVA: Randomized Complete Block Design modeling approach [5, 71] involves classifying n *heterogeneous* subjects into r homogeneous groups, called blocks so that c subjects in each block can then be randomly assigned, one each, to the c levels of the factor of interest prior to the performance of a two-tailed F test, to determine the existence of significant treatment effects (Note that $n = rc$).

The primary reason to select this experimental design, one that separates subject variability from variability within data, is to reduce experimental error as much as possible. The observed data for each dataset with a specific level of noise filtering constitutes a replication. Since for each such dataset, the observed data is not affected by the level of noise filtering, blocking by dataset with specific level of noise filtering will make the experiment more powerful by reducing the experimental error variability [5].

NECM (Normalized Expected Cost of Misclassification), computed for each

classification technique applied to the datasets with different levels of noise filtering, was employed as the response variable in our experimental design. It is worth noting here that there are certain implicit assumptions, such as normality of the data and randomness of the variable, etc., that come with ANOVA, and that we did not observe significant deviations from these assumptions in our empirical investigation.

Experimental design models were built using the NECM values computed for the four different cost ratios ($\frac{C_M}{C_I}$) : 10, 20, 30, and 50. Two-Way ANOVA models for our comparative study involved 25 factor treatments (twenty five classification methods), and 5 blocks (one block for the original noisy dataset, and the other four blocks for the datasets with noise filtering by agreement of 13 or more, 17 or more, 20 or more, and 23 or more classification techniques) for JM1 system and 4 blocks (one block for the original noisy KC2 dataset, and the other three blocks for the datasets with noise filtering by agreement of 13 or more, 17 or more, and 23 or more classification techniques) for KC2 system.

To develop the ANOVA procedure for a randomized complete block design, Y_{ij}, the observation in the i^{th} block of B ($i = 1, 2, ..., b$) under the j^{th} level of factor A ($j = 1, 2, ..., a$), can be represented by the model,

$$Y_{ij} = \mu + A_j + B_i + \epsilon_{ij} \tag{3.3}$$

where,

$\mu =$ *overall* effect or mean common to all observations.

$A_j = \mu_{.j}$ - μ, a treatment effect peculiar to the j^{th} level of factor A (classification technique).

$B_i = \mu_{i.}$ - μ, a block effect (dataset with a specific level of filtering) peculiar to the i^{th} block of B.

ϵ_{ij} = *random variation* or *experimental error* associated with the observation in the i^{th} block of B under the j^{th} level of factor A.

$\mu_{.j}$ = true mean for the j^{th} level of factor A.

$\mu_{i.}$ = true mean for the i^{th} block of B.

Y_{ij} is NECM value in the context of this study.

The two-way ANOVA block design results for the twenty five classification techniques are presented in the Section 4.5.

3.4.5 Multiple Pairwise Comparisons

When comparing more than two means, an ANOVA F-Test is useful to determine if the population means are significantly different from each other or not; however, it does not indicate which means differ from which of the other means. Multiple comparison methods are useful in obtaining detailed information about the differences among the various population means. A variety of multiple comparison methods are available, such as Fisher's least-significant-difference test, Tukey's test, Scheffe's test, Bonferroni's test, and Waller-Duncan k-ratio t-test [5, 41, 93, 104].

In our comparative study, we employed the Tukey's multiple comparison

test [37, 41, 60], in which the two means are declared significantly different by the Tukey-Kramer criterion if

$$|\bar{Y}_{.i} - \bar{Y}_{.j}|/s\sqrt{(1/n_i + 1/n_j)/2} \geq q(p; c, n-c) \tag{3.4}$$

where $\bar{Y}_{.i}$ and $\bar{Y}_{.j}$ are the means of group i and group j respectively, n_i and n_j are the number of observations in the two groups, s is the root mean square error based on $n-c$ degrees of freedom, p is the significance level, and $q(p; c, n-c)$ is the p-level critical value of a studentized range distribution with c and $n-c$ degrees of freedom. For equal group sizes, Tukey's method rejects the null hypothesis of equal population means if

$$|\bar{Y}_{.i} - \bar{Y}_{.j}| \geq q(\alpha; c, n-c)s\sqrt{n} \tag{3.5}$$

3.5 Z-Test Comparison of Two Proportions

The following approximate testing procedure was used for statistically comparing two proportions [115]. Using this procedure, we compared two proportions of the instances identified as noisy with two different noise filtering approaches. The results are reported in Section 4.7.

Let $\hat{p}_1 = X_1/n_1$ and $\hat{p}_2 = X_2/n_2$ be the two proportions, where X_i is a count for a sample of size n_i. If we want to test the hypothesis $H_0 : p_1 = p_2$, with alternate hypothesis $H_A : p_1 > p_2$ then

$$Z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\frac{\bar{p}\bar{q}}{n_1} + \frac{\bar{p}\bar{q}}{n_2}}} \tag{3.6}$$

where $\bar{p} = \frac{X_1+X_2}{n_1+n_2} = \frac{n_1\hat{p}_1+n_2\hat{p}_2}{n_1+n_2}$, and $\bar{q} = 1 - \bar{p}$.

When $n_1 = n_2$, as in our case, then we get the following reductions:

$$Z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{2\frac{\bar{p}\bar{q}}{n}}} \tag{3.7}$$

where n is the common sample size, and $\bar{p} = \frac{\hat{p}_1+\hat{p}_2}{2}$.

Assuming normal distributions for $\hat{p}_1$ and $\hat{p}_2$, we determine the proportion of the normal curve which is greater than or equal to the computed value of Z. This proportion is the level of significance. It is most easily found in a table of proportions of the normal curve (one-tailed) by looking up the proportion corresponding to the z-value. If the level of significance is less than a specified limit (usually 5%), the null hypothesis is rejected.

3.6 Software Quality Classification Techniques

This section presents a brief description of the classification techniques used as base-level classifiers to construct the ensemble classifier for filtering out noisy instances. The aim of this section is to give a brief overview of the classification techniques involved, and not (due to lack of space) to present an extensive algorithmic detail. Our research group has performed extensive empirical research in the area of Software Quality Classification modeling using all of the methods discussed herein.

3.6.1 Case-Based Reasoning

Case-Based Reasoning (CBR) [57, 63] is a technique that aims to find solutions to new problems, based on past experiences, which are represented by "cases" in a "case library". The case library and the associated retrieval and decision rules constitute a CBR model. In the context of a classification problem, each case in the case library has known attributes and class membership. The working hypothesis of CBR is that an instance under examination has probably the same class label as the instance(s) with similar features or attributes.

A CBR system can take advantage of availability of new or revised information by adding new cases or removing obsolete cases from the case library. Its good scalability provides fast retrieval even as the size of the case library scales up. CBR systems can be designed to alert users when a new case is outside the bounds of current experience.

3.6.2 TREEDISC

TREEDISC is a SAS macro implementation of modified CHAID algorithm [45]. It constructs a decision tree to predict a specified categorical dependent variable from one or more predictor (independent) variables. The decision tree is computed by recursively partitioning the data set into two or more subsets of observations, based on the categories of one of the predictor variables until some stopping criterion is met. The variable that is most significantly associated with the dependent variable

according to a chi-squared test of independence in contingency table is selected to be the predictor variable.

Decision tree-based models are built by varying model parameters in order to achieve the preferred balance between the misclassification error rates, and to avoid overfitting of classification trees [54].

3.6.3 Logistic Regression

Logistic Regression is a statistical modeling technique that offers good model interpretation. Independent variables in logistic regression may be categorical, discrete, or continuous. However, the categorical variables need to be encoded (e.g., 0, 1) to facilitate classification modeling.

Let x_j be the j^{th} independent variable, and let $\mathbf{x}_i$ be the vector of the i^{th} instance's independent variable values. In the context of two-group classification, an instance belonging to one of the two classes can be designated as an 'event'. Let q be the probability of the event, and thus $\frac{q}{(1-q)}$ is the odds of the event. The Logistic Regression model has the form,

$$\log\left(\frac{q}{1-q}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_j x_j + \beta_m x_m \tag{3.8}$$

where, log means the natural logarithm, β_j is the regression coefficient associated with independent variable x_j, and m is the number of independent variables.

Given a list of candidate independent variables and a significance level, α, some of the estimated coefficients may not be significantly different from zero. Such variables should not be included in the final model.

3.6.4 Lines-Of-Code

Lines-of-code is one of the most important measures that can represent the complexity of software program modules. The classifier based on lines-of-code works on the hypothesis that the larger the number of lines of code for a software program module, the more complex the software program module, and the greater the chance that program module is fault-prone(fp). The procedure involved in the classifier based on lines of code is as follows: 1) sort the modules in the ascending order of the metric, LOC (lines-of-code). 2) For a given threshold value thd_{loc}, calculate the two misclassification error (Type I and Type II) rates. In our study, all the modules with LOC less than thd_{loc} are predicted as nfp, fp otherwise. 3) Empirically determine the final threshold value thd_{loc} that satisfies desired model selection strategy. We adopted the strategy of selecting a model with the most balanced rates of misclassification of both types.

3.6.5 Genetic Programming

Genetic Programming (GP) is a domain in the field of machine learning systems [59]. A unique advantage of GP is that a solution evolves automatically from the training data set. The evolution process in GP imitates the Darwinian principle

of survival and reproduction of the fittest individuals. Each individual in GP is an S-expression composed of functions and terminals provided by the problem. A fitness value of an individual (or model in our case) indicates its quality with respect to the problem domain. Hence, it gives a probability of who can be selected for mating and reproducing for the next generation. We direct the reader to [3] for the basics of GP.

Use of appropriate fitness functions is an important part of the algorithm. Weighted average cost of misclassification can be used as a fitness function. Since GP is a multi-objective optimization algorithm, tree size is often used as one of the fitness functions, but is not a primary one.

3.6.6 Artificial Neural Networks

Artificial neural networks (ANN) are systems that are deliberately constructed to make use of some organizational principles resembling those of the human brain. According to methods of learning rules, ANN can be classified mainly into two categories: supervised-learning networks and unsupervised-learning networks [66].

Backpropagation [92] is the most popular training algorithm for multilayer neural networks. The algorithm initializes the network with a random set of weights and basis, and the network trains from a set of input-output pairs. The *batch training* algorithm computes the weight update for each input sample, and stores these values (without changing the weights) during a pass through the training set (epoch). At

the end of each epoch, all the weight updates are added together, and only then will the weights be updated with the composite value. We confined our study to feedforward supervised-learning neural networks, in particular backpropagated [67, 73] neural networks.

3.6.7 Rule-Based Modeling

Rule-based modeling (RBM) was proposed by our research team [68] for use in the software quality classification domain. In the context of RBM, if m is the number of independent variables, there are 2^m rules. Each rule is a Boolean function consisting of one or more Boolean AND operators, the independent variables' values x_{ij}, and their critical values c_j. Based on its critical value, each independent variable (x_{ij}) can have two possible values, $x_{ij} \leq c_j$ and $x_{ij} > c_j$ (0 or 1). Consequently, each rule has a distinct index, representing one of the unique 2^m possible combinations. The instances in the training data set are assigned to the rules. The subsequent work is to classify the rules based on the pre-defined model selection strategy and finally determine the class of instances in the test data set.

3.6.8 Rough Sets

Based on the classical set theory, rough sets were introduced in 1982 by Pawlak [58]. Using the concept of equivalence relations, partitions of a set of objects can be formed, subsets of significant attributes identified, and decision rules extracted, all based on the attribute values of the objects. Rough set theory can

be used to analyze the dependency relationship between the independent variables and the dependent variable to determine whether the dependent attribute can be characterized by the values of the independent attributes.

In practice, it is often the case that not all the independent variables are equally significant indicator/predictor of the dependent variable (class label in the context of classification). Hence, selecting proper attributes for prediction/classification is imperative for a technique to be successful. Rough set theory can be used to identify subsets of attributes, called 'reducts', that have the same discrimination power as a complete set of attributes. Once the reducts are identified, a set of decision rules are obtained to perform classification. When the domain values of an attribute are continuous and relatively large, rough set theory requires that they be discretized.

3.6.9 Combining Classification Technique

Analogous to how people make decisions by consulting with a panel of experts, we can combine multiple classifiers (experts) to achieve a combined classification [113]. The combined classifier may significantly enhance the accuracy of the individual classifier. We will present *Bagging* [7], followed by *Boosting* [27], *LogitBoost* [29], and *MetaCost* [19]. Detailed information about the techniques may be found in the references above and in [51].

3.6.9.1 Bagging

The simplest way to combine classifiers is to randomly re-sample from the original training data set, build a classifier for each re-sampled dataset, and use the prediction of each classifier in a simple vote to obtain the combined decision on the test data. This technique is known as *Bagging*. The re-sampling is performed with replacement: an instance may be re-sampled more than once, others may not be re-sampled. The re-sampled data sets usually have the same size as the original training data set. The combined decision or *final hypothesis* for classification (a class) is obtained using an unweighed vote.

The classifier used for the combined decision is referred to as the weak learner. One requirement for the weak learner is to be unstable [7, 113]. The instability of the weak learner ensures that small changes in the training data will yield significantly different models. There is no point in combining very similar learners since they will provide very similar outcomes. Decision trees and neural networks are typical unstable learners [8]. One may also notice that all the learners are generated independently from each other. Thus, Bagging is an algorithm that is easy to implement on parallelized architectures [7]. Comprehensive description of the algorithm can be found in [34].

3.6.9.2 Boosting

Boosting [28] exploits the instability of weak learners in a way very similar to Bagging. However, while Bagging generates the classifiers independently of each other, the Boosting algorithm seeks classifiers that complement each other, rather than generating them randomly [113]. The main difference between Boosting and Bagging is that while Bagging was obtained by generating independent samples, Boosting is an iterative method (each model is generated based on previous results). The idea in Boosting is to favor classifiers that perform better on instances that were previously misclassified. Each classifier is thus influenced by the performance of the previous classifiers. In addition, while Bagging uses an unweighed vote to generate the final hypothesis, Boosting weighs each classifier's contribution in the combined decision based on its performance. Thus, the best classifiers are given more importance in the combined decision.

In Boosting, the weight updating is straightforward: the weight of correctly classified instances is decreased while the weight of misclassified instances is increased. The next classifier will then focus on the *hard* to classify correctly instances: those with high weights. Thus, each instance's weight holds the history of all the previous classification (correct, incorrect). The complete Boosting algorithm (AdaBoost) presented by Freund and Schapire [27] was used as one of the classifiers in this study.

3.6.9.3 LogitBoost

LogitBoost is a re-derivation of AdaBoost as a method for fitting an additive model in a forward stagewise manner [29]. The idea here is to fit an additive model by minimizing the squared error loss in a forward stagewise manner. In LogitBoost, Boosting is viewed as an approximation of additive modeling on a logistic scale using the maximum Bernoulli likelihood as a criterion [29]. The result is an additive logistic model composed of functions that represent the weak hypotheses.

The additive symmetric logistic model is fitted in a forward stagewise manner using Newton steps [29]. Variables are included sequentially in a stepwise regression, and the coefficients of variables already included in the model remain constant. The likelihood values are based on estimates of class probabilities. A model is fitted with observation weights to produce a new weak hypothesis. The complete description of LogitBoost is given in [29].

3.6.9.4 MetaCost

MetaCost is a cost-sensitive meta learning method. This method treats the underlying classifier as a black box, requiring no knowledge of its functioning or change to it. MetaCost is based on wrapping a "meta-learning" stage around the error-based classifier in such a way that the classifier effectively minimizes cost while seeking to minimize error rates.

MetaCost uses a variant of Breiman's [7] bagging as the ensemble method.

MetaCost differs from bagging in that the number of examples (modules/instances) in each resample may be smaller than the training size (in the bagging procedure, the number of the examples in each resample is the same as that of the original training data set). This allows it to be more efficient. If the classifier being used produces class probabilities, a class' vote is estimated as the unweighted average of its probabilities, given the models and the example. Also, when estimating class probability for a given training example $\mathbf{x}$, MetaCost always takes all the models generated into consideration or only those that were learned on resamples the example was not included in. The first type of estimate is likely to have lower variance, because it is based on a large number of samples, while the second is likely to have lower statistical bias, because it is not influenced by the example's own class in the training set.

The working procedure of MetaCost can be simply expressed as follows: 1) yielding multiple bootstrap replicates of the training data set, and learning a classifier on each replicate; 2) estimating each class' probability for each example by the fraction of votes that it receives from the ensemble; 3) using the expected cost of misclassification to relabel each training example with the estimated optimal class; 4) reapplying the classifier to the relabelled training data set. Readers may refer to [19] for more details on the MetaCost algorithm.

3.6.10 Decision Table

Decision Table is one the simplest methods for learning from input data. The concepts/rules learnt from the input data have the same form as input - the form of a decision table, a list of rules in a table format. The problem of constructing a decision table involves selection of appropriate attributes for inclusion, and getting rid of irrelevant attributes. When determining a class for a test instance, all one has to do is to look up the appropriate conditions in the list of the rules - the decision table.

Because they permit one to display succinctly the conditions that must be satisfied before prescribed actions are to be performed, decision tables are becoming popular in computer programming and system design as devices for organizing logic [88].

Decision tables are attractive because of their simplicity, and because they are very easy to understand, when concise in size. Decision tables are also appealing in real-time environment, since they provide a constant classification time on average [55].

3.6.11 Alternating Decision Tree

Alternating Decision Trees (ADTrees) resemble the *option trees* described by Buntine in [11] and further developed by Kohavi et al. in [56]. ADTree algorithm, a

relatively new machine learning technique proposed by Freund and Mason [26], combines the power of boosting and decision trees in a very simple manner generalizing decision trees, voted decision trees, and voted decision stumps. Since, ADTree has alternating layers of decision nodes and prediction nodes in its tree structure, it is called *Alternating Decision Tree*. Freund and Mason [26] have shown that ADTree is able to achieve classification accuracy comparable to boosted decision tree algorithms or algorithms that combine boosting in their implementation, keeping the generated rules much smaller and easily interpretable.

3.6.12 SMO

Sequential Minimal Optimization (SMO), proposed by Platt [81], is a conceptually simple, but subtle algorithm for training support vector machines, which involves solving a very large quadratic programming (QP) optimization problem, using Osuna's theorem to ensure convergence. The problem is resolved by divide and conquer approach in that the large QP problem is divided into smaller pieces of QP problems (*subproblems*)which are then solved analytically in steps instead of the traditional time-consuming way of numerical QP optimization. SMO, with its novel approach of QP optimization, reduces time complexity dramatically, and can be, in many cases, more than 1000 times faster than its traditional counterpart, the PCG (Projected Conjugate Gradient) Chunking algorithm, reports Platt [82]. SMO holds its appeal also because of its scalability. It is capable to handle large training

dataset, since the memory requirements of SMO is linearly dependent on the size of training dataset.

3.6.13 IB1

Instance-based learning is a popular classification scheme. The working hypothesis of the technique is that the instance under examination (test case) would belong to the same class as that of other similar instances. Different instance-based learning algorithms vary in the context of the selected number of nearest neighbors, measures used to compute similarity between instances, and the solution algorithm for predicting the class of a test instance, etc.

IB1, WEKA's implementation of 1 instance-based classifier, uses only one nearest neighbor to predict the class of a test instance. The similarity measure used is Euclidean distance. Despite its simplicity, it can achieve reasonable classification accuracy. Cover and Hart [16] demonstrated that 1NN (1-instance- based) classifier performs as well as Bayesian classifiers do.

3.6.14 IBk

IBk is WEKA's implementation of an instance-based learning technique with k nearest neighbors. Selecting only one nearest neighbor to predict the class of a test instance, especially in the presence of noise, may lead to increased inaccuracy [113]. Selecting more than one nearest neighbor is a much more realistic approach that is less easily influenced by noisy exemplars. In IBk, the class of the test case

is predicted by majority voting of the k nearest neighbors. Like IB1, the similarity measure used to determine the nearest neighbors is Euclidean distance. But unlike IB1, when using IBk, the attributes need to be normalized.

As the number of nearest neighbors increases, the associated computation time, of course, increases. The computation time increases linearly also with the increase in the training set size. Indeed, IBk can be computationally expensive at times. But, incremental training and testing could be employed to overcome this issue.

3.6.15 PART

PART is a simple, yet surprisingly effective, method for learning decision lists based on the repeated generation of partial decision trees in a separate-and-conquer manner [25]. PART, unlike the two dominant practical implementations of rule learners, C4.5 [85] and Ripper [13], avoids the time consuming phase of postprocessing for global optimization.

PART employs the *separate-and-conquer* strategy in that it builds a rule, removes the instances covered by the rule, and continues creating rules recursively for the remaining instances until none are left[25]. It may sound silly to repeatedly build a decision tree just to create one rule and then to discard it. But the process has certain advantages which should not be overlooked. Besides its simplicity, PART

offers protection against overpruning because of the way it combines the *separate-and-conquer* approach with the decision trees, adding flexibility and speed to the process. Despite its simplicity, PART produces rule sets that are at least as accurate as those generated by its counterparts, i.e. C4.5 and RIPPER. [25].

3.6.16 OneR

OneR algorithm, introduced by Holte [40], is one of the simplest algorithms available in machine learning. Despite its simplicity, it compares favorably to many of the *state-of-the-art* machine learning techniques. It chooses the most informative single attribute, and bases the rule on this attribute alone. In practice, simple rules often achieve surprisingly high accuracy, which could be attributed to the rudimentary underlying structure of many real-world datasets.

The fundamental concept of the algorithm is succinctly presented in [72]:

For each attribute a, a rule is generated as follows: For each value v from the domain of a, select the set of instances where a has value v. Let c be the most frequent class in that set. Add the following clause to the rule for a: *if a has value v then the class is c.* Calculate the classification accuracy of this rule. The rule with the highest classification accuracy is used.

The algorithm requires attributes to be discrete, and can handle missing values.

3.6.17 JRip

JRip is WEKA's implementation of the rule-based learning algorithm, RIPPER (Repeated Incremental Pruning to Produce Error Reduction) - a modification of the IREP (Incremental Reduced Error Pruning). RIPPER was proposed by Cohen [13], and was shown to compare favorably with C4.5. Both RIPPER and C4.5 rules start with an initial model and iteratively improve it using heuristic techniques. However, for large noisy datasets, the former generally seems to start with an initial model that is about the right size, while the latter starts with an extremely large initial model. This means that RIPPER is more search-efficient.

Based on the empirical work done with IREP, Cohen [13] proposed RIPPER, suggesting three modifications to the IREP algorithm. In order to overcome the occasional failure of IREP to converge, a different and more intuitive metric to evaluate the rules during pruning phase was introduced. In view of IREP's undue sensitivity to the "small disjunct problems" [39], total description length criterion was proposed to replace the error rate as a stopping criterion while building the ruleset. Finally, a postpass that *optimizes* a rule set in an attempt to more closely approximate conventional reduced error pruning was introduced.

3.6.18 Ridor

The ripple-down rule(RIDOR) technique was introduced by Compton and Jansen as a methodology for acquisition and maintenance of large rule-based systems [14, 15].

The basic idea behind the technique is to make incremental changes while constructing and maintaining a complex knowledge structure in a well-defined and restricted manner such that the effects of the changes do not propagate globally, and are well confined in the structure, unlike standard production rules. In RIDOR, rule activation is considered only in the context of other rule activation [30]. The rules thus formed can be viewed as a binary decision tree with a compound clause at each decision node. Unlike a standard decision tree, the rule at each decision node does not necessarily cover all the instances - a decision can be reached at an internal node. However, similar to a standard decision tree, only one decision node is activated for an instance, which makes maintaining the ruleset easier [30].

3.6.19 J48

J48 is the implementation of C4.5, the landmark Decision Tree algorithm introduced by Quinlan [85]. The C4.5 algorithm is an inductive supervised learning system which employs decision trees to represent the underlying structure of the input data. The algorithm is comprised of four principal components: decision tree generator, production rule generator, decision tree interpreter, and production rule

interpreter, for constructing and evaluating the classification tree models.

The C4.5 algorithm requires certain pre-processing of data in order for it to build decision tree models. Some of these include attribute value description type, predefined discrete classes, and sufficient number of observations for supervised learning.

The classification tree is initially empty, and the algorithm begins adding decision and leaf nodes, starting with the root node. At each decision node, the decision is based on only one attribute, which makes the tree easier to understand. C4.5 algorithm, much like its counterpart CART, builds a complete tree, and then prunes it to avoid overfitting, which may seem wasteful. But the act of fully exploring the decision tree and then pruning it is worth the extra computation effort for the improved accuracy.

3.6.20 NaiveBayes

Naive Bayes is one of the most simplistic techniques available for classification. This simple and intuitive method, based on Bayesian rule of conditional probability, "naively" assumes that attributes are independent of each other given the class, which may not be completely true in the real world.

Despite the over-simplification of the actual relationship between the attributes, Naive Bayes has been shown to perform fairly well, especially when used along with feature selection techniques that remove irrelevant attributes [24]. It is

important to discard irrelevant and redundant attributes to ensure that the algorithm performs to its capability. It can handle missing values without any problem.

3.6.21 Hyperpipes

Hyperpipes also belongs to the category of the simplest classification techniques. As described in [80], for each class label, a hyperpipe that would contain all the instances having the same class label is constructed. For each hyperpipe, the attribute bounds are observed and recorded from the instances it contains. When classifying a test instance, the class label associated with the hyperpipe that most contains the test instance is assigned.

Hyperpipes is an extremely simple algorithm, but has the advantage of being extremely fast, and works quite well in presence of many attributes. It cannot handle missing values in the test cases.

3.6.22 LWLStump

Locally Weighted Learning (LWL) is a non-adaptive lazy-learning technique that is gaining popularity in the machine learning community. Atkeson et al. have surveyed locally weighted learning in [2]. Local weighting reduces unnecessary bias of global function fitting, and gives more flexibility, retaining the desirable properties such as smoothness, and statistical analyzability [2]. LWL uses locally-weighted training to combine training data, using a distance function to fit a surface to

nearby points. It must be used in conjunction with another classifier to perform classification, e.g. Decision Stump in the case of LWLStump.

There are mainly three different requirements for Locally Weighted Learning [2]: Distance function, Separable criterion, and Sufficient data. There is a need for a measure of relevance for learning with Local weighting. The more commonly used measure of relevance is the distance metric. Additive separability criterion is typically used so that the training criterion is not a general function of the predictions of the training instances. LWL also calls for sufficiency of the data, which, of course, is subjective, and depends on the nature of the problem domain.

Chapter 4

EXPERIMENTS

The focus of this chapter is to describe the experiments conducted and the results obtained during our empirical investigation.

4.1 System Description

For the case studies reported in this study, we used data from two NASA projects, namely JM1 and KC2, written in C++, which are available from the Metrics Data Program(MDP) website [4]. This website provides access to the data repository containing software metrics and associated error data at the function/method level. The data repository stores and organizes the data which has been collected and validated by the Metrics Data Program. This data has been made available through this website with the approval of the project(s) which have worked in cooperation with the Metrics Data Program.

The two data sets are denoted by JM1 and KC2, the former being the larger (10,883 modules) and the latter being the smaller(520 modules) of the two. Even

[4] http://mdp.ivv.nasa.gov

though KC2 actually contained more than 3,000 modules, for our study, we considered only 520 modules - the modules which were developed by NASA software developers, and were not COTS software.

Of the 10,883 modules in JM1 dataset, 2,105 modules had errors, ranging from 1 to 26. The remaining 8,778 modules were error-free. After removing inconsistent instances (the instances with identical independent variables, but with different class labels) and the instances with missing values, size of the JM1 dataset reduced from 10,883 instances to 8,850 instances. Out of the 8,850 modules in the JM1 dataset, the dataset denoted by *JM1-8850* in Table 4.1, 1,687 modules had one or more defects, whereas the remaining 7,163 modules did not have any defects.

While it was observed that there were some inconsistent instances among the 520 instances in the KC2 dataset denoted by *KC2-520* in Table 4.1, we decided not to toss them out, for the dataset size was rather small, and we wanted to explore whether these inconsistent examples get filtered out by our noise elimination approach, which was later confirmed to be true. It was found that all the inconsistent examples for the KC2 system were misclassified by all the twenty five classification techniques, and hence were filtered out by the noise elimination process. Of the 520 modules in the KC2-520 dataset, 106 had errors ranging from 1 to 13; while the remaining 414 were error-free.

Table 4.1: Dataset Details for JM1 and KC2 Systems

	nfp modules		*fp* modules		Total
Dataset	Count	Proportion	Count	Proportion	Count
JM1-8850	7163	80.94%	1687	19.06%	8850
JM1-4425-Fit	3581	80.93%	844	19.07%	4425
JM1-4425-Test	3582	80.95%	843	19.05%	4425
JM1-23C-Fit	3143	80.67%	753	19.33%	3896
JM1-23C-Test	3143	80.69%	752	19.31%	3895
JM1-20C-Fit	2862	80.44%	696	19.56%	3558
JM1-20C-Test	2861	80.43%	696	19.57%	3557
JM1-17C-Fit	2670	80.52%	646	19.48%	3316
JM1-17C-Test	2670	80.52%	646	19.48%	3316
JM1-13C-Fit	2431	80.84%	576	19.16%	3007
JM1-13C-Test	2430	80.84%	576	19.16%	3006
KC2-520	414	79.62%	106	20.38%	520
KC2-260-Fit	207	79.62%	53	20.38%	260
KC2-260-Test	207	79.62%	53	20.38%	260
KC2-23C-Fit	181	79.04%	48	20.96%	229
KC2-23C-Test	182	79.48%	47	20.52%	229
KC2-17C-Fit	173	79.72%	44	20.28%	217
KC2-17C-Test	172	79.63%	44	20.37%	216
KC2-13C-Fit	166	79.43%	43	20.57%	209
KC2-13C-Test	166	79.43%	43	20.57%	209

4.2 Noise Elimination

We performed noise elimination using the proposed ensemble-classifier approach for both the software systems (JM1 and KC2). The filtering was based on the performance of twenty five different classification techniques on the JM1-8850 and KC2-520 datasets for the JM1 and KC2 systems respectively. For most classification techniques, the predictions on which the filtering was based were obtained using 10-fold cross-validation, with a few exceptions as mentioned in the Section 3.4.2.

Experimenting with as many as twenty five classifiers enabled us to explore several levels of filtering. For the JM1 system, we decided to have four different levels of filtering denoted by 13C, 17C, 20C, and 23C, with 13C being the least conservative and 23C being the most conservative amongst the four. Noise filtering at 13C level, a noise filtering level where all the instances misclassified by 13 or more classification techniques have been eliminated, is analogous to majority filtering since we are using twenty five classification techniques.

We strived to follow the same filtering approach for the KC2 system. However, since the number of modules to be eliminated at the 20C level was not significantly less than that at the 23C level, we decided to skip the 20C level for the KC2 system. Hence, in the case of the KC2 system, the noise elimination was performed at three different levels: 13C, 17C, and 23C. The notations, however, hold the same meaning for the KC2 system as well.

We did not perform consensus filtering (25C in our case), for it appeared to be too stringent a criterion for noise elimination with twenty five classification techniques.

Having performed the noise elimination, each dataset was proportionately split into two halves: fit and test sets. The notations used for each dataset and distribution of *fault-prone* and *not fault-prone* modules in each dataset is summarized in Table 4.1. Most of the notations are self explanatory, with each one having prefix for the respective software system. For example, JM1-8850 stands for the

original dataset (with 8850 modules) used for noise elimination; JM1-4425-Fit and JM1-4425-Test are the training and the evaluation dataset splits respectively generated before noise elimination; and JM1-23C-Fit and JM1-23C-Test stand for the fit and test dataset splits respectively generated after noise elimination at the 23C level. The same also follows for the KC2 system.

It was surprising to note that the distribution of *fp* and *nfp* modules in the datasets remains almost the same after the noise elimination process at different levels of filtering, for both the systems (JM1 and KC2).

Table 4.2 displays the quality-of-fit in terms of misclassification error statistics for all the twenty five classification techniques on the datasets (JM1 – 8850 and KC2 – 520) used for noise elimination for the software systems JM1 and KC2 respectively. The relatively higher magnitude of the misclassification errors indicate that the datasets are likely to be noisy, which is confirmed by the improvement in classification accuracy with increasing level of noise filtering (going from the most conservative level-23C to the least conservative level-13C). One obvious observation from Table 4.2 is that while the misclassification errors are still high for the KC2 dataset, they are not as high as those for the JM1 dataset, which may be the indication that relatively less amount of noise is present in the KC2 dataset than in the JM1 dataset.

Table 4.2: Quality-of-Fit Results for JM1-8850 and KC2-520 Datasets

	JM1-8850		KC2-520	
Methods	Type I	Type II	Type I	Type II
CBR	30.70%	30.88%	21.50%	20.75%
TD	30.78%	29.16%	18.60%	16.04%
LR	34.23%	33.97%	20.77%	21.70%
LOC	34.85%	34.08%	20.53%	19.81%
GP	34.71%	32.66%	18.36%	16.98%
ANN	38.06%	30.35%	21.26%	21.70%
LBOOST	34.72%	32.72%	22.22%	20.75%
RBM	33.71%	33.08%	17.39%	16.04%
BAG	30.59%	30.76%	21.50%	20.75%
RSET	31.62%	30.94%	16.18%	14.15%
MCOST	33.67%	33.61%	23.43%	21.70%
ABOOST	33.41%	33.79%	28.26%	29.25%
DTABLE	34.29%	34.32%	18.84%	18.87%
ADT	33.83%	33.61%	19.81%	19.81%
SMO	34.09%	33.97%	20.77%	20.75%
IB1	34.73%	34.74%	23.67%	24.53%
IBK	32.70%	32.48%	20.53%	19.81%
PART	33.16%	33.14%	20.77%	19.81%
ONER	34.50%	34.38%	20.05%	19.81%
JRIP	33.18%	33.08%	19.81%	19.81%
RDR	33.94%	34.02%	18.84%	19.81%
J48	32.56%	32.42%	19.57%	19.81%
NBAYES	34.12%	33.97%	21.26%	21.70%
HPIPES	37.97%	38.29%	23.91%	23.58%
LWLS	33.59%	33.61%	20.05%	19.81%
Average	33.75%	33.12%	20.71%	20.30%
Std. Dev	1.80%	1.80%	2.41%	2.93%
Median	33.83%	33.61%	20.53%	19.81%
Min	30.59%	29.16%	16.18%	14.15%
Max	38.06%	38.29%	28.26%	29.25%

4.3 Classification Results

The performance of all the classification techniques on the datasets with different levels of noise is presented in terms of the Type I and Type II errors for both the software systems, in the following subsections.

4.3.1 Misclassification Summary for the JM1 System

Presented in this subsection are the classification performance (both the quality-of-fit and the predictive performance) results in terms of Type I and Type II errors for the datasets of the JM1 system, with different levels of noise.

Table 4.3 displays the misclassification rates for all the classifiers on the JM1-4425 datasets (Fit and Test). JM1-4425-Fit dataset was used to build the models, and the selected model for each classification technique was applied to the JM1-4425-Test dataset for evaluation. The JM1-4425-Fit and JM1-4425-Test datasets are the datasets generated by impartially splitting the original JM1-8850 dataset without any noise removal. The fact that these are not noise-free datasets is reflected in the higher values of misclassification error rates.

In the Tables 4.4, 4.5, 4.6, and 4.7, classification performance of all the classification techniques is displayed for datasets with all the four different levels of noise filtering (23C, 20C, 17C, and 13C) for the JM1 system. It is evident from these results that as more and more noise is removed from the dataset, there is improvement in both the quality of fit and the predictive performance of all the classifiers.

Table 4.3: Classification Accuracy Results for JM1-4425 Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	33.01%	32.23%	32.52%	31.91%
TD	31.50%	30.21%	31.85%	36.54%
LR	33.96%	34.12%	35.34%	33.10%
LOC	34.54%	33.89%	35.15%	34.28%
GP	34.35%	32.46%	35.26%	32.62%
ANN	38.82%	29.86%	37.74%	29.77%
LBOOST	35.80%	33.77%	35.29%	32.15%
RBM	32.62%	32.94%	33.92%	36.30%
BAG	34.13%	31.75%	33.47%	29.54%
RSET	32.73%	31.28%	33.98%	34.28%
MCOST	36.92%	34.00%	41.09%	22.89%
ABOOST	34.82%	34.72%	37.47%	31.32%
DTABLE	34.15%	34.12%	34.42%	36.30%
ADT	34.60%	34.36%	26.69%	41.76%
SMO	33.65%	33.77%	34.28%	34.52%
IB1	37.84%	37.68%	38.55%	33.69%
IBK	33.54%	33.65%	33.56%	31.91%
PART	34.79%	34.72%	49.78%	22.78%
ONER	34.93%	35.07%	37.55%	35.94%
JRIP	34.04%	34.24%	38.92%	30.25%
RDR	34.74%	34.60%	42.66%	26.45%
J48	33.98%	34.00%	26.47%	43.18%
NBAYES	33.20%	33.53%	34.06%	34.28%
HPIPES	38.59%	39.10%	36.91%	37.01%
LWLS	34.35%	34.60%	34.31%	30.37%
Average	34.62%	33.79%	35.65%	32.93%
Std. Dev	1.78%	1.96%	4.67%	4.74%
Median	34.35%	34.00%	35.15%	33.10%
Min	31.50%	29.86%	26.47%	22.78%
Max	38.82%	39.10%	49.78%	43.18%

Table 4.4: Classification Accuracy Results for JM1-23C Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	23.80%	23.11%	22.62%	23.14%
TD	23.13%	18.19%	25.17%	26.60%
LR	24.98%	24.97%	24.94%	25.80%
LOC	25.99%	25.23%	25.52%	26.99%
GP	24.75%	23.11%	24.21%	27.79%
ANN	25.36%	23.90%	25.14%	26.60%
LBOOST	25.93%	24.83%	29.62%	24.07%
RBM	25.04%	23.51%	25.58%	26.73%
BAG	23.35%	21.91%	21.99%	20.61%
RSET	22.14%	21.38%	23.74%	28.46%
MCOST	24.69%	25.76%	23.45%	26.20%
ABOOST	26.19%	25.90%	23.93%	23.54%
DTABLE	27.08%	26.29%	25.42%	28.19%
ADT	25.17%	25.23%	26.66%	25.66%
SMO	24.59%	24.70%	24.72%	27.39%
IB1	26.22%	25.23%	27.04%	24.73%
IBK	24.94%	24.97%	24.09%	26.86%
PART	23.96%	23.24%	21.64%	27.66%
ONER	25.04%	25.90%	29.18%	23.40%
JRIP	24.98%	24.44%	26.41%	25.93%
RDR	25.14%	25.23%	24.63%	27.53%
J48	24.72%	24.70%	24.75%	20.88%
NBAYES	25.01%	25.10%	25.99%	27.26%
HPIPES	28.92%	28.82%	26.73%	27.66%
LWLS	24.69%	24.70%	24.47%	25.40%
Average	25.03%	24.41%	25.10%	25.80%
Std. Dev	1.32%	1.97%	1.89%	2.14%
Median	24.98%	24.83%	24.94%	26.60%
Min	22.14%	18.19%	21.64%	20.61%
Max	28.92%	28.82%	29.62%	28.46%

Table 4.5: Classification Accuracy Results for JM1-20C Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	18.03%	18.25%	17.23%	18.53%
TD	15.55%	12.36%	17.02%	18.97%
LR	18.41%	18.25%	19.12%	18.53%
LOC	20.16%	19.25%	20.31%	18.25%
GP	17.30%	18.25%	18.04%	20.69%
ANN	16.35%	19.83%	16.67%	19.97%
LBOOST	19.60%	18.25%	18.42%	18.39%
RBM	17.44%	16.38%	18.59%	20.55%
BAG	16.98%	16.38%	16.64%	14.94%
RSET	14.88%	15.66%	19.12%	19.54%
MCOST	18.80%	17.39%	20.03%	15.52%
ABOOST	16.70%	15.95%	16.46%	16.67%
DTABLE	18.48%	18.53%	23.66%	15.09%
ADT	17.99%	18.10%	16.22%	20.55%
SMO	18.59%	18.39%	18.70%	19.25%
IB1	19.01%	18.82%	18.63%	19.25%
IBK	17.09%	17.10%	16.53%	18.97%
PART	16.91%	16.95%	16.04%	18.53%
ONER	19.74%	19.54%	22.44%	16.67%
JRIP	17.68%	17.67%	14.47%	21.84%
RDR	17.65%	18.53%	20.10%	16.95%
J48	17.12%	17.10%	16.71%	19.83%
NBAYES	19.43%	19.54%	19.47%	20.26%
HPIPES	21.52%	20.26%	21.01%	20.83%
LWLS	19.32%	16.38%	19.61%	16.95%
Average	18.03%	17.72%	18.45%	18.62%
Std. Dev	1.50%	1.67%	2.13%	1.87%
Median	17.99%	18.25%	18.59%	18.97%
Min	14.88%	12.36%	14.47%	14.94%
Max	21.52%	20.26%	23.66%	21.84%

Table 4.6: Classification Accuracy Results for JM1-17C Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	13.67%	13.00%	13.67%	10.53%
TD	11.31%	8.51%	13.82%	13.31%
LR	12.62%	12.54%	13.07%	13.78%
LOC	14.42%	13.16%	15.32%	14.09%
GP	12.02%	12.07%	12.02%	14.09%
ANN	13.03%	11.92%	13.22%	14.09%
LBOOST	14.04%	13.47%	14.04%	13.47%
RBM	13.07%	11.76%	13.45%	14.09%
BAG	12.43%	11.30%	12.21%	8.20%
RSET	10.37%	10.22%	13.71%	13.47%
MCOST	11.80%	12.85%	13.60%	12.38%
ABOOST	12.47%	12.07%	11.65%	9.60%
DTABLE	13.37%	13.00%	14.27%	10.53%
ADT	10.90%	10.84%	12.62%	11.15%
SMO	12.81%	12.69%	13.07%	14.24%
IB1	13.48%	13.62%	15.24%	10.84%
IBK	11.42%	11.30%	12.13%	10.99%
PART	11.84%	11.92%	11.54%	10.06%
ONER	13.00%	12.69%	15.06%	13.93%
JRIP	12.36%	12.23%	13.71%	10.22%
RDR	12.32%	11.92%	10.67%	14.24%
J48	12.47%	12.38%	13.41%	11.30%
NBAYES	13.97%	13.78%	13.60%	14.71%
HPIPES	14.61%	15.02%	22.55%	13.93%
LWLS	11.65%	11.76%	13.30%	11.15%
Average	12.62%	12.24%	13.64%	12.33%
Std. Dev	1.08%	1.27%	2.17%	1.88%
Median	12.47%	12.23%	13.45%	13.31%
Min	10.37%	8.51%	10.67%	8.20%
Max	14.61%	15.02%	22.55%	14.71%

Table 4.7: Classification Accuracy Results for JM1-13C Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	8.02%	7.29%	6.23%	10.07%
TD	5.02%	4.17%	4.94%	6.42%
LR	5.43%	5.73%	4.36%	6.94%
LOC	6.75%	7.12%	5.72%	8.16%
GP	5.72%	5.90%	5.43%	7.29%
ANN	6.38%	6.08%	4.94%	7.29%
LBOOST	5.88%	6.08%	5.84%	6.08%
RBM	5.88%	5.21%	6.63%	7.29%
BAG	5.18%	5.21%	4.90%	5.03%
RSET	4.44%	4.17%	7.41%	6.08%
MCOST	5.96%	5.38%	4.81%	5.38%
ABOOST	3.95%	5.90%	3.58%	5.21%
DTABLE	6.21%	6.25%	7.41%	6.60%
ADT	4.73%	5.03%	5.10%	3.47%
SMO	6.21%	6.08%	5.93%	7.29%
IB1	6.33%	6.25%	6.54%	7.99%
IBK	5.02%	5.38%	4.73%	6.60%
PART	5.31%	5.03%	5.60%	5.73%
ONER	6.42%	6.25%	8.27%	6.94%
JRIP	5.72%	5.73%	7.00%	4.51%
RDR	6.42%	6.77%	8.15%	5.03%
J48	5.02%	5.21%	4.81%	5.56%
NBAYES	7.53%	7.29%	6.91%	7.99%
HPIPES	8.64%	8.16%	6.58%	13.72%
LWLS	5.10%	5.03%	5.64%	5.90%
Average	5.89%	5.87%	5.90%	6.74%
Std. Dev	1.08%	0.96%	1.20%	2.00%
Median	5.88%	5.90%	5.72%	6.60%
Min	3.95%	4.17%	3.58%	3.47%
Max	8.64%	8.16%	8.27%	13.72%

4.3.2 Misclassification Summary for the KC2 System

Presented in this subsection are the classification performance (both the quality-of-fit and the predictive performance) results in terms of Type I and Type II errors for the datasets of the KC2 system, with different levels of noise.

Table 4.8 displays the misclassification rates for all the classifiers on the KC2-260 datasets (Fit and Test). The KC2-260-Fit dataset was used to build the models, and the selected model for each classification technique was applied to the KC2-260-Test dataset for evaluation. The KC2-260-Fit and KC2-260-Test datasets are the datasets generated by impartially splitting the original KC2-560 dataset without any noise removal. Again, the fact that these are not noise-free datasets is reflected in the higher values of misclassification error rates.

In the Tables 4.9, 4.10, and 4.11, classification performance of all the classification techniques is displayed for datasets with all the three different levels (23C, 17C, and 13C) of noise filtering for the KC2 system. It is evident from these results that as more and more noise is removed from the dataset, there is improvement in both the quality of fit and the predictive performance of all the classifiers.

4.4 ECM Results

As explained in the Section 3.4.3, comparing as many as twenty five different classification methods based on the two types of misclassification rates can very well turn out to be a difficult and complex task, especially in the presence of many

Table 4.8: Classification Accuracy Results for KC2-260 Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	19.81%	20.75%	22.71%	16.98%
TD	19.81%	20.75%	20.77%	24.53%
LR	21.74%	20.75%	20.29%	16.98%
LOC	21.26%	18.87%	20.29%	18.87%
GP	16.91%	9.43%	20.77%	26.42%
ANN	23.67%	15.09%	27.05%	18.87%
LBOOST	24.64%	24.53%	28.50%	20.75%
RBM	18.84%	18.87%	20.77%	20.75%
BAG	25.60%	24.53%	22.71%	13.21%
RSET	18.84%	18.87%	17.87%	24.53%
MCOST	26.57%	26.42%	20.29%	24.53%
ABOOST	20.77%	24.53%	28.50%	26.42%
DTABLE	21.74%	18.87%	21.26%	20.75%
ADT	20.77%	22.64%	17.39%	32.08%
SMO	19.81%	20.75%	21.74%	22.64%
IB1	21.26%	20.75%	25.12%	37.74%
IBK	21.74%	18.87%	22.22%	20.75%
PART	23.67%	22.64%	22.22%	28.30%
ONER	20.77%	20.75%	20.77%	20.75%
JRIP	23.19%	22.64%	25.60%	16.98%
RDR	20.29%	22.64%	28.02%	15.09%
J48	23.67%	22.64%	31.40%	18.87%
NBAYES	21.74%	20.75%	20.77%	20.75%
HPIPES	21.74%	22.64%	25.12%	18.87%
LWLS	20.29%	20.75%	21.26%	26.42%
Average	21.57%	20.83%	22.94%	22.11%
Std. Dev	2.22%	3.38%	3.55%	5.46%
Median	21.26%	20.75%	21.74%	20.75%
Min	16.91%	9.43%	17.39%	13.21%
Max	26.57%	26.42%	31.40%	37.74%

Table 4.9: Classification Accuracy Results for KC2-23C Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	12.15%	12.50%	9.89%	10.64%
TD	12.71%	8.33%	13.74%	6.38%
LR	8.84%	10.42%	5.49%	14.89%
LOC	11.60%	10.42%	7.14%	10.64%
GP	4.42%	10.42%	4.40%	17.02%
ANN	9.94%	10.42%	7.69%	17.02%
LBOOST	10.50%	12.50%	6.04%	17.02%
RBM	8.84%	6.25%	10.44%	12.77%
BAG	10.50%	10.42%	6.59%	17.02%
RSET	10.50%	10.42%	6.59%	8.51%
MCOST	12.15%	14.58%	11.54%	8.51%
ABOOST	13.81%	10.42%	7.14%	10.64%
DTABLE	10.50%	10.42%	5.49%	17.02%
ADT	9.39%	8.33%	6.04%	8.51%
SMO	10.50%	10.42%	8.24%	12.77%
IB1	15.47%	6.25%	14.84%	12.77%
IBK	9.39%	10.42%	5.49%	12.77%
PART	11.60%	10.42%	13.74%	21.28%
ONER	10.50%	10.42%	7.14%	14.89%
JRIP	11.05%	8.33%	7.14%	14.89%
RDR	11.05%	8.33%	6.59%	17.02%
J48	11.60%	10.42%	13.74%	21.28%
NBAYES	11.60%	8.33%	8.24%	17.02%
HPIPES	10.50%	10.42%	9.34%	12.77%
LWLS	11.05%	10.42%	4.95%	17.02%
Average	10.81%	10.00%	8.31%	14.04%
Std. Dev	1.99%	1.80%	3.06%	3.93%
Median	10.50%	10.42%	7.14%	14.89%
Min	4.42%	6.25%	4.40%	6.38%
Max	15.47%	14.58%	14.84%	21.28%

Table 4.10: Classification Accuracy Results for KC2-17C Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	5.20%	4.55%	7.56%	2.27%
TD	3.47%	6.82%	2.91%	4.55%
LR	2.89%	2.27%	3.49%	2.27%
LOC	4.05%	4.55%	2.33%	6.82%
GP	0.58%	0.00%	1.74%	6.82%
ANN	12.14%	2.27%	12.21%	4.55%
LBOOST	4.62%	6.82%	5.23%	2.27%
RBM	4.62%	0.00%	5.81%	2.27%
BAG	6.36%	6.82%	5.23%	0.00%
RSET	4.05%	4.55%	2.33%	6.82%
MCOST	5.78%	6.82%	4.65%	6.82%
ABOOST	4.05%	6.82%	4.65%	4.55%
DTABLE	6.36%	4.55%	4.65%	4.55%
ADT	4.05%	4.55%	1.74%	4.55%
SMO	5.20%	6.82%	4.07%	4.55%
IB1	7.51%	9.09%	9.30%	6.82%
IBK	5.78%	6.82%	5.81%	4.55%
PART	5.20%	6.82%	3.49%	11.36%
ONER	4.05%	4.55%	2.91%	0.00%
JRIP	7.51%	6.82%	5.81%	4.55%
RDR	5.78%	6.82%	5.23%	0.00%
J48	5.78%	6.82%	6.40%	9.09%
NBAYES	6.36%	6.82%	5.23%	4.55%
HPIPES	3.47%	4.55%	3.49%	6.82%
LWLS	6.36%	6.82%	6.98%	0.00%
Average	5.25%	5.36%	4.93%	4.45%
Std. Dev	2.10%	2.26%	2.39%	2.90%
Median	5.20%	6.82%	4.65%	4.55%
Min	0.58%	0.00%	1.74%	0.00%
Max	12.14%	9.09%	12.21%	11.36%

Table 4.11: Classification Accuracy Results for KC2-13C Datasets

	Fit Set		Test Set	
Methods	Type I	Type II	Type I	Type II
CBR	1.81%	2.33%	5.42%	6.98%
TD	0.60%	0.00%	1.81%	2.33%
LR	5.42%	2.33%	7.83%	9.30%
LOC	3.61%	2.33%	2.41%	2.33%
GP	0.00%	0.00%	0.00%	6.98%
ANN	4.22%	0.00%	4.22%	6.98%
LBOOST	0.60%	0.00%	0.00%	4.65%
RBM	0.00%	0.00%	1.20%	4.65%
BAG	0.60%	0.00%	0.00%	4.65%
RSET	4.82%	4.65%	5.42%	2.33%
MCOST	0.60%	0.00%	0.00%	4.65%
ABOOST	0.60%	0.00%	0.00%	4.65%
DTABLE	0.60%	0.00%	0.00%	4.65%
ADT	0.60%	0.00%	0.00%	4.65%
SMO	2.41%	2.33%	3.01%	6.98%
IB1	3.61%	0.00%	4.82%	6.98%
IBK	3.61%	0.00%	6.63%	6.98%
PART	0.00%	0.00%	0.00%	4.65%
ONER	0.60%	0.00%	0.00%	4.65%
JRIP	0.60%	0.00%	0.00%	4.65%
RDR	0.60%	0.00%	0.00%	4.65%
J48	0.00%	0.00%	0.00%	4.65%
NBAYES	4.82%	2.33%	3.61%	2.33%
HPIPES	4.22%	2.33%	1.81%	4.65%
LWLS	0.00%	0.00%	0.00%	9.30%
Average	1.78%	0.74%	1.93%	5.21%
Std. Dev	1.87%	1.29%	2.48%	1.93%
Median	0.60%	0.00%	0.00%	4.65%
Min	0.00%	0.00%	0.00%	2.33%
Max	5.42%	4.65%	7.83%	9.30%

datasets (datasets with different level of noise). To make the task easier, we decided to evaluate the performance of classifiers in terms of the Normalized Expected Cost of Misclassification. NECM was computed from the misclassification error statistics at four different cost ratios: 10, 20, 30, and 50 for the reasons explained in the Section 3.4.3.

Presented in the following subsections are the classification performance results for all the classification techniques in terms of NECM computed for the training and evaluation datasets at different levels of noise for both the software systems.

4.4.1 ECM Results for the JM1 System

Tables 4.12, 4.13, 4.14, and 4.15 display the classification performance results of all the classification techniques on the datasets with different levels of noise, for the JM1 system. It is evident from these results that the value of expected cost of misclassification improves (goes down) as the level of noise elimination goes from the most conservative (23C) to the least conservative (13C).

Table 4.12: NECM Results for JM1 Dataset, c=10

	JM1-13C		JM1-17C		JM1-20C		JM1-23C	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Test
CBR	0.2045	0.2433	0.3633	0.3151	0.5020	0.5013	0.6386	0.6293
TD	0.1204	0.1630	0.2569	0.3706	0.3668	0.5080	0.5382	0.7166
LR	0.1536	0.1683	0.3459	0.3736	0.5051	0.5164	0.6840	0.6994
LOC	0.1909	0.2026	0.3724	0.3978	0.5388	0.5204	0.6974	0.7271
GP	0.1593	0.1836	0.3320	0.3712	0.4961	0.5499	0.6463	0.7320
ANN	0.1679	0.1796	0.3372	0.3809	0.5194	0.5249	0.6666	0.7163
LBOOST	0.1640	0.1637	0.3755	0.3755	0.5146	0.5080	0.6892	0.7037
RBM	0.1473	0.1933	0.3344	0.3827	0.4607	0.5516	0.6563	0.7225
BAG	0.1417	0.1361	0.3203	0.2581	0.4570	0.4262	0.6119	0.5754
RSET	0.1157	0.1763	0.2826	0.3727	0.4261	0.5361	0.5919	0.7409
MCOST	0.1513	0.1420	0.3453	0.3507	0.4913	0.4647	0.6971	0.6950
ABOOST	0.1450	0.1287	0.3356	0.2808	0.4463	0.4585	0.7118	0.6475
DTABLE	0.1699	0.1863	0.3610	0.3200	0.5112	0.4855	0.7266	0.7494
ADT	0.1347	0.1078	0.2989	0.3188	0.4989	0.5325	0.6907	0.7107
SMO	0.1666	0.1876	0.3504	0.3827	0.5093	0.5271	0.6758	0.7284
IB1	0.1709	0.2059	0.3739	0.3338	0.5211	0.5266	0.6992	0.6958
IBK	0.1437	0.1647	0.3121	0.3118	0.4719	0.5041	0.6838	0.7130
PART	0.1393	0.1550	0.3275	0.2889	0.4677	0.4917	0.6425	0.7086
ONER	0.1716	0.1999	0.3519	0.3926	0.5410	0.5066	0.7025	0.6873
JRIP	0.1560	0.1430	0.3378	0.3094	0.4879	0.5437	0.6738	0.7137
RDR	0.1816	0.1623	0.3314	0.3634	0.5045	0.4934	0.6905	0.7302
J48	0.1403	0.1454	0.3417	0.3281	0.4722	0.5224	0.6768	0.6028
NBAYES	0.2005	0.2089	0.3809	0.3960	0.5385	0.5530	0.6869	0.7361
HPIPES	0.2261	0.3160	0.4101	0.4530	0.5694	0.5766	0.7903	0.7497
LWLS	0.1377	0.1587	0.3230	0.3242	0.4758	0.4895	0.6766	0.6878
Average	0.1600	0.1769	0.3401	0.3501	0.4917	0.5127	0.6738	0.7008
Std. Dev	0.0262	0.0414	0.0323	0.0444	0.0418	0.0331	0.0474	0.0438
Median	0.1560	0.1683	0.3378	0.3634	0.4989	0.5164	0.6838	0.7130
Min	0.1157	0.1078	0.2569	0.2581	0.3668	0.4262	0.5382	0.5754
Max	0.2261	0.3160	0.4101	0.4530	0.5694	0.5766	0.7903	0.7497

Table 4.12: NECM Results for JM1 Dataset, c=10, contd...

	JM1-8850	JM1-4425	
Methods	Fit	Fit	Test
CBR	0.8372	0.8818	0.8712
TD	0.8051	0.8312	0.9539
LR	0.9245	0.9256	0.9166
LOC	0.9318	0.9259	0.9376
GP	0.9035	0.8972	0.9069
ANN	0.8866	0.8836	0.8728
LBOOST	0.9047	0.9338	0.8981
RBM	0.9034	0.8922	0.9661
BAG	0.8340	0.8818	0.8337
RSET	0.8458	0.8615	0.9281
MCOST	0.9132	0.9473	0.7688
ABOOST	0.9145	0.9440	0.8999
DTABLE	0.9318	0.9272	0.9702
ADT	0.9145	0.9354	1.0115
SMO	0.9234	0.9164	0.9351
IB1	0.9433	1.0249	0.9539
IBK	0.8838	0.9132	0.8795
PART	0.9000	0.9437	0.8368
ONER	0.9346	0.9516	0.9887
JRIP	0.8991	0.9286	0.8913
RDR	0.9233	0.9410	0.8493
J48	0.8816	0.9236	1.0368
NBAYES	0.9236	0.9082	0.9288
HPIPES	1.0373	1.0581	1.0038
LWLS	0.9125	0.9379	0.8563
Average	0.9045	0.9246	0.9158
Std. Dev	0.0445	0.0460	0.0630
Median	0.9125	0.9259	0.9166
Min	0.8051	0.8312	0.7688
Max	1.0373	1.0581	1.0368

Table 4.13: NECM Results for JM1 Dataset, c=20

	JM1-13C		JM1-17C		JM1-20C		JM1-23C	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Test
CBR	0.3442	0.4362	0.6166	0.5202	0.8589	0.8639	1.0852	1.0760
TD	0.2002	0.2861	0.4228	0.6300	0.6085	0.8791	0.8899	1.2300
LR	0.2634	0.3014	0.5902	0.6420	0.8620	0.8791	1.1666	1.1974
LOC	0.3272	0.3589	0.6288	0.6722	0.9154	0.8774	1.1851	1.2483
GP	0.2724	0.3234	0.5672	0.6457	0.8530	0.9547	1.0929	1.2685
ANN	0.2843	0.3194	0.5694	0.6553	0.9073	0.9157	1.1286	1.2298
LBOOST	0.2803	0.2801	0.6378	0.6378	0.8716	0.8679	1.1691	1.1684
RBM	0.2471	0.3330	0.5636	0.6571	0.7811	0.9536	1.1106	1.2385
BAG	0.2414	0.2325	0.5404	0.4180	0.7774	0.7186	1.0354	0.9733
RSET	0.1955	0.2927	0.4816	0.6351	0.7324	0.9185	1.0051	1.2904
MCOST	0.2544	0.2452	0.5956	0.5920	0.8314	0.7683	1.1951	1.2008
ABOOST	0.2581	0.2285	0.5709	0.4677	0.7583	0.7846	1.2123	1.1019
DTABLE	0.2897	0.3127	0.6143	0.5250	0.8738	0.7807	1.2349	1.2937
ADT	0.2311	0.1743	0.5100	0.5359	0.8530	0.9345	1.1784	1.2062
SMO	0.2830	0.3273	0.5977	0.6601	0.8690	0.9039	1.1532	1.2573
IB1	0.2907	0.3589	0.6393	0.5449	0.8893	0.9033	1.1869	1.1733
IBK	0.2468	0.2911	0.5323	0.5259	0.8064	0.8752	1.1663	1.2316
PART	0.2358	0.2648	0.5597	0.4849	0.7993	0.8544	1.0916	1.2426
ONER	0.2913	0.3330	0.5992	0.6641	0.9233	0.8327	1.2030	1.1392
JRIP	0.2657	0.2295	0.5760	0.5084	0.8336	0.9710	1.1460	1.2144
RDR	0.3113	0.2588	0.5636	0.6408	0.8671	0.8251	1.1781	1.2616
J48	0.2401	0.2518	0.5829	0.5483	0.8066	0.9103	1.1543	1.0059
NBAYES	0.3402	0.3619	0.6493	0.6824	0.9207	0.9494	1.1720	1.2624
HPIPES	0.3824	0.5788	0.7027	0.7244	0.9657	0.9843	1.3473	1.2837
LWLS	0.2341	0.2718	0.5522	0.5413	0.7962	0.8212	1.1540	1.1782
Average	0.2724	0.3061	0.5786	0.5904	0.8384	0.8771	1.1457	1.1989
Std. Dev	0.0444	0.0792	0.0569	0.0794	0.0741	0.0676	0.0853	0.0837
Median	0.2657	0.2927	0.5760	0.6300	0.8530	0.8791	1.1663	1.2298
Min	0.1955	0.1743	0.4228	0.4180	0.6085	0.7186	0.8899	0.9733
Max	0.3824	0.5788	0.7027	0.7244	0.9657	0.9843	1.3473	1.2937

Table 4.13: NECM Results for JM1 Dataset, c=20, contd...

	JM1-8850	JM1-4425	
Methods	Fit	Fit	Test
CBR	1.4259	1.4965	1.4791
TD	1.3610	1.4075	1.6499
LR	1.5720	1.5765	1.5471
LOC	1.5815	1.5722	1.5907
GP	1.5261	1.5164	1.5284
ANN	1.4651	1.4531	1.4400
LBOOST	1.5285	1.5779	1.5105
RBM	1.5339	1.5205	1.6576
BAG	1.4205	1.4875	1.3964
RSET	1.4356	1.4581	1.5812
MCOST	1.5539	1.5959	1.2050
ABOOST	1.5585	1.6061	1.4965
DTABLE	1.5860	1.5781	1.6617
ADT	1.5551	1.5907	1.8070
SMO	1.5708	1.5605	1.5928
IB1	1.6054	1.7435	1.5957
IBK	1.5031	1.5550	1.4875
PART	1.5316	1.6059	1.2707
ONER	1.5899	1.6206	1.6734
JRIP	1.5296	1.5817	1.4676
RDR	1.5719	1.6009	1.3532
J48	1.4997	1.5722	1.8594
NBAYES	1.5711	1.5478	1.5819
HPIPES	1.7672	1.8038	1.7089
LWLS	1.5532	1.5977	1.4348
Average	1.5359	1.5691	1.5431
Std. Dev	0.0780	0.0827	0.1511
Median	1.5532	1.5765	1.5471
Min	1.3610	1.4075	1.2050
Max	1.7672	1.8038	1.8594

Table 4.14: NECM Results for JM1 Dataset, c=30

	JM1-13C		JM1-17C		JM1-20C		JM1-23C	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Test
CBR	0.4839	0.6291	0.8698	0.7252	1.2158	1.2266	1.5318	1.5227
TD	0.2800	0.4092	0.5887	0.8893	0.8502	1.2502	1.2415	1.7435
LR	0.3731	0.4345	0.8344	0.9104	1.2189	1.2418	1.6491	1.6955
LOC	0.4636	0.5153	0.8851	0.9466	1.2920	1.2345	1.6727	1.7694
GP	0.3854	0.4631	0.8025	0.9201	1.2099	1.3596	1.5395	1.8051
ANN	0.4007	0.4591	0.8016	0.9297	1.2951	1.3064	1.5906	1.7433
LBOOST	0.3967	0.3965	0.9002	0.9002	1.2285	1.2277	1.6491	1.6331
RBM	0.3469	0.4727	0.7928	0.9315	1.1015	1.3556	1.5649	1.7546
BAG	0.3412	0.3290	0.7606	0.5778	1.0978	1.0110	1.4589	1.3712
RSET	0.2754	0.4092	0.6806	0.8975	1.0388	1.3008	1.4184	1.8398
MCOST	0.3575	0.3483	0.8459	0.8332	1.1714	1.0720	1.6930	1.7065
ABOOST	0.3711	0.3283	0.8061	0.6547	1.0703	1.1108	1.7128	1.5564
DTABLE	0.4094	0.4391	0.8676	0.7301	1.2364	1.0759	1.7431	1.8380
ADT	0.3276	0.2409	0.7210	0.7530	1.2071	1.3365	1.6661	1.7017
SMO	0.3994	0.4671	0.8450	0.9376	1.2288	1.2806	1.6306	1.7861
IB1	0.4104	0.5120	0.9047	0.7560	1.2574	1.2800	1.6745	1.6508
IBK	0.3499	0.4175	0.7524	0.7400	1.1408	1.2463	1.6489	1.7502
PART	0.3322	0.3746	0.7919	0.6809	1.1310	1.2170	1.5408	1.7766
ONER	0.4110	0.4661	0.8465	0.9355	1.3055	1.1588	1.7035	1.5910
JRIP	0.3755	0.3160	0.8142	0.7075	1.1793	1.3984	1.6183	1.7150
RDR	0.4410	0.3553	0.7958	0.9183	1.2296	1.1569	1.6658	1.7931
J48	0.3399	0.3583	0.8242	0.7684	1.1411	1.2983	1.6317	1.4090
NBAYES	0.4799	0.5150	0.9177	0.9689	1.3030	1.3458	1.6571	1.7887
HPIPES	0.5387	0.8417	0.9952	0.9958	1.3620	1.3919	1.9043	1.8177
LWLS	0.3306	0.3849	0.7814	0.7584	1.1166	1.1529	1.6314	1.6685
Average	0.3848	0.4353	0.8170	0.8307	1.1852	1.2414	1.6175	1.6971
Std. Dev	0.0626	0.1173	0.0817	0.1154	0.1067	0.1036	0.1234	0.1245
Median	0.3755	0.4175	0.8142	0.8893	1.2099	1.2463	1.6489	1.7433
Min	0.2754	0.2409	0.5887	0.5778	0.8502	1.0110	1.2415	1.3712
Max	0.5387	0.8417	0.9952	0.9958	1.3620	1.3984	1.9043	1.8398

Table 4.14: NECM Results for JM1 Dataset, c=30, contd...

	JM1-8850	JM1-4425	
Methods	Fit	Fit	Test
CBR	2.0146	2.1112	2.0870
TD	1.9169	1.9837	2.3460
LR	2.2194	2.2273	2.1776
LOC	2.2312	2.2185	2.2438
GP	2.1487	2.1356	2.1498
ANN	2.0436	2.0226	2.0072
LBOOST	2.1522	2.2219	2.1229
RBM	2.1644	2.1487	2.3492
BAG	2.0069	2.0931	1.9591
RSET	2.0254	2.0547	2.2344
MCOST	2.1946	2.2445	1.6411
ABOOST	2.2026	2.2683	2.0931
DTABLE	2.2402	2.2289	2.3532
ADT	2.1958	2.2461	2.6025
SMO	2.2183	2.2045	2.2504
IB1	2.2676	2.4621	2.2375
IBK	2.1223	2.1968	2.0954
PART	2.1633	2.2680	1.7046
ONER	2.2453	2.2895	2.3582
JRIP	2.1601	2.2348	2.0438
RDR	2.2205	2.2608	1.8572
J48	2.1177	2.2208	2.6820
NBAYES	2.2185	2.1873	2.2350
HPIPES	2.4972	2.5496	2.4140
LWLS	2.1939	2.2576	2.0133
Average	2.1672	2.2135	2.1703
Std. Dev	0.1120	0.1198	0.2409
Median	2.1939	2.2219	2.1776
Min	1.9169	1.9837	1.6411
Max	2.4972	2.5496	2.6820

Table 4.15: NECM Results for JM1 Dataset, c=50

	JM1-13C		JM1-17C		JM1-20C		JM1-23C	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Test
CBR	0.7632	1.0150	1.3764	1.1354	1.9297	1.9519	2.4251	2.4162
TD	0.4396	0.6554	0.9204	1.4080	1.3336	1.9924	1.9448	2.7705
LR	0.5926	0.7006	1.3230	1.4472	1.9328	1.9671	2.6142	2.6917
LOC	0.7363	0.8280	1.3978	1.4955	2.0453	1.9486	2.6481	2.8118
GP	0.6116	0.7425	1.2729	1.4689	1.9238	2.1692	2.4328	2.8783
ANN	0.6335	0.7385	1.2660	1.4786	2.0708	2.0880	2.5146	2.7702
LBOOST	0.6295	0.6294	1.4249	1.4249	1.9424	1.9474	2.6091	2.5625
RBM	0.5464	0.7522	1.2512	1.4804	1.7423	2.1597	2.4736	2.7866
BAG	0.5407	0.5220	1.2008	0.8975	1.7386	1.5957	2.3060	2.1671
RSET	0.4350	0.6420	1.0787	1.4222	1.6515	2.0655	2.2449	2.9386
MCOST	0.5637	0.5546	1.3465	1.3157	1.8516	1.6792	2.6889	2.7181
ABOOST	0.5973	0.5279	1.2765	1.0286	1.6942	1.7630	2.7138	2.4652
DTABLE	0.6488	0.6919	1.3742	1.1402	1.9615	1.6663	2.7595	2.9266
ADT	0.5205	0.3739	1.1432	1.1873	1.9154	2.1406	2.6414	2.6927
SMO	0.6322	0.7465	1.3396	1.4925	1.9483	2.0340	2.5855	2.8439
IB1	0.6498	0.8180	1.4355	1.1782	1.9938	2.0335	2.6499	2.6059
IBK	0.5560	0.6703	1.1927	1.1683	1.8097	1.9885	2.6140	2.7874
PART	0.5251	0.5941	1.2563	1.0730	1.7943	1.9424	2.4392	2.8447
ONER	0.6505	0.7322	1.3411	1.4783	2.0700	1.8111	2.7046	2.4947
JRIP	0.5949	0.4890	1.2907	1.1055	1.8707	2.2530	2.5629	2.7163
RDR	0.7004	0.5482	1.2603	1.4732	1.9547	1.8204	2.6412	2.8560
J48	0.5394	0.5712	1.3067	1.2087	1.8100	2.0742	2.5865	2.2151
NBAYES	0.7592	0.8210	1.4545	1.5419	2.0675	2.1386	2.6273	2.8413
HPIPES	0.8513	1.3673	1.5802	1.5386	2.1546	2.2072	3.0182	2.8858
LWLS	0.5234	0.6111	1.2397	1.1927	1.7574	1.8164	2.5862	2.6493
Average	0.6096	0.6937	1.2940	1.3113	1.8786	1.9702	2.5613	2.6935
Std. Dev	0.0992	0.1938	0.1312	0.1881	0.1720	0.1762	0.1995	0.2066
Median	0.5973	0.6703	1.2907	1.4080	1.9238	1.9885	2.6091	2.7702
Min	0.4350	0.3739	0.9204	0.8975	1.3336	1.5957	1.9448	2.1671
Max	0.8513	1.3673	1.5802	1.5419	2.1546	2.2530	3.0182	2.9386

Table 4.15: NECM Results for JM1 Dataset, c=50, contd...

	JM1-8850	JM1-4425	
Methods	Fit	Fit	Test
CBR	3.1920	3.3406	3.3028
TD	3.0288	3.1363	3.7381
LR	3.5144	3.5290	3.4386
LOC	3.5306	3.5112	3.5501
GP	3.3939	3.3740	3.3928
ANN	3.2007	3.1616	3.1417
LBOOST	3.3997	3.5101	3.3478
RBM	3.4254	3.4052	3.7322
BAG	3.1798	3.3045	3.0845
RSET	3.2051	3.2479	3.5406
MCOST	3.4759	3.5417	2.5134
ABOOST	3.4907	3.5926	3.2863
DTABLE	3.5487	3.5306	3.7363
ADT	3.4772	3.5568	4.1934
SMO	3.5132	3.4927	3.5656
IB1	3.5919	3.8994	3.5211
IBK	3.3607	3.4805	3.3112
PART	3.4266	3.5923	2.5724
ONER	3.5560	3.6273	3.7277
JRIP	3.4211	3.5410	3.1964
RDR	3.5176	3.5806	2.8651
J48	3.3539	3.5180	4.3272
NBAYES	3.5134	3.4664	3.5412
HPIPES	3.9571	4.0411	3.8242
LWLS	3.4753	3.5774	3.1704
Average	3.4300	3.5023	3.4248
Std. Dev	0.1802	0.1945	0.4211
Median	3.4753	3.5180	3.4386
Min	3.0288	3.1363	2.5134
Max	3.9571	4.0411	4.3272

4.4.2 ECM Results for the KC2 System

Tables 4.16, 4.17, 4.18, and 4.19 display the classification performance results of all the classification techniques on the datasets with different levels of noise, for the KC2 system. Again, it is evident from these results that the value of expected cost of misclassification improves (goes down) as the level of noise elimination goes from the most conservative (23C) to the least conservative (13C).

4.5 ANOVA Results

As mentioned in the Section 3.4.4, Two Way: Randomized Complete Block Design approach was employed to investigate whether or not the twenty five classification techniques and the datasets with different levels of noise filtering yield significantly different NECM values with respect to one another respectively. The NECM computed for the fit and test data sets, was used as the response variable for the ANOVA models. The results are presented in the Tables 4.20, 4.21, 4.22, and 4.23.

The notations used in these tables are as follows: DF - degrees of freedom, SS - sums of squares, MS - mean squares, and F - the F statistic.

With p-values less than 0.0001 displayed as 0.0000, Table 4.20 indicates that for the JM1 system, quality-of-fit wise, the classification methods we used performed significantly differently from one another ($\alpha = 1\%$), and the datasets with different levels of noise filtering are also significantly different ($\alpha = 1\%$) from one another

Table 4.16: NECM Results for KC2 Dataset, c=10

	KC2-13C		KC2-17C		KC2-23C		KC2-520	KC2-260	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Fit	Test
CBR	0.0622	0.1866	0.1336	0.1065	0.3581	0.2969	0.5942	0.5808	0.5269
TD	0.0048	0.0622	0.1659	0.1157	0.2751	0.2402	0.4750	0.5808	0.6654
LR	0.0909	0.2536	0.0691	0.0741	0.2882	0.3493	0.6077	0.5962	0.5077
LOC	0.0766	0.0670	0.1244	0.1574	0.3100	0.2751	0.5673	0.5538	0.5462
GP	0.0000	0.1435	0.0046	0.1528	0.2533	0.3843	0.4923	0.3269	0.7038
ANN	0.0335	0.1770	0.1429	0.1898	0.2969	0.4105	0.6115	0.4962	0.6000
LBOOST	0.0048	0.0957	0.1751	0.0880	0.3450	0.3974	0.6000	0.6962	0.6500
RBM	0.0000	0.1053	0.0369	0.0926	0.2009	0.3450	0.4654	0.5346	0.5885
BAG	0.0048	0.0957	0.1889	0.0417	0.3013	0.4017	0.5942	0.7038	0.4500
RSET	0.1340	0.0909	0.1244	0.1574	0.3013	0.2271	0.4173	0.5346	0.6423
MCOST	0.0048	0.0957	0.1843	0.1759	0.4017	0.2664	0.6288	0.7500	0.6615
ABOOST	0.0048	0.0957	0.1705	0.1296	0.3275	0.2751	0.8212	0.6654	0.7654
DTABLE	0.0048	0.0957	0.1429	0.1296	0.3013	0.3930	0.5346	0.5577	0.5923
ADT	0.0048	0.0957	0.1244	0.1065	0.2489	0.2227	0.5615	0.6269	0.7923
SMO	0.0670	0.1675	0.1797	0.1250	0.3013	0.3275	0.5885	0.5808	0.6346
IB1	0.0287	0.1818	0.2442	0.2130	0.2533	0.3799	0.6885	0.5923	0.9692
IBK	0.0287	0.1962	0.1843	0.1389	0.2926	0.3057	0.5673	0.5577	0.6000
PART	0.0000	0.0957	0.1797	0.2593	0.3100	0.5459	0.5692	0.6500	0.7538
ONER	0.0048	0.0957	0.1244	0.0231	0.3013	0.3624	0.5635	0.5885	0.5885
JRIP	0.0048	0.0957	0.1982	0.1389	0.2620	0.3624	0.5615	0.6462	0.5500
RDR	0.0048	0.0957	0.1843	0.0417	0.2620	0.4017	0.5538	0.6231	0.5308
J48	0.0000	0.0957	0.1843	0.2361	0.3100	0.5459	0.5596	0.6500	0.6346
NBAYES	0.0861	0.0766	0.1889	0.1343	0.2664	0.4148	0.6115	0.5962	0.5885
HPIPES	0.0813	0.1100	0.1198	0.1667	0.3013	0.3362	0.6712	0.6346	0.5846
LWLS	0.0000	0.1914	0.1889	0.0556	0.3057	0.3886	0.5635	0.5846	0.7077
Average	0.0295	0.1225	0.1506	0.1300	0.2950	0.3542	0.5788	0.5963	0.6334
Std. Dev	0.0386	0.0494	0.0532	0.0593	0.0397	0.0828	0.0784	0.0809	0.1082
Median	0.0048	0.0957	0.1705	0.1296	0.3013	0.3624	0.5673	0.5923	0.6000
Min	0.0000	0.0622	0.0046	0.0231	0.2009	0.2227	0.4173	0.3269	0.4500
Max	0.1340	0.2536	0.2442	0.2593	0.4017	0.5459	0.8212	0.7500	0.9692

Table 4.17: NECM Results for KC2 Dataset, c=20

	KC2-13C		KC2-17C		KC2-23C		KC2-520	KC2-260	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Fit	Test
CBR	0.1100	0.3301	0.2258	0.1528	0.6201	0.5153	1.0173	1.0038	0.8731
TD	0.0048	0.1100	0.3041	0.2083	0.4498	0.3712	0.8019	1.0038	1.1654
LR	0.1388	0.4450	0.1152	0.1204	0.5066	0.6550	1.0500	1.0192	0.8538
LOC	0.1244	0.1148	0.2166	0.2963	0.5284	0.4934	0.9712	0.9385	0.9308
GP	0.0000	0.2871	0.0046	0.2917	0.4716	0.7336	0.8385	0.5192	1.2423
ANN	0.0335	0.3206	0.1889	0.2824	0.5153	0.7598	1.0538	0.8038	0.9846
LBOOST	0.0048	0.1914	0.3134	0.1343	0.6070	0.7467	1.0231	1.1962	1.0731
RBM	0.0000	0.2010	0.0369	0.1389	0.3319	0.6070	0.7923	0.9192	1.0115
BAG	0.0048	0.1914	0.3272	0.0417	0.5197	0.7511	1.0173	1.2038	0.7192
RSET	0.2297	0.1388	0.2166	0.2963	0.5197	0.4017	0.7058	0.9192	1.1423
MCOST	0.0048	0.1914	0.3226	0.3148	0.7074	0.4410	1.0712	1.2885	1.1615
ABOOST	0.0048	0.1914	0.3088	0.2222	0.5459	0.4934	1.4173	1.1654	1.3038
DTABLE	0.0048	0.1914	0.2350	0.2222	0.5197	0.7424	0.9192	0.9423	1.0154
ADT	0.0048	0.1914	0.2166	0.1991	0.4236	0.3974	0.9654	1.0885	1.4462
SMO	0.1148	0.3110	0.3180	0.2176	0.5197	0.5895	1.0115	1.0038	1.0962
IB1	0.0287	0.3254	0.4286	0.3519	0.3843	0.6419	1.1885	1.0154	1.7385
IBK	0.0287	0.3397	0.3226	0.2315	0.5109	0.5677	0.9712	0.9423	1.0231
PART	0.0000	0.1914	0.3180	0.4907	0.5284	0.9825	0.9731	1.1115	1.3308
ONER	0.0048	0.1914	0.2166	0.0231	0.5197	0.6681	0.9673	1.0115	1.0115
JRIP	0.0048	0.1914	0.3364	0.2315	0.4367	0.6681	0.9654	1.1077	0.8962
RDR	0.0048	0.1914	0.3226	0.0417	0.4367	0.7511	0.9577	1.0846	0.8385
J48	0.0000	0.1914	0.3226	0.4213	0.5284	0.9825	0.9635	1.1115	1.0192
NBAYES	0.1340	0.1244	0.3272	0.2269	0.4410	0.7642	1.0538	1.0192	1.0115
HPIPES	0.1292	0.2057	0.2120	0.3056	0.5197	0.5983	1.1519	1.0962	0.9692
LWLS	0.0000	0.3828	0.3272	0.0556	0.5240	0.7380	0.9673	1.0077	1.2462
Average	0.0448	0.2297	0.2594	0.2207	0.5046	0.6424	0.9926	1.0209	1.0842
Std. Dev	0.0646	0.0875	0.0979	0.1167	0.0760	0.1617	0.1381	0.1493	0.2177
Median	0.0048	0.1914	0.3088	0.2222	0.5197	0.6550	0.9712	1.0154	1.0192
Min	0.0000	0.1100	0.0046	0.0231	0.3319	0.3712	0.7058	0.5192	0.7192
Max	0.2297	0.4450	0.4286	0.4907	0.7074	0.9825	1.4173	1.2885	1.7385

Table 4.18: NECM Results for KC2 Dataset, c=30

	KC2-13C		KC2-17C		KC2-23C		KC2-520	KC2-260	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Fit	Test
CBR	0.1579	0.4737	0.3180	0.1991	0.8821	0.7336	1.4404	1.4269	1.2192
TD	0.0048	0.1579	0.4424	0.3009	0.6245	0.5022	1.1288	1.4269	1.6654
LR	0.1866	0.6364	0.1613	0.1667	0.7249	0.9607	1.4923	1.4423	1.2000
LOC	0.1722	0.1627	0.3088	0.4352	0.7467	0.7118	1.3750	1.3231	1.3154
GP	0.0000	0.4306	0.0046	0.4306	0.6900	1.0830	1.1846	0.7115	1.7808
ANN	0.0335	0.4641	0.2350	0.3750	0.7336	1.1092	1.4962	1.1115	1.3692
LBOOST	0.0048	0.2871	0.4516	0.1806	0.8690	1.0961	1.4462	1.6962	1.4962
RBM	0.0000	0.2967	0.0369	0.1852	0.4629	0.8690	1.1192	1.3038	1.4346
BAG	0.0048	0.2871	0.4654	0.0417	0.7380	1.1004	1.4404	1.7038	0.9885
RSET	0.3254	0.1866	0.3088	0.4352	0.7380	0.5764	0.9942	1.3038	1.6423
MCOST	0.0048	0.2871	0.4608	0.4537	1.0131	0.6157	1.5135	1.8269	1.6615
ABOOST	0.0048	0.2871	0.4470	0.3148	0.7642	0.7118	2.0135	1.6654	1.8423
DTABLE	0.0048	0.2871	0.3272	0.3148	0.7380	1.0917	1.3038	1.3269	1.4385
ADT	0.0048	0.2871	0.3088	0.2917	0.5983	0.5721	1.3692	1.5500	2.1000
SMO	0.1627	0.4545	0.4562	0.3102	0.7380	0.8515	1.4346	1.4269	1.5577
IB1	0.0287	0.4689	0.6129	0.4907	0.5153	0.9039	1.6885	1.4385	2.5077
IBK	0.0287	0.4833	0.4608	0.3241	0.7293	0.8297	1.3750	1.3269	1.4462
PART	0.0000	0.2871	0.4562	0.7222	0.7467	1.4192	1.3769	1.5731	1.9077
ONER	0.0048	0.2871	0.3088	0.0231	0.7380	0.9738	1.3712	1.4346	1.4346
JRIP	0.0048	0.2871	0.4747	0.3241	0.6114	0.9738	1.3692	1.5692	1.2423
RDR	0.0048	0.2871	0.4608	0.0417	0.6114	1.1004	1.3615	1.5462	1.1462
J48	0.0000	0.2871	0.4608	0.6065	0.7467	1.4192	1.3673	1.5731	1.4038
NBAYES	0.1818	0.1722	0.4654	0.3194	0.6157	1.1135	1.4962	1.4423	1.4346
HPIPES	0.1770	0.3014	0.3041	0.4444	0.7380	0.8603	1.6327	1.5577	1.3538
LWLS	0.0000	0.5742	0.4654	0.0556	0.7424	1.0873	1.3712	1.4308	1.7846
Average	0.0601	0.3368	0.3681	0.3115	0.7142	0.9307	1.4065	1.4455	1.5349
Std. Dev	0.0910	0.1266	0.1433	0.1752	0.1132	0.2418	0.1979	0.2179	0.3284
Median	0.0048	0.2871	0.4470	0.3148	0.7380	0.9607	1.3750	1.4385	1.4385
Min	0.0000	0.1579	0.0046	0.0231	0.4629	0.5022	0.9942	0.7115	0.9885
Max	0.3254	0.6364	0.6129	0.7222	1.0131	1.4192	2.0135	1.8269	2.5077

Table 4.19: NECM Results for KC2 Dataset, c=50

	KC2-13C		KC2-17C		KC2-23C		KC2-520	KC2-260	
Methods	Fit	Test	Fit	Test	Fit	Test	Fit	Fit	Test
CBR	0.2536	0.7608	0.5023	0.2917	1.4061	1.1703	2.2865	2.2731	1.9115
TD	0.0048	0.2536	0.7189	0.4861	0.9738	0.7642	1.7827	2.2731	2.6654
LR	0.2823	1.0191	0.2535	0.2593	1.1616	1.5721	2.3769	2.2885	1.8923
LOC	0.2679	0.2584	0.4931	0.7130	1.1834	1.1485	2.1827	2.0923	2.0846
GP	0.0000	0.7177	0.0046	0.7083	1.1266	1.7817	1.8769	1.0962	2.8577
ANN	0.0335	0.7512	0.3272	0.5602	1.1703	1.8079	2.3808	1.7269	2.1385
LBOOST	0.0048	0.4785	0.7281	0.2731	1.3930	1.7948	2.2923	2.6962	2.3423
RBM	0.0000	0.4880	0.0369	0.2778	0.7249	1.3930	1.7731	2.0731	2.2808
BAG	0.0048	0.4785	0.7419	0.0417	1.1747	1.7991	2.2865	2.7038	1.5269
RSET	0.5167	0.2823	0.4931	0.7130	1.1747	0.9258	1.5712	2.0731	2.6423
MCOST	0.0048	0.4785	0.7373	0.7315	1.6245	0.9651	2.3981	2.9038	2.6615
ABOOST	0.0048	0.4785	0.7235	0.5000	1.2009	1.1485	3.2058	2.6654	2.9192
DTABLE	0.0048	0.4785	0.5115	0.5000	1.1747	1.7904	2.0731	2.0962	2.2846
ADT	0.0048	0.4785	0.4931	0.4769	0.9476	0.9214	2.1769	2.4731	3.4077
SMO	0.2584	0.7416	0.7327	0.4954	1.1747	1.3755	2.2808	2.2731	2.4808
IB1	0.0287	0.7560	0.9816	0.7685	0.7773	1.4279	2.6885	2.2846	4.0462
IBK	0.0287	0.7703	0.7373	0.5093	1.1659	1.3537	2.1827	2.0962	2.2923
PART	0.0000	0.4785	0.7327	1.1852	1.1834	2.2926	2.1846	2.4962	3.0615
ONER	0.0048	0.4785	0.4931	0.0231	1.1747	1.5852	2.1788	2.2808	2.2808
JRIP	0.0048	0.4785	0.7512	0.5093	0.9607	1.5852	2.1769	2.4923	1.9346
RDR	0.0048	0.4785	0.7373	0.0417	0.9607	1.7991	2.1692	2.4692	1.7615
J48	0.0000	0.4785	0.7373	0.9769	1.1834	2.2926	2.1750	2.4962	2.1731
NBAYES	0.2775	0.2679	0.7419	0.5046	0.9651	1.8122	2.3808	2.2885	2.2808
HPIPES	0.2727	0.4928	0.4885	0.7222	1.1747	1.3843	2.5942	2.4808	2.1231
LWLS	0.0000	0.9569	0.7419	0.0556	1.1790	1.7860	2.1788	2.2769	2.8615
Average	0.0907	0.5512	0.5856	0.4930	1.1334	1.5071	2.2342	2.2948	2.4365
Std. Dev	0.1441	0.2055	0.2347	0.2927	0.1884	0.4028	0.3175	0.3555	0.5506
Median	0.0048	0.4785	0.7235	0.5000	1.1747	1.5721	2.1827	2.2846	2.2846
Min	0.0000	0.2536	0.0046	0.0231	0.7249	0.7642	1.5712	1.0962	1.5269
Max	0.5167	1.0191	0.9816	1.1852	1.6245	2.2926	3.2058	2.9038	4.0462

Table 4.20: Two-Way ANOVA Models for JM1 Fit Datasets

$\frac{C_{II}}{C_I}$	Source	DF	SS	MS	F	p-value
10	Method	24	0.1428	0.0059	12.56	0.0000
	Dataset	4	8.7528	2.1882	4621.05	0.0000
	Error	96	0.0455	0.0005		
	Total	124	8.9411			
20	Method	24	0.4438	0.0185	11.69	0.0000
	Dataset	4	25.1779	6.2945	3978.83	0.0000
	Error	96	0.1519	0.0016		
	Total	124	25.7736			
30	Method	24	0.9124	0.0380	11.24	0.0000
	Dataset	4	50.0842	12.5210	3700.81	0.0000
	Error	96	0.3248	0.0034		
	Total	124	51.3214			
50	Method	24	2.3523	0.0980	10.81	0.0000
	Dataset	4	125.3404	31.3350	3456.85	0.0000
	Error	96	0.8702	0.0091		
	Total	124	128.5628			

suggesting that significant amount of noise is removed at the different levels of noise filtering. We can observe similar pattern in the Table 4.21 for the fit datasets of KC2 systems as in the Table 4.20 for the JM1 system.

However, we are more interested in the results based on the predictive performance of classifiers than those based on the quality of fit. Examining the ANOVA results based on the predictive performance of the classifiers for both the software systems (Tables 4.22 and 4.23) reveals that for all the cost ratios, the NECM values across the datasets (with different noise filtering levels) are significantly different ($\alpha = 1\%$), indicated by p-values less than or equal to 0.0001, i.e. 0.01%, similar to the quality-of-fit results. But it is a different story when it comes to the classification

Table 4.21: Two-Way ANOVA Models for KC2 Fit Datasets

$\frac{C_{II}}{C_I}$	Source	DF	SS	MS	F	p-value
10	Method	24	0.1254	0.0052	2.17	0.0062
	Dataset	3	4.4798	1.4933	620.39	0.0000
	Error	72	0.1733	0.0024		
	Total	99	4.7786			
20	Method	24	0.4074	0.0170	2.05	0.0104
	Dataset	3	13.2316	4.4105	532.78	0.0000
	Error	72	0.5960	0.0083		
	Total	99	14.2350			
30	Method	24	0.8566	0.0357	2.00	0.0127
	Dataset	3	26.6105	8.8702	497.83	0.0000
	Error	72	1.2829	0.0178		
	Total	99	28.7500			
50	Method	24	2.2566	0.0940	1.96	0.0150
	Dataset	3	67.2502	22.4167	467.98	0.0000
	Error	72	3.4488	0.0479		
	Total	99	72.9556			

techniques. While all the classification techniques perform significantly differently on the test datasets for the JM1 system, indicated by very low (in some cases lower than 0.0001) p-values, for the KC2 system, the p-values are much higher for the *Classification Method* factor, indicating that classification techniques are not significantly different at significance level $\alpha = 5\%$ for all the four cost-ratios examined. However, if the significance level were raised to 10%, the classification techniques would be significantly different in terms of their predictive performance only for the cost ratio of 10, as the corresponding p-value of 8.7% would be lower than the specified significance level ($\alpha = 10\%$).

In summary, the surprising finding is that at significance level $\alpha = 5\%$,

Table 4.22: Two-Way ANOVA Models for JM1 Test Datasets

$\frac{C_{II}}{C_I}$	Source	DF	SS	MS	F	p-value
10	Method	24	0.1376	0.0057	4.65	0.0000
	Dataset	4	8.3841	2.0960	1698.48	0.0000
	Error	96	0.1185	0.0012		
	Total	124	8.6402			
20	Method	24	0.4834	0.0201	3.00	0.0000
	Dataset	4	23.7994	5.9499	886.33	0.0000
	Error	96	0.6444	0.0067		
	Total	124	24.9272			
30	Method	24	1.0532	0.0439	2.60	0.0005
	Dataset	4	47.0862	11.7715	698.28	0.0000
	Error	96	1.6184	0.0169		
	Total	124	49.7578			
50	Method	24	2.8653	0.1194	2.33	0.0020
	Dataset	4	117.2742	29.3185	573.23	0.0000
	Error	96	4.9100	0.0511		
	Total	124	125.0495			

Table 4.23: Two-Way ANOVA Models for KC2 Test Datasets

$\frac{C_{II}}{C_I}$	Source	DF	SS	MS	F	p-value
10	Method	24	0.1984	0.0083	1.53	0.0870
	Dataset	3	4.3524	1.4508	267.93	0.0000
	Error	72	0.3899	0.0054		
	Total	99	4.9406			
20	Method	24	0.6924	0.0289	1.31	0.1884
	Dataset	3	12.6190	4.2063	191.31	0.0000
	Error	72	1.5831	0.0220		
	Total	99	14.8945			
30	Method	24	1.5009	0.0625	1.25	0.2343
	Dataset	3	25.2124	8.4041	167.51	0.0000
	Error	72	3.6122	0.0502		
	Total	99	30.3255			
50	Method	24	4.0610	0.1692	1.20	0.2743
	Dataset	3	63.3793	21.1264	149.44	0.0000
	Error	72	10.1784	0.1414		
	Total	99	77.6186			

predictive performance of all the classification techniques is significantly different on the datasets for the JM1 system, but not for the KC2 system.

The other finding that there was significant difference ($\alpha = 1\%$) between the datasets with different levels of noise filtering statistically confirmed our intuitive assumption that the classification performance would improve as more and more noise is eliminated. This was also apparent as the NECM values went down when we went from the most conservative level (23C) to the least conservative level (13C) of noise filtering.

4.6 Multiple Pairwise Comparison Results

From ANOVA Tables for the JM1 system, it was found that the classification methods we used performed significantly differently ($\alpha = 0.01$) from one another in terms of quality-of-fit and predictive quality. On the contrary, ANOVA Tables for the KC2 system revealed that the classification methods were significantly different ($\alpha = 0.01$) in terms of their performance on only the fit dataset, and not the test dataset. It was surprising to observe that for the KC2 system, the classification methods used were not significantly different ($\alpha = 0.05$) in terms of their predictive quality.

From ANOVA tables, we only get to know whether or not there is a significant difference among the given factors (classification techniques in our case). But to get an insight into which factor is different from which other factor(s), if at all, it is

necessary to perform multiple pairwise comparison. In the results presented in the following subsections, the classification techniques that were not significantly different from each other at a given significance level have been clustered in one group, indicated by an identical block letter assigned next to each such technique. Different block letters indicate different clusters of classification methods. It is noteworthy here that a classification technique can, in fact, belong to more than one clusters. It should also be noted that in all these tables, the classification methods being compared have been sorted in the descending order of the respective mean value of NECM at the given cost ratio.

4.6.1 Multiple Pairwise Comparison Results for JM1 System

Tables 4.24, 4.25, 4.26, and 4.27 show the results of multiple pairwise comparisons in terms of quality-of-fit of the different classifiers, for significance level of 1%, 5%, and 10%, at the cost ratios of 10, 20, 30, and 50 respectively.

Let's understand these tables with the help of some examples. Take Table 4.24 for instance. In Table 4.24, for significance level of 1%, we can say that Hyperpipes and IB1 do not perform significantly differently from each other, and belong to the same cluster (cluster A) of classifiers. If we were to compare the performance of Hyperpipes with that of LOC, we could say that they do not belong to the same cluster, implying that they are significantly different in terms of their performance at significance level of 1%. IB1 belongs to cluster A and LOC belongs

Table 4.24: Multiple Pairwise Comparison Results for JM1-Fit Datasets, c=10

Tukey Grouping												Mean	N	Methods
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$								
	A				A				A			0.6108	5	HPIPES
	B				B		B		A			0.5580	5	IB1
C	B		C		B		B		C			0.5451	5	LOC
C	B		C		B		B		C			0.5437	5	ONER
C	B		C		B		B		C			0.5430	5	NBAYES
C	B		C		B		B		C	D		0.5392	5	DTABLE
C	B		C		B		B		C	D		0.5354	5	LBOOST
C	B	D	C		B	D	B		C	D		0.5298	5	RDR
C	B	D	C		B	D	B		C	D		0.5265	5	MCOST
C	B	D	C		B	D	B		C	D		0.5237	5	SMO
C	B	D	C		B	D	B		C	D		0.5229	5	LR
C	B	D	C		B	D	B		C	D		0.5180	5	CBR
C	B	D	C		B	D	B		C	D		0.5168	5	JRIP
C	B	D	C		B	D	B		C	D		0.5165	5	ABOOST
C	B	D	C		B	D	B		C	D		0.5149	5	ANN
C	B	D	C		B	D	B	E	C	D		0.5117	5	ADT
C	B	D	C		B	D	B	E	C	D		0.5109	5	J48
C	B	D	C		B	D	B	E	C	D		0.5102	5	LWLS
C		D	C	E	B	D	B	E	C	D		0.5062	5	GP
C		D	C	E		D	B	E	C	D		0.5049	5	IBK
C		D	C	E		D	B	E	C	D		0.5041	5	PART
C	E	D	C	E		D		E	C	D		0.4982	5	RBM
	E	D		E		D		E		D		0.4825	5	BAG
F	E			E	F			E	F			0.4556	5	RSET
F					F				F			0.4227	5	TD

to cluster C. This does not mean that they both are significantly different, because there is another cluster (cluster B), which both of them belong to. Hence, we can say that IB1 and LOC are not significantly different in terms of their performance at significance level of 1%. On the same line of argument, we can draw many comparisons between various classification techniques to see whether or not they are significantly different from each other.

Table 4.25: Multiple Pairwise Comparison Results for JM1-Fit Datasets, c=20

Tukey Grouping					
$\alpha = 0.10$	$\alpha = 0.05$	$\alpha = 0.01$	Mean	N	Methods
A	A	A	1.0404	5	HPIPES
B	B A	B A	0.9499	5	IB1
C B	B C	B	0.9275	5	ONER
C B	B C	B	0.9260	5	NBAYES
C B	B C	B	0.9257	5	LOC
C B	B C	B C	0.9181	5	DTABLE
C B	B C D	B C	0.9073	5	LBOOST
C B D	B C D	B C	0.9042	5	RDR
C B D	B C D	B C	0.8945	5	MCOST
C B D	B C D	B C	0.8927	5	SMO
C B D	B C D	B C	0.8917	5	LR
C B D	B C D	B C D	0.8811	5	ABOOST
C B D	B C D	B C D	0.8806	5	JRIP
C B D	B C D	B C D	0.8803	5	CBR
C B D	B C D	B C D	0.8726	5	ADT
C B D	B C D	B C D	0.8712	5	J48
C B D	B E C D	B C D	0.8685	5	ANN
C B D	B E C D	B C D	0.8669	5	LWLS
C E D	B E C D	B C D	0.8614	5	IBK
C E D	B E C D	B C D	0.8604	5	GP
C E D	B E C D	B C D	0.8585	5	PART
C E D	E C D	B C D	0.8446	5	RBM
E D	E D	C D	0.8164	5	BAG
F E	E F	E D	0.7746	5	RSET
F	F	E	0.7058	5	TD

It is evident from Tables 4.24, 4.25, 4.26, and 4.27 that for the JM1 system, Hyperpipes and IB1 perform consistently poorly across a range of cost-ratios, while Treedisc and Roughsets exhibit very good quality-of-fit characteristics.

While quality-of-fit may give a practitioner certain degree of confidence in the classification models built, it is the predictive quality of a classifier that really matters. Tables 4.28, 4.29, 4.30, and 4.31 show how the classification techniques

Table 4.26: Multiple Pairwise Comparison Results for JM1-Fit Datasets, c=30

Tukey Grouping												Mean	N	Methods
$\alpha = 0.10$				$\alpha = 0.05$				$\alpha = 0.01$						
		A				A			A			1.4700	5	HPIPES
B		A		B		A		B	A			1.3418	5	IB1
B		C		B		C		B				1.3112	5	ONER
B		C		B		C		B				1.3090	5	NBAYES
B		C		B		C		B	C			1.3064	5	LOC
B		C		B		C		B	C			1.2971	5	DTABLE
B		C	D	B		C	D	B	C			1.2793	5	LBOOST
B		C	D	B		C	D	B	C			1.2786	5	RDR
B		C	D	B		C	D	B	C			1.2625	5	MCOST
B		C	D	B		C	D	B	C			1.2617	5	SMO
B		C	D	B		C	D	B	C			1.2606	5	LR
B		C	D	B		C	D	B	C	D		1.2457	5	ABOOST
B		C	D	B		C	D	B	C	D		1.2444	5	JRIP
B		C	D	B		C	D	B	C	D		1.2425	5	CBR
B		C	D	B		C	D	B	C	D		1.2336	5	ADT
B		C	D	B	E	C	D	B	C	D		1.2315	5	J48
B		C	D	B	E	C	D	B	C	D		1.2235	5	LWLS
B	E	C	D	B	E	C	D	B	C	D		1.2221	5	ANN
B	E	C	D	B	E	C	D	B	C	D		1.2178	5	IBK
B	E	C	D	B	E	C	D	B	C	D		1.2146	5	GP
B	E	C	D	B	E	C	D	B	C	D		1.2128	5	PART
	E	C	D		E	C	D	B	C	D		1.1910	5	RBM
	E		D		E		D		C	D		1.1503	5	BAG
	E	F			E	F			E	D		1.0936	5	RSET
		F				F			E			0.9888	5	TD

compare with each other in terms of predictive quality. It is evident from these tables that Hyperpipes consistently performs very poorly as compared to other classifiers, whereas Bagging, Adaboost, and MetaCost are among the classifiers that perform consistently well on the test data across a range of cost-ratios.

Table 4.27: Multiple Pairwise Comparison Results for JM1-Fit Datasets, c=50

Tukey Grouping											Mean	N	Methods
$\alpha = 0.10$				$\alpha = 0.05$				$\alpha = 0.01$					
		A				A			A		2.3291	5	HPIPES
B		A		B		A		B	A		2.1257	5	IB1
B		C		B		C		B	A		2.0787	5	ONER
B		C		B		C		B	A	C	2.0750	5	NBAYES
B		C		B		C		B		C	2.0677	5	LOC
B		C		B		C		B		C	2.0549	5	DTABLE
B		C	D	B		C	D	B		C	2.0274	5	RDR
B		C	D	B		C	D	B		C	2.0232	5	LBOOST
B		C	D	B		C	D	B		C	1.9996	5	SMO
B		C	D	B		C	D	B		C	1.9985	5	MCOST
B		C	D	B		C	D	B		C	1.9983	5	LR
B		C	D	B		C	D	B	D	C	1.9749	5	ABOOST
B		C	D	B		C	D	B	D	C	1.9721	5	JRIP
B		C	D	B		C	D	B	D	C	1.9670	5	CBR
B		C	D	B	E	C	D	B	D	C	1.9555	5	ADT
B		C	D	B	E	C	D	B	D	C	1.9521	5	J48
B	E	C	D	B	E	C	D	B	D	C	1.9369	5	LWLS
B	E	C	D	B	E	C	D	B	D	C	1.9306	5	IBK
B	E	C	D	B	E	C	D	B	D	C	1.9293	5	ANN
B	E	C	D	B	E	C	D	B	D	C	1.9230	5	GP
B	E	C	D	B	E	C	D	B	D	C	1.9214	5	PART
	E	C	D		E	C	D	B	D	C	1.8837	5	RBM
	E		D		E		D		D	C	1.8181	5	BAG
	E	F			E	F		E	D		1.7316	5	RSET
		F				F		E			1.5550	5	TD

4.6.2 Multiple Pairwise Comparison Results for KC2 System

Multiple pairwise comparison results for KC2 system, presented in Tables 4.32, 4.33, 4.34, and 4.35 reveal a rather different trend from the one exhibited by the results for the JM1 system. MetaCost, LogitBoost, and Bagging perform consistently poorly in terms of quality-of-fit, with MetaCost being the worst. GP appears to consistently outperform all the classifiers as far as the quality-of-fit goes.

Table 4.28: Multiple Pairwise Comparison Results for JM1-Test Datasets, c=10

Tukey Grouping										
$\alpha = 0.10$			$\alpha = 0.05$		$\alpha = 0.01$			Mean	N	Methods
	A			A		A		0.6198	5	HPIPES
B	A		B	A	B	A		0.5646	5	NBAYES
B	A		B	A	B	A		0.5632	5	RBM
B	A	C	B	A	B	A		0.5571	5	LOC
B	A	C	B	A	B	A		0.5550	5	ONER
B	A	C	B	A	B	A		0.5522	5	SMO
B	A	C	B	A	B	A		0.5509	5	RSET
B	A	C	B	A	B	A		0.5487	5	GP
B	A	C	B	A	B	A		0.5432	5	IB1
B	A	C	B	A	B	A		0.5424	5	TD
B	A	C	B	A	B	A		0.5423	5	DTABLE
B		C	B	A	B	A	C	0.5362	5	ADT
B		C	B		B	A	C	0.5349	5	ANN
B		C	B		B	A	C	0.5349	5	LR
B		C	B		B	A	C	0.5298	5	LBOOST
B		C	B	C	B	A	C	0.5271	5	J48
B	D	C	B	C	B		C	0.5202	5	JRIP
B	D	C	B	C	B		C	0.5197	5	RDR
B	D	C	B	C	B		C	0.5146	5	IBK
B	D	C	B	C	B		C	0.5120	5	CBR
B	D	C	B	C	B		C	0.5033	5	LWLS
B	D	C	B	C	B		C	0.4962	5	PART
	D	C	B	C	B		C	0.4843	5	MCOST
	D	C	B	C	B		C	0.4831	5	ABOOST
	D			C			C	0.4459	5	BAG

As far as the predictive quality is concerned for KC2, the classification techniques perform differently only at significance level of 10%, at the cost ratio of 10 (Table 4.36). For all other cost-ratios and significance levels, statistically speaking, there is no significant difference [5] in the predictive performance of the different

[5] That there is no significant difference in the performance of two classifiers does not imply that the two classifiers are misclassifying identical instances.

Table 4.29: Multiple Pairwise Comparison Results for JM1-Test Datasets, c=20

Tukey Grouping									Mean	N	Methods
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$					
	A			A			A		1.0560	5	HPIPES
B	A		B	A		B	A		0.9680	5	RBM
B	A		B	A		B	A		0.9676	5	NBAYES
B	A		B	A		B	A		0.9495	5	LOC
B	A		B	A		B	A		0.9483	5	SMO
B	A		B	A		B	A		0.9441	5	GP
B	A		B	A		B	A		0.9436	5	RSET
B	A		B	A	C	B	A		0.9350	5	TD
B	A		B	A	C	B	A		0.9316	5	ADT
B	A	C	B	A	C	B	A		0.9285	5	ONER
B	A	C	B	A	C	B	A		0.9152	5	IB1
B	A	C	B	A	C	B	A		0.9152	5	J48
B	A	C	B	A	C	B	A		0.9148	5	DTABLE
B	A	C	B	A	C	B	A		0.9134	5	LR
B	A	C	B	A	C	B	A		0.9120	5	ANN
B	A	C	B	A	C	B	A		0.8929	5	LBOOST
B	A	C	B	A	C	B	A		0.8823	5	IBK
B	A	C	B	A	C	B	A		0.8782	5	JRIP
B	A	C	B	A	C	B	A		0.8751	5	CBR
B		C	B	A	C	B	A		0.8679	5	RDR
B		C	B		C	B	A		0.8495	5	LWLS
B		C	B		C	B			0.8235	5	PART
B		C	B		C	B			0.8159	5	ABOOST
B		C	B		C	B			0.8023	5	MCOST
		C			C	B			0.7478	5	BAG

classifiers for the KC2 system. It is evident from Tables 4.37, 4.38, and 4.39 that all the classification techniques belong to the same cluster, indicating that there is no significant difference in their performance at the given significance level and cost-ratio.

Table 4.30: Multiple Pairwise Comparison Results for JM1-Test Datasets, c=30

Tukey Grouping									Mean	N	Methods
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$					
	A			A			A		1.4922	5	HPIPES
B	A		B	A		B	A		1.3727	5	RBM
B	A		B	A		B	A		1.3707	5	NBAYES
B	A		B	A	C	B	A		1.3444	5	SMO
B	A		B	A	C	B	A		1.3419	5	LOC
B	A		B	A	C	B	A		1.3395	5	GP
B	A	C	B	A	C	B	A		1.3363	5	RSET
B	A	C	B	A	C	B	A		1.3276	5	TD
B	A	C	B	A	C	B	A		1.3269	5	ADT
B	A	C	B	A	C	B	A		1.3032	5	J48
B	A	C	B	A	C	B	A		1.3019	5	ONER
B	A	C	B	A	C	B	A		1.2920	5	LR
B	A	C	B	A	C	B	A		1.2892	5	ANN
B	A	C	B	A	C	B	A		1.2873	5	IB1
B	A	C	B	A	C	B	A		1.2873	5	DTABLE
B	A	C	B	A	C	B	A		1.2561	5	LBOOST
B	A	C	B	A	C	B	A		1.2499	5	IBK
B	A	C	B	A	C	B	A		1.2381	5	CBR
B	A	C	B	A	C	B	A		1.2362	5	JRIP
B	A	C	B	A	C	B	A		1.2161	5	RDR
B		C	B	A	C	B	A		1.1956	5	LWLS
B		C	B		C	B	A		1.1508	5	PART
B		C	B		C	B	A		1.1487	5	ABOOST
B		C	B		C	B			1.1202	5	MCOST
		C			C	B			1.0496	5	BAG

4.6.3 Discussion

From the results presented here, it can be observed that there is a lot of overlap between different clusters, and that classification methods performing well on a particular dataset may not necessarily perform as well on some other dataset(s), even if the datasets are from the same domain.

And hence, in our opinion, basing the noise elimination procedure on a few

Table 4.31: Multiple Pairwise Comparison Results for JM1-Test Datasets, c=50

Tukey Grouping									Mean	N	Methods
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$					
	A			A			A		2.3646	5	HPIPES
B	A		B	A		B	A		2.1822	5	RBM
B	A		B	A		B	A		2.1768	5	NBAYES
B	A	C	B	A		B	A		2.1365	5	SMO
B	A	C	B	A		B	A		2.1304	5	GP
B	A	C	B	A		B	A		2.1268	5	LOC
B	A	C	B	A		B	A		2.1218	5	RSET
B	A	C	B	A		B	A		2.1176	5	ADT
B	A	C	B	A		B	A		2.1129	5	TD
B	A	C	B	A		B	A		2.0793	5	J48
B	A	C	B	A		B	A		2.0490	5	LR
B	A	C	B	A		B	A		2.0488	5	ONER
B	A	C	B	A		B	A		2.0434	5	ANN
B	A	C	B	A		B	A		2.0323	5	DTABLE
B	A	C	B	A		B	A		2.0313	5	IB1
B	A	C	B	A		B	A		1.9851	5	IBK
B	A	C	B	A		B	A		1.9824	5	LBOOST
B	A	C	B	A		B	A		1.9643	5	CBR
B	A	C	B	A		B	A		1.9521	5	JRIP
B	A	C	B	A		B	A		1.9126	5	RDR
B	A	C	B	A		B	A		1.8880	5	LWLS
B		C	B			B	A		1.8142	5	ABOOST
B		C	B			B	A		1.8053	5	PART
B		C	B			B	A		1.7562	5	MCOST
		C	B			B			1.6534	5	BAG

selected (base-level) classification techniques (as in [9, 10]) may not be the most appropriate strategy, for the few selected classification techniques may or may not be the most appropriate ones for the data at hand because of possible limitations of the associated representation language. This problem is analogous to situations in which removing outliers does little to improve the quality of fit with a first-order linear regression model if the correct model of the data is quadratic.

Table 4.32: Multiple Pairwise Comparison Results for KC2-Fit Datasets, c=10

Tukey Grouping									Mean	N	Methods
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$					
	A			A			A		0.3352	4	MCOST
B	A		B	A			A		0.3053	4	LBOOST
B	A		B	A			A		0.2997	4	BAG
B	A		B	A		B	A		0.2921	4	ABOOST
B	A		B	A		B	A		0.2861	4	J48
B	A		B	A		B	A		0.2849	4	PART
B	A		B	A		B	A		0.2844	4	NBAYES
B	A		B	A		B	A		0.2843	4	HPIPES
B	A		B	A		B	A		0.2837	4	CBR
B	A		B	A		B	A		0.2822	4	SMO
B	A		B	A		B	A		0.2796	4	IB1
B	A		B	A	C	B	A		0.2778	4	JRIP
B	A		B	A	C	B	A		0.2736	4	RSET
B	A		B	A	C	B	A		0.2698	4	LWLS
B	A	C	B	A	C	B	A		0.2686	4	RDR
B	A	C	B	A	C	B	A		0.2662	4	LOC
B	A	C	B	A	C	B	A		0.2658	4	IBK
B	A	C	B	A	C	B	A		0.2611	4	LR
B	A	C	B	A	C	B	A		0.2566	4	TD
B	A	C	B	A	C	B	A		0.2548	4	ONER
B	A	C	B	A	C	B	A		0.2517	4	DTABLE
B	A	C	B	A	C	B	A		0.2513	4	ADT
B	A	C	B	A	C	B	A		0.2424	4	ANN
B		C	B		C	B	A		0.1931	4	RBM
		C			C	B			0.1462	4	GP

Secondly, there may not be much to choose between different classifiers in terms of their performance, evinced by the big overlaps of clusters of classifiers for JM1 and (in most cases) no significant difference in predictive performance of classifiers for KC2. Trying to explore new classification techniques that may (not necessarily) marginally improve the classification accuracy may not be worth the effort if the training data are noisy to begin with. In our opinion, instead of trying

Table 4.33: Multiple Pairwise Comparison Results for KC2-Fit Datasets, c=20

Tukey Grouping														
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$			Mean	N	Methods
	A			A			A		0.5808	4	MCOST
B	A		B	A			A		0.5303	4	LBOOST
B	A		B	A		B	A		0.5139	4	BAG
B	A		B	A		B	A		0.5062	4	ABOOST
B	A		B	A	C	B	A		0.4906	4	J48
B	A		B	A	C	B	A		0.4900	4	CBR
B	A		B	A	C	B	A		0.4895	4	PART
B	A		B	A	C	B	A		0.4892	4	HPIPES
B	A		B	A	C	B	A		0.4891	4	SMO
B	A		B	A	C	B	A		0.4804	4	NBAYES
B	A	C	B	A	C	B	A		0.4714	4	JRIP
B	A	C	B	A	C	B	A		0.4713	4	RSET
B	A	C	B	A	C	B	A		0.4647	4	LWLS
B	A	C	B	A	C	B	A		0.4642	4	IB1
B	A	C	B	A	C	B	A		0.4622	4	RDR
B	A	C	B	A	C	B	A		0.4520	4	LOC
B	A	C	B	A	C	B	A		0.4511	4	IBK
B	A	C	B	A	C	B	A		0.4449	4	LR
B	A	C	B	A	C	B	A		0.4406	4	TD
B	A	C	B	A	C	B	A		0.4381	4	ONER
B	A	C	B	A	C	B	A		0.4334	4	ADT
B	A	C	B	A	C	B	A		0.4254	4	DTABLE
B	A	C	B	A	C	B	A		0.3854	4	ANN
B		C	B		C	B	A		0.3220	4	RBM
		C			C	B			0.2489	4	GP

to explore the use of state-of-the-art classification technique(s), which may (if at all) improve the classification accuracy by a small margin, one should strive to ensure that the training data is noise-free using appropriate noise-handling techniques. The improvement in classification accuracy after noise elimination is much more significant than (possible) marginal improvement in accuracy with the use of a state-of-the-art technique.

Table 4.34: Multiple Pairwise Comparison Results for KC2-Fit Datasets, c=30

Tukey Grouping									Mean	N	Methods
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$					
	A			A			A		0.8264	4	MCOST
B	A		B	A		B	A		0.7554	4	LBOOST
B	A		B	A		B	A		0.7280	4	BAG
B	A		B	A		B	A		0.7203	4	ABOOST
B	A		B	A	C	B	A		0.6962	4	CBR
B	A		B	A	C	B	A		0.6960	4	SMO
B	A		B	A	C	B	A		0.6952	4	J48
B	A		B	A	C	B	A		0.6942	4	HPIPES
B	A		B	A	C	B	A		0.6940	4	PART
B	A	C	B	A	C	B	A		0.6763	4	NBAYES
B	A	C	B	A	C	B	A		0.6690	4	RSET
B	A	C	B	A	C	B	A		0.6650	4	JRIP
B	A	C	B	A	C	B	A		0.6596	4	LWLS
B	A	C	B	A	C	B	A		0.6558	4	RDR
B	A	C	B	A	C	B	A		0.6488	4	IB1
B	A	C	B	A	C	B	A		0.6377	4	LOC
B	A	C	B	A	C	B	A		0.6364	4	IBK
B	A	C	B	A	C	B	A		0.6288	4	LR
B	A	C	B	A	C	B	A		0.6246	4	TD
B	A	C	B	A	C	B	A		0.6215	4	ONER
B	A	C	B	A	C	B	A		0.6155	4	ADT
B	A	C	B	A	C	B	A		0.5992	4	DTABLE
B	A	C	B	A	C	B	A		0.5284	4	ANN
B		C	B		C	B	A		0.4509	4	RBM
		C			C	B			0.3515	4	GP

4.7 Z-Test Comparison Results of Two Proportions

As mentioned in Section 3.5, Z-test was carried out to compare two different proportions. The two proportions being compared are the proportions of the instances identified as noisy by two different noise filtering approaches.

First, we compared the proportion of the instances identified as noisy (and

Table 4.35: Multiple Pairwise Comparison Results for KC2-Fit Datasets, c=50

Tukey Grouping									Mean	N	Methods
$\alpha = 0.10$			$\alpha = 0.05$			$\alpha = 0.01$					
	A			A			A		1.3176	4	MCOST
B	A		B	A		B	A		1.2055	4	LBOOST
B	A		B	A		B	A		1.1563	4	BAG
B	A		B	A		B	A		1.1486	4	ABOOST
B	A		B	A	C	B	A		1.1097	4	SMO
B	A		B	A	C	B	A		1.1088	4	CBR
B	A	C	B	A	C	B	A		1.1042	4	J48
B	A	C	B	A	C	B	A		1.1042	4	HPIPES
B	A	C	B	A	C	B	A		1.1031	4	PART
B	A	C	B	A	C	B	A		1.0682	4	NBAYES
B	A	C	B	A	C	B	A		1.0644	4	RSET
B	A	C	B	A	C	B	A		1.0522	4	JRIP
B	A	C	B	A	C	B	A		1.0495	4	LWLS
B	A	C	B	A	C	B	A		1.0430	4	RDR
B	A	C	B	A	C	B	A		1.0180	4	IB1
B	A	C	B	A	C	B	A		1.0092	4	LOC
B	A	C	B	A	C	B	A		1.0070	4	IBK
B	A	C	B	A	C	B	A		0.9964	4	LR
B	A	C	B	A	C	B	A		0.9926	4	TD
B	A	C	B	A	C	B	A		0.9883	4	ONER
B	A	C	B	A	C	B	A		0.9796	4	ADT
B	A	C	B	A	C	B	A		0.9468	4	DTABLE
B	A	C	B	A	C	B	A		0.8145	4	ANN
B		C	B		C	B	A		0.7087	4	RBM
		C			C	B			0.5568	4	GP

hence eliminated) by our consensus filter (an ensemble-classifier filter with consensus of 25 classification techniques) to the proportion of the instances identified as noisy by Brodley and Utgoff's [10] consensus filter (an ensemble-classifier filter with consensus of 5 classification techniques: J48, IBk, SMO, JRIP, and LWLStump). For the JM1 software system, our consensus filter removed only 321 out of the 8850 instances, as compared to 1425 out of 8850 instances removed using Brodley and

Table 4.36: Multiple Pairwise Comparison Results for KC2-Test Datasets, c=10

Tukey Grouping $\alpha = 0.10$		Mean	N	Methods
	A	0.4360	4	IB1
B	A	0.4137	4	PART
B	A	0.3781	4	J48
B	A	0.3461	4	GP
B	A	0.3443	4	ANN
B	A	0.3358	4	LWLS
B	A	0.3165	4	ABOOST
B	A	0.3137	4	SMO
B	A	0.3102	4	IBK
B	A	0.3078	4	LBOOST
B	A	0.3043	4	ADT
B	A	0.3035	4	NBAYES
B	A	0.3027	4	DTABLE
B	A	0.2999	4	MCOST
B	A	0.2994	4	HPIPES
B	A	0.2962	4	LR
B	A	0.2868	4	JRIP
B	A	0.2828	4	RBM
B	A	0.2794	4	RSET
B	A	0.2792	4	CBR
B	A	0.2709	4	TD
B	A	0.2675	4	RDR
B	A	0.2674	4	ONER
B	A	0.2614	4	LOC
B		0.2473	4	BAG

Utgoff's filter. When these two proportions were compared using Z-test [115], the computed z-value came out to be of a very high magnitude (as high as 27.82), indicating that the two proportions are statistically different at significance level of 1%. This leads us to conclude that our consensus filtering approach (with twenty five classification techniques) is, statistically speaking, much more conservative than the consensus filtering approach with only five classification techniques (as in [10]).

Table 4.37: Multiple Pairwise Comparison Results for KC2-Test Datasets, c=20

Tukey Grouping $\alpha = 0.10$	Mean	N	Methods
A	0.7644	4	IB1
A	0.7489	4	PART
A	0.6536	4	J48
A	0.6387	4	GP
A	0.6056	4	LWLS
A	0.5869	4	ANN
A	0.5585	4	ADT
A	0.5536	4	SMO
A	0.5527	4	ABOOST
A	0.5428	4	DTABLE
A	0.5405	4	IBK
A	0.5364	4	LBOOST
A	0.5317	4	NBAYES
A	0.5272	4	MCOST
A	0.5197	4	HPIPES
A	0.5186	4	LR
A	0.4968	4	JRIP
A	0.4948	4	RSET
A	0.4896	4	RBM
A	0.4735	4	ONER
A	0.4678	4	CBR
A	0.4637	4	TD
A	0.4588	4	LOC
A	0.4557	4	RDR
A	0.4258	4	BAG

Similarly, we also compared the proportions of the instances identified as noisy by the ensemble-classifier filter with consensus of five [10] and three [9] (J48, IBk, SMO) classification techniques. For the JM1 software system, consensus filter with three classification techniques removed 1696 out of 8850 instances, as compared to 1425 out of 8850 instances removed by consensus filter with five classification techniques. The Z-test for comparing these two proportions also resulted in a relatively

Table 4.38: Multiple Pairwise Comparison Results for KC2-Test Datasets, c=30

Tukey Grouping $\alpha = 0.10$	Mean	N	Methods
A	1.0928	4	IB1
A	1.0841	4	PART
A	0.9312	4	GP
A	0.9292	4	J48
A	0.8754	4	LWLS
A	0.8294	4	ANN
A	0.8127	4	ADT
A	0.7935	4	SMO
A	0.7890	4	ABOOST
A	0.7830	4	DTABLE
A	0.7708	4	IBK
A	0.7650	4	LBOOST
A	0.7600	4	NBAYES
A	0.7545	4	MCOST
A	0.7409	4	LR
A	0.7400	4	HPIPES
A	0.7101	4	RSET
A	0.7068	4	JRIP
A	0.6964	4	RBM
A	0.6797	4	ONER
A	0.6566	4	TD
A	0.6564	4	CBR
A	0.6563	4	LOC
A	0.6438	4	RDR
A	0.6044	4	BAG

higher value of z (5.3449 to be precise), suggesting that these two proportions are statistically different at significance level of 1%.

From the two comparisons described here above, it is clear that as the number of classifiers for an ensemble-classifier consensus filter increases, the filter becomes more conservative at eliminating noise. A relatively more conservative approach to noise elimination may be of interest especially when the training data are scarce.

Table 4.39: Multiple Pairwise Comparison Results for KC2-Test Datasets, c=50

Tukey Grouping $\alpha = 0.10$	Mean	N	Methods
A	1.7544	4	PART
A	1.7497	4	IB1
A	1.5163	4	GP
A	1.4802	4	J48
A	1.415	4	LWLS
A	1.3211	4	ADT
A	1.3144	4	ANN
A	1.2733	4	SMO
A	1.2634	4	DTABLE
A	1.2615	4	ABOOST
A	1.2314	4	IBK
A	1.2222	4	LBOOST
A	1.2164	4	NBAYES
A	1.2091	4	MCOST
A	1.1857	4	LR
A	1.1806	4	HPIPES
A	1.1408	4	RSET
A	1.1269	4	JRIP
A	1.1099	4	RBM
A	1.0919	4	ONER
A	1.0511	4	LOC
A	1.0423	4	TD
A	1.0336	4	CBR
A	1.0202	4	RDR
A	0.9615	4	BAG

In such cases, an ensemble-classifier consensus filter with relatively higher number of classifiers could be the answer. Besides this, using ensemble-classifier filter with relatively large number of (base-level) classifiers can also facilitate in achieving the desired level of conservativeness. Also, intuitively, the level of confidence in the noise removal process increases when the process is based on relatively large number of classifiers, as the possibility of results getting influenced by a few classifiers with

certain bias towards the training data is slimmer.

4.8 Predictive Performance Results

This section presents the predictive performance results of different classification techniques when the models built from the JM1 training datasets were applied to the KC2 datasets and vice versa (Tables 4.40 to 4.94). Results for one of the classification techniques (ANN) were not available, and hence are not presented herein.

4.8.1 Predictive Performance of JM1 Models on KC2 Datasets

This subsection provides the results of predictive quality of the classification models built on the JM1-Fit datasets and evaluated on the KC2 datasets. Predictive performance of classifiers is presented in terms of the two misclassification error rates (Type I and Type II) in Tables 4.40 to 4.45. For each noise filtering level, KC2-Fit and KC2-Test were combined to create one evaluation set each.

Table 4.40: Predictive Quality of JM1-8850 models on KC2 datasets

	KC2-520		KC2-23C		KC2-17C		KC2-13C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
CBR	22.95%	31.13%	16.25%	25.26%	14.49%	21.59%	12.65%	19.77%
TD	27.78%	20.75%	23.42%	10.53%	31.88%	5.68%	30.42%	6.98%
LR	18.36%	27.36%	7.16%	18.95%	3.48%	12.50%	2.11%	10.47%
LOC	23.19%	18.87%	12.40%	9.47%	7.83%	3.41%	5.42%	2.33%
GP	24.88%	17.92%	14.33%	8.42%	9.86%	2.27%	6.63%	2.33%
LBOOST	20.29%	23.58%	9.64%	15.79%	5.22%	10.23%	4.82%	9.30%
RBM	22.46%	24.53%	12.40%	16.84%	8.41%	10.23%	6.33%	10.47%
BAG	18.12%	26.42%	11.29%	16.84%	7.54%	13.64%	6.33%	12.79%
RSET	20.05%	29.25%	10.19%	22.11%	6.67%	15.91%	4.22%	13.95%
MCOST	17.87%	34.91%	10.74%	29.47%	8.41%	25.00%	7.53%	24.42%
ABOOST	27.29%	37.74%	23.42%	34.74%	22.61%	31.82%	20.48%	30.23%
DTABLE	23.91%	25.47%	13.50%	17.89%	9.86%	11.36%	8.13%	10.47%
ADT	24.64%	21.70%	14.88%	13.68%	11.30%	7.95%	9.94%	6.98%
SMO	18.60%	24.53%	7.16%	15.79%	2.90%	9.09%	2.11%	8.14%
IB1	28.26%	49.06%	24.52%	44.21%	22.90%	43.18%	22.29%	44.19%
IBK	18.84%	28.30%	9.64%	22.11%	6.67%	15.91%	5.42%	15.12%
PART	20.29%	33.96%	13.77%	26.32%	11.30%	22.73%	10.24%	22.09%
ONER	22.95%	25.47%	13.50%	17.89%	9.86%	11.36%	7.83%	11.63%
JRIP	22.46%	24.53%	12.95%	17.89%	9.28%	11.36%	8.43%	10.47%
RDR	31.16%	19.81%	22.59%	13.68%	20.00%	9.09%	18.67%	8.14%
J48	17.63%	31.13%	11.85%	25.26%	9.57%	20.45%	8.13%	19.77%
NBAYES	18.84%	19.81%	7.44%	10.53%	2.90%	4.55%	2.41%	2.33%
HPIPES	66.18%	24.53%	64.46%	24.21%	65.51%	18.18%	67.77%	17.44%
LWLS	21.01%	29.25%	13.22%	23.16%	10.43%	18.18%	9.34%	18.60%

Table 4.41: Predictive Quality of JM1-4425 models on KC2 datasets

	KC2-520		KC2-23C		KC2-17C		KC2-13C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
CBR	22.71%	32.08%	15.70%	27.37%	14.20%	25.00%	12.95%	23.26%
TD	24.15%	19.81%	22.31%	10.53%	26.67%	5.68%	26.51%	5.81%
LR	20.77%	21.70%	9.64%	12.63%	5.51%	5.68%	3.31%	4.65%
LOC	23.19%	18.87%	12.40%	9.47%	7.83%	3.41%	5.42%	2.33%
GP	20.77%	18.87%	9.64%	10.53%	5.80%	4.55%	4.22%	4.65%
LBOOST	23.43%	19.81%	13.22%	11.58%	8.99%	5.68%	7.23%	4.65%
RBM	20.29%	19.81%	9.09%	10.53%	4.64%	4.55%	2.71%	4.65%
BAG	23.43%	21.70%	13.77%	14.74%	10.14%	11.36%	8.13%	10.47%
RSET	19.32%	23.58%	8.54%	15.79%	5.22%	9.09%	3.61%	6.98%
MCOST	28.74%	17.92%	19.56%	10.53%	15.94%	5.68%	14.16%	5.81%
ABOOST	29.71%	32.08%	24.79%	27.37%	23.77%	23.86%	22.89%	22.09%
DTABLE	22.95%	23.58%	12.95%	14.74%	8.70%	9.09%	6.63%	8.14%
ADT	19.81%	24.53%	8.82%	16.84%	4.64%	11.36%	4.22%	10.47%
SMO	18.36%	26.42%	7.16%	17.89%	2.90%	11.36%	2.41%	9.30%
IB1	25.85%	36.79%	21.49%	32.63%	20.29%	30.68%	19.58%	30.23%
IBK	20.05%	27.36%	9.64%	18.95%	5.22%	12.50%	3.61%	11.63%
PART	29.71%	21.70%	23.14%	14.74%	20.00%	12.50%	18.07%	11.63%
ONER	23.91%	23.58%	13.77%	14.74%	10.43%	9.09%	8.13%	8.14%
JRIP	24.64%	18.87%	14.33%	11.58%	10.14%	5.68%	8.43%	5.81%
RDR	31.88%	20.75%	22.31%	14.74%	18.55%	9.09%	17.17%	8.14%
J48	17.87%	25.47%	6.89%	16.84%	2.61%	11.36%	2.11%	10.47%
NBAYES	18.60%	21.70%	7.16%	12.63%	2.61%	6.82%	1.81%	4.65%
HPIPES	56.04%	30.19%	52.07%	26.32%	51.59%	20.45%	52.41%	19.77%
LWLS	21.26%	24.53%	11.29%	17.89%	7.54%	11.36%	6.02%	10.47%

Table 4.42: Predictive Quality of JM1-23C models on KC2 datasets

	KC2-520		KC2-23C		KC2-17C		KC2-13C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
CBR	18.12%	32.08%	10.47%	26.32%	8.70%	22.73%	7.23%	20.93%
TD	22.22%	25.47%	16.80%	8.42%	25.22%	3.41%	24.10%	2.33%
LR	21.98%	18.87%	11.02%	9.47%	6.38%	3.41%	3.92%	3.49%
LOC	23.19%	18.87%	12.40%	9.47%	7.83%	3.41%	5.42%	2.33%
GP	24.40%	20.75%	14.05%	11.58%	9.57%	5.68%	6.63%	4.65%
LBOOST	26.81%	18.87%	16.80%	9.47%	12.46%	3.41%	9.34%	3.49%
RBM	22.22%	22.64%	11.29%	14.74%	7.25%	9.09%	6.02%	9.30%
BAG	23.19%	25.47%	13.50%	17.89%	9.86%	12.50%	7.23%	10.47%
RSET	20.05%	25.47%	10.47%	16.84%	6.67%	11.36%	5.42%	10.47%
MCOST	22.22%	29.25%	13.50%	22.11%	10.72%	15.91%	9.04%	13.95%
ABOOST	19.57%	32.08%	11.29%	25.26%	9.28%	20.45%	7.53%	18.60%
DTABLE	19.57%	23.58%	10.19%	14.74%	6.67%	9.09%	4.82%	6.98%
ADT	26.33%	21.70%	16.25%	12.63%	11.88%	6.82%	8.73%	6.98%
SMO	17.63%	23.58%	6.34%	14.74%	2.32%	7.95%	2.11%	5.81%
IB1	21.50%	33.96%	14.88%	28.42%	11.88%	23.86%	10.24%	23.26%
IBK	18.60%	23.58%	7.44%	14.74%	2.90%	7.95%	1.20%	6.98%
PART	20.77%	27.36%	11.29%	20.00%	8.12%	14.77%	7.23%	12.79%
ONER	22.95%	18.87%	12.12%	10.53%	7.54%	4.55%	5.12%	3.49%
JRIP	25.12%	27.36%	15.98%	20.00%	13.04%	14.77%	11.45%	15.12%
RDR	20.05%	22.64%	9.09%	13.68%	4.64%	6.82%	3.01%	6.98%
J48	25.12%	23.58%	15.70%	16.84%	12.75%	12.50%	11.45%	11.63%
NBAYES	20.77%	19.81%	9.64%	10.53%	5.22%	4.55%	3.01%	2.33%
HPIPES	19.81%	24.53%	8.54%	15.79%	3.77%	10.23%	2.71%	8.14%
LWLS	22.95%	22.64%	13.50%	14.74%	9.86%	9.09%	7.53%	8.14%

Table 4.43: Predictive Quality of JM1-20C models on KC2 datasets

	KC2-520		KC2-23C		KC2-17C		KC2-13C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
CBR	18.60%	28.30%	10.19%	22.11%	8.41%	18.18%	6.63%	17.44%
TD	14.01%	33.96%	11.02%	15.79%	12.46%	6.82%	10.54%	6.98%
LR	22.95%	18.87%	12.40%	9.47%	7.83%	3.41%	5.12%	3.49%
LOC	24.88%	17.92%	14.33%	8.42%	9.86%	2.27%	7.23%	2.33%
GP	22.95%	17.92%	12.12%	9.47%	7.54%	3.41%	5.42%	3.49%
LBOOST	21.50%	25.47%	11.29%	16.84%	7.54%	10.23%	5.72%	10.47%
RBM	20.05%	22.64%	8.82%	13.68%	4.64%	7.95%	3.31%	6.98%
BAG	21.98%	27.36%	11.85%	20.00%	7.83%	13.64%	6.33%	11.63%
RSET	18.84%	22.64%	7.44%	13.68%	3.19%	7.95%	1.20%	6.98%
MCOST	21.74%	29.25%	11.85%	22.11%	8.41%	15.91%	6.93%	15.12%
ABOOST	18.12%	25.47%	8.82%	18.95%	6.09%	12.50%	4.82%	10.47%
DTABLE	21.01%	20.75%	11.02%	12.63%	6.96%	7.95%	5.12%	6.98%
ADT	20.53%	24.53%	10.47%	16.84%	6.67%	10.23%	4.52%	9.30%
SMO	20.05%	22.64%	9.09%	13.68%	4.64%	6.82%	2.41%	5.81%
IB1	18.36%	27.36%	9.64%	21.05%	7.25%	15.91%	5.42%	15.12%
IBK	20.77%	23.58%	9.92%	15.79%	6.38%	9.09%	3.92%	8.14%
PART	16.67%	32.08%	7.44%	25.26%	4.93%	19.32%	3.92%	18.60%
ONER	24.64%	18.87%	14.05%	9.47%	9.57%	3.41%	6.93%	3.49%
JRIP	18.36%	28.30%	8.26%	21.05%	4.35%	14.77%	3.01%	12.79%
RDR	24.40%	18.87%	14.05%	11.58%	9.57%	5.68%	8.43%	4.65%
J48	19.32%	29.25%	9.92%	22.11%	6.09%	17.05%	5.12%	15.12%
NBAYES	20.05%	21.70%	8.82%	12.63%	4.35%	6.82%	2.71%	4.65%
HPIPES	17.63%	27.36%	7.99%	18.95%	4.35%	12.50%	3.31%	11.63%
LWLS	21.26%	22.64%	11.85%	15.79%	8.70%	10.23%	6.93%	8.14%

Table 4.44: Predictive Quality of JM1-17C models on KC2 datasets

	KC2-520		KC2-23C		KC2-17C		KC2-13C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
CBR	19.32%	30.19%	10.74%	22.11%	8.41%	17.05%	6.33%	15.12%
TD	14.25%	33.96%	11.29%	15.79%	7.83%	4.55%	8.73%	1.16%
LR	23.43%	19.81%	12.67%	10.53%	8.12%	3.41%	5.12%	3.49%
LOC	24.88%	17.92%	14.33%	8.42%	9.86%	2.27%	7.23%	2.33%
GP	21.98%	19.81%	11.02%	10.53%	6.38%	4.55%	4.22%	3.49%
LBOOST	24.15%	18.87%	13.77%	9.47%	9.57%	3.41%	7.23%	3.49%
RBM	19.08%	25.47%	8.82%	13.68%	3.48%	10.23%	1.81%	8.14%
BAG	22.46%	19.81%	11.85%	11.58%	7.25%	4.55%	5.12%	3.49%
RSET	21.74%	20.75%	11.02%	13.68%	6.67%	7.95%	5.72%	6.98%
MCOST	22.95%	23.58%	12.95%	15.79%	8.70%	9.09%	6.63%	8.14%
ABOOST	21.01%	25.47%	10.47%	17.89%	6.96%	11.36%	5.12%	10.47%
DTABLE	23.19%	25.47%	13.22%	16.84%	9.57%	10.23%	7.23%	9.30%
ADT	21.26%	20.75%	10.74%	11.58%	6.67%	4.55%	4.82%	3.49%
SMO	21.26%	22.64%	10.47%	13.68%	6.09%	6.82%	4.22%	5.81%
IB1	21.98%	26.42%	12.40%	20.00%	8.70%	13.64%	7.23%	12.79%
IBK	22.95%	22.64%	12.40%	13.68%	8.41%	6.82%	6.63%	5.81%
PART	21.98%	26.42%	11.57%	18.95%	7.83%	12.50%	6.93%	11.63%
ONER	26.33%	18.87%	15.98%	9.47%	11.59%	3.41%	8.73%	3.49%
JRIP	21.74%	19.81%	10.74%	10.53%	6.38%	4.55%	4.52%	4.65%
RDR	20.05%	20.75%	9.64%	12.63%	5.51%	5.68%	3.31%	5.81%
J48	21.26%	21.70%	11.29%	13.68%	8.41%	7.95%	6.33%	6.98%
NBAYES	21.01%	19.81%	9.92%	10.53%	5.51%	4.55%	3.31%	3.49%
HPIPES	28.02%	18.87%	17.91%	11.58%	13.62%	5.68%	12.65%	4.65%
LWLS	22.22%	23.58%	11.85%	14.74%	7.83%	7.95%	6.33%	8.14%

Table 4.45: Predictive Quality of JM1-13C models on KC2 datasets

	KC2-520		KC2-23C		KC2-17C		KC2-13C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
CBR	19.32%	31.13%	9.64%	23.16%	6.67%	17.05%	5.12%	16.28%
TD	14.49%	31.13%	8.82%	17.89%	8.12%	5.68%	6.33%	4.65%
LR	21.50%	19.81%	10.47%	10.53%	5.80%	4.55%	3.31%	3.49%
LOC	21.50%	18.87%	10.47%	9.47%	5.80%	3.41%	3.61%	2.33%
GP	23.43%	18.87%	12.67%	9.47%	8.12%	3.41%	5.72%	3.49%
LBOOST	21.74%	19.81%	10.74%	10.53%	6.09%	3.41%	3.61%	2.33%
RBM	20.77%	20.75%	9.64%	11.58%	4.93%	5.68%	3.31%	5.81%
BAG	23.43%	19.81%	12.67%	11.58%	8.12%	5.68%	6.02%	4.65%
RSET	23.67%	22.64%	13.22%	13.68%	8.99%	7.95%	6.33%	6.98%
MCOST	20.29%	21.70%	9.09%	12.63%	4.35%	6.82%	3.31%	5.81%
ABOOST	21.74%	20.75%	11.02%	12.63%	6.67%	6.82%	5.12%	5.81%
DTABLE	24.40%	17.92%	13.77%	8.42%	9.28%	2.27%	6.33%	2.33%
ADT	21.50%	20.75%	10.47%	12.63%	6.09%	5.68%	3.92%	4.65%
SMO	19.08%	23.58%	7.99%	14.74%	3.48%	7.95%	2.11%	6.98%
IB1	19.32%	22.64%	8.54%	14.74%	4.35%	9.09%	2.71%	8.14%
IBK	18.60%	24.53%	7.71%	15.79%	3.48%	9.09%	2.41%	8.14%
PART	23.67%	19.81%	12.95%	11.58%	8.41%	5.68%	7.53%	4.65%
ONER	26.33%	18.87%	15.98%	9.47%	11.59%	3.41%	8.73%	3.49%
JRIP	24.15%	19.81%	13.50%	10.53%	8.99%	4.55%	6.33%	3.49%
RDR	24.40%	17.92%	13.77%	9.47%	9.28%	3.41%	7.83%	2.33%
J48	23.91%	19.81%	13.22%	10.53%	8.70%	4.55%	6.33%	3.49%
NBAYES	20.29%	19.81%	9.09%	10.53%	4.64%	4.55%	2.71%	3.49%
HPIPES	18.84%	25.47%	7.71%	16.84%	3.19%	11.36%	2.71%	10.47%
LWLS	20.05%	23.58%	9.09%	15.79%	4.93%	9.09%	2.71%	8.14%

Table 4.46: Predictive Quality of KC2-520 models on JM1 datasets

	JM1-8850		JM1-23C		JM1-20C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II
CBR	51.78%	23.59%	45.63%	17.14%	40.64%	13.00%
TD	30.70%	44.81%	29.03%	34.29%	27.90%	28.81%
LR	40.29%	30.82%	31.99%	22.99%	25.69%	18.03%
LOC	30.67%	37.29%	21.03%	29.70%	13.49%	24.07%
GP	30.29%	39.06%	20.73%	31.69%	13.79%	26.29%
LBOOST	45.19%	28.81%	38.24%	22.13%	32.81%	17.74%
RBM	38.34%	32.84%	30.02%	25.51%	24.17%	20.47%
BAG	39.24%	32.90%	30.86%	26.51%	24.50%	21.34%
RSET	39.08%	37.52%	31.82%	32.03%	27.29%	27.37%
MCOST	49.53%	26.73%	42.60%	21.06%	37.83%	17.24%
ABOOST	37.14%	40.66%	30.27%	35.75%	26.68%	31.97%
DTABLE	39.13%	31.12%	30.70%	23.72%	24.41%	18.25%
ADT	35.56%	34.50%	26.79%	27.57%	20.74%	22.41%
SMO	42.06%	30.82%	34.16%	22.99%	27.96%	17.96%
IB1	46.47%	32.31%	43.54%	26.98%	39.09%	25.07%
IBK	41.16%	32.25%	34.60%	22.06%	26.75%	18.68%
PART	36.42%	37.70%	28.30%	31.96%	22.66%	27.01%
ONER	34.85%	34.08%	25.76%	26.11%	18.54%	20.26%
JRIP	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
RDR	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
J48	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
NBAYES	36.17%	32.54%	27.28%	24.58%	20.27%	18.82%
HPIPES	24.86%	47.36%	16.12%	41.20%	11.13%	36.64%
LWLS	39.55%	34.62%	31.16%	28.04%	25.90%	22.49%

4.8.2 Predictive Performance of KC2 Models on JM1 Datasets

This subsection provides the results of predictive quality of the classification models built on the KC2-Fit datasets and evaluated on the JM1 datasets. Predictive performance of classifiers is presented in terms of the two misclassification error rates (Type I and Type II) in Tables 4.46 to 4.50. For each noise filtering level, JM1-Fit and JM1-Test were combined to create one evaluation set each.

Table 4.46: Predictive Quality of KC2-520 models on JM1 datasets, contd...

	JM1-17C		JM1-13C	
Methods	Type I	Type II	Type I	Type II
CBR	36.85%	10.22%	32.30%	6.68%
TD	26.03%	25.00%	26.48%	17.62%
LR	21.16%	13.47%	16.21%	7.47%
LOC	8.30%	18.42%	3.58%	10.24%
GP	9.12%	21.05%	5.04%	13.54%
LBOOST	29.18%	14.47%	24.85%	10.76%
RBM	19.93%	16.10%	15.57%	9.81%
BAG	20.06%	17.03%	16.11%	10.07%
RSET	24.59%	23.68%	21.27%	17.80%
MCOST	34.29%	13.70%	30.69%	9.03%
ABOOST	24.76%	29.18%	22.49%	25.69%
DTABLE	19.64%	14.01%	14.94%	7.47%
ADT	16.46%	17.80%	12.36%	11.46%
SMO	23.45%	13.39%	18.29%	7.55%
IB1	35.11%	25.77%	34.91%	17.88%
IBK	19.53%	14.94%	17.32%	7.90%
PART	18.48%	22.37%	14.77%	15.89%
ONER	13.22%	14.86%	7.51%	7.20%
JRIP	16.20%	17.88%	12.08%	11.46%
RDR	16.20%	17.88%	12.08%	11.46%
J48	16.20%	17.88%	12.08%	11.46%
NBAYES	15.17%	13.78%	9.94%	6.34%
HPIPES	8.45%	32.66%	6.27%	26.22%
LWLS	20.96%	18.27%	18.12%	11.81%

Table 4.47: Predictive Quality of KC2-260 models on JM1 datasets

	JM1-8850		JM1-23C		JM1-20C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II
CBR	55.37%	22.11%	49.81%	16.35%	45.31%	12.21%
TD	26.37%	44.16%	23.11%	30.63%	20.69%	21.91%
LR	41.83%	29.05%	33.82%	21.86%	27.94%	17.03%
LOC	32.30%	36.10%	22.88%	28.37%	15.50%	22.63%
GP	44.86%	30.82%	37.77%	25.45%	32.83%	20.98%
LBOOST	56.15%	25.31%	51.46%	20.73%	48.33%	18.18%
RBM	43.86%	29.40%	36.54%	22.26%	31.50%	17.82%
BAG	47.26%	28.63%	40.03%	23.26%	34.47%	18.75%
RSET	34.50%	35.74%	25.44%	28.31%	18.66%	22.92%
MCOST	43.36%	31.30%	36.00%	25.51%	30.89%	21.05%
ABOOST	44.84%	38.00%	39.80%	34.22%	36.47%	30.96%
DTABLE	42.58%	31.59%	35.59%	24.39%	30.11%	19.83%
ADT	38.57%	37.52%	30.93%	32.89%	26.07%	28.23%
SMO	40.57%	34.68%	33.49%	27.57%	28.06%	23.06%
IB1	44.62%	36.28%	40.95%	29.57%	35.75%	27.59%
IBK	41.78%	31.65%	38.94%	20.13%	28.29%	16.95%
PART	45.55%	32.31%	39.82%	28.04%	36.78%	24.78%
ONER	41.38%	38.53%	34.70%	34.55%	31.29%	31.32%
JRIP	45.64%	29.34%	38.55%	22.79%	33.69%	18.82%
RDR	48.95%	24.24%	41.85%	17.08%	36.34%	12.72%
J48	54.10%	33.67%	51.26%	30.63%	49.64%	29.24%
NBAYES	37.61%	32.90%	28.92%	24.92%	22.09%	19.04%
HPIPES	47.31%	25.31%	39.96%	17.54%	34.21%	12.57%
LWLS	38.28%	44.87%	34.93%	40.80%	32.08%	38.72%

Table 4.47: Predictive Quality of KC2-260 models on JM1 datasets, contd...

	JM1-17C		JM1-13C	
Methods	Type I	Type II	Type I	Type II
CBR	41.85%	9.75%	37.63%	6.60%
TD	19.25%	15.33%	17.34%	6.68%
LR	23.75%	12.93%	19.15%	7.38%
LOC	10.15%	17.18%	5.14%	9.11%
GP	29.36%	17.65%	26.19%	12.50%
LBOOST	46.22%	16.18%	42.85%	14.32%
RBM	27.98%	14.40%	23.95%	10.24%
BAG	30.69%	14.78%	27.46%	9.03%
RSET	14.18%	18.42%	10.16%	11.37%
MCOST	27.53%	17.49%	23.93%	12.24%
ABOOST	34.44%	28.33%	31.70%	25.17%
DTABLE	25.97%	16.49%	21.66%	11.20%
ADT	23.15%	24.38%	20.63%	18.84%
SMO	23.86%	19.20%	18.84%	14.15%
IB1	34.01%	26.70%	31.06%	20.92%
IBK	21.97%	13.54%	19.79%	6.08%
PART	34.74%	22.76%	32.65%	19.79%
ONER	29.25%	28.79%	27.20%	23.87%
JRIP	30.09%	15.33%	26.09%	11.81%
RDR	32.30%	9.13%	27.53%	5.21%
J48	48.67%	28.02%	46.95%	28.04%
NBAYES	16.93%	13.85%	11.46%	6.86%
HPIPES	29.81%	9.13%	24.71%	4.34%
LWLS	30.81%	35.60%	28.72%	31.42%

Table 4.48: Predictive Quality of KC2-23C models on JM1 datasets

	JM1-8850		JM1-23C		JM1-20C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II
CBR	53.08%	23.89%	47.18%	17.81%	42.46%	13.94%
TD	24.43%	47.84%	21.30%	33.36%	18.50%	23.56%
LR	32.70%	37.76%	23.64%	30.90%	17.60%	25.72%
LOC	30.67%	37.29%	21.03%	29.70%	13.49%	24.07%
GP	24.81%	45.29%	15.05%	38.67%	8.98%	33.76%
LBOOST	35.78%	38.17%	27.12%	33.49%	21.37%	28.74%
RBM	45.76%	29.76%	38.72%	23.59%	33.72%	19.61%
BAG	34.23%	36.87%	25.17%	30.10%	18.54%	24.78%
RSET	42.01%	29.46%	33.96%	21.99%	27.70%	16.81%
MCOST	46.54%	26.50%	39.09%	19.20%	33.23%	14.22%
ABOOST	41.13%	33.85%	33.25%	27.84%	27.90%	22.99%
DTABLE	30.98%	38.00%	21.51%	30.56%	14.57%	25.07%
ADT	38.11%	34.85%	29.62%	28.50%	23.40%	23.28%
SMO	39.31%	30.94%	30.85%	22.72%	24.11%	17.10%
IB1	47.02%	33.31%	38.42%	27.84%	37.90%	25.57%
IBK	37.47%	33.55%	30.94%	22.66%	22.26%	18.61%
PART	39.76%	35.57%	32.74%	29.90%	28.18%	26.08%
ONER	41.50%	30.59%	33.41%	23.26%	27.22%	17.89%
JRIP	41.50%	30.59%	33.41%	23.26%	27.22%	17.89%
RDR	40.28%	31.59%	32.02%	24.12%	25.76%	18.82%
J48	39.76%	35.57%	32.74%	29.90%	28.18%	26.08%
NBAYES	36.49%	33.02%	27.68%	25.18%	20.88%	19.32%
HPIPES	37.76%	31.54%	29.16%	23.79%	22.59%	18.18%
LWLS	35.17%	36.34%	27.03%	29.63%	19.01%	23.85%

Table 4.48: Predictive Quality of KC2-23C models on JM1 datasets, contd...

	JM1-17C		JM1-13C	
Methods	Type I	Type II	Type I	Type II
CBR	38.86%	11.07%	34.56%	7.21%
TD	17.08%	16.56%	14.91%	8.68%
LR	13.48%	20.74%	9.75%	13.89%
LOC	8.30%	18.42%	3.58%	10.24%
GP	5.62%	28.95%	3.00%	21.53%
LBOOST	18.15%	23.99%	15.80%	16.49%
RBM	30.39%	15.87%	26.35%	10.59%
BAG	14.42%	20.20%	10.92%	12.67%
RSET	22.90%	12.38%	17.98%	6.16%
MCOST	28.73%	10.37%	23.78%	5.03%
ABOOST	24.46%	19.27%	21.13%	13.37%
DTABLE	9.94%	19.89%	5.76%	12.07%
ADT	19.23%	18.65%	15.59%	11.72%
SMO	19.14%	12.38%	13.60%	5.99%
IB1	33.75%	25.62%	33.57%	18.14%
IBK	13.90%	15.56%	11.05%	6.77%
PART	25.22%	22.29%	22.03%	17.01%
ONER	22.51%	13.62%	17.59%	7.47%
JRIP	22.51%	13.62%	17.59%	7.47%
RDR	20.96%	14.47%	16.09%	8.07%
J48	25.22%	22.29%	22.03%	17.01%
NBAYES	15.97%	14.16%	10.90%	6.60%
HPIPES	17.88%	13.54%	12.80%	7.03%
LWLS	15.13%	19.97%	10.62%	11.55%

Table 4.49: Predictive Quality of KC2-17C models on JM1 datasets

	JM1-8850		JM1-23C		JM1-20C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II
CBR	50.47%	23.53%	43.57%	16.15%	38.06%	11.42%
TD	16.95%	55.90%	12.52%	40.33%	10.15%	28.16%
LR	35.38%	34.20%	26.50%	26.91%	19.99%	21.41%
LOC	29.46%	38.47%	19.65%	31.03%	12.13%	25.43%
GP	34.11%	35.98%	25.17%	29.04%	19.01%	23.85%
LBOOST	41.74%	30.17%	33.68%	22.86%	27.43%	17.60%
RBM	42.62%	28.69%	34.62%	21.26%	28.45%	16.38%
BAG	41.13%	30.23%	32.91%	22.66%	26.45%	17.10%
RSET	29.46%	38.47%	19.65%	31.03%	12.13%	25.43%
MCOST	39.12%	31.30%	30.67%	23.85%	24.15%	18.53%
ABOOST	39.61%	31.00%	31.20%	23.72%	24.64%	18.53%
DTABLE	32.79%	35.51%	23.48%	28.50%	16.44%	23.42%
ADT	33.78%	34.14%	24.55%	26.58%	17.39%	20.83%
SMO	41.73%	30.29%	33.65%	22.19%	27.45%	17.10%
IB1	43.59%	28.87%	38.74%	19.67%	32.36%	14.22%
IBK	38.45%	33.43%	34.86%	22.06%	25.02%	18.46%
PART	37.05%	33.79%	28.43%	26.51%	21.96%	21.26%
ONER	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
JRIP	41.60%	29.99%	33.47%	22.52%	27.15%	17.39%
RDR	39.70%	29.99%	31.29%	22.46%	24.59%	16.81%
J48	39.86%	31.59%	31.63%	24.72%	25.51%	19.61%
NBAYES	36.05%	32.96%	27.14%	24.98%	20.11%	19.18%
HPIPES	38.43%	31.65%	29.84%	23.79%	23.26%	18.25%
LWLS	47.24%	25.13%	39.34%	18.07%	32.34%	13.36%

Table 4.49: Predictive Quality of KC2-17C models on JM1 datasets, contd...

	JM1-17C		JM1-13C	
Methods	Type I	Type II	Type I	Type II
CBR	33.88%	7.97%	28.92%	3.56%
TD	8.30%	18.42%	5.14%	9.11%
LR	15.41%	16.49%	11.19%	9.64%
LOC	7.10%	19.81%	2.84%	11.28%
GP	15.07%	19.66%	11.34%	13.28%
LBOOST	22.72%	13.24%	17.79%	7.20%
RBM	23.80%	12.07%	18.93%	5.82%
BAG	21.50%	12.31%	16.25%	5.73%
RSET	7.10%	19.81%	2.84%	11.28%
MCOST	19.34%	13.78%	14.48%	7.03%
ABOOST	19.81%	13.93%	14.69%	7.64%
DTABLE	11.91%	18.50%	8.13%	11.37%
ADT	12.32%	15.79%	7.74%	8.33%
SMO	22.90%	12.54%	17.73%	6.51%
IB1	25.66%	11.69%	22.81%	5.38%
IBK	18.78%	15.48%	15.57%	6.77%
PART	17.51%	17.03%	13.23%	10.59%
ONER	16.20%	17.88%	12.08%	11.46%
JRIP	22.42%	12.85%	17.53%	6.68%
RDR	19.46%	11.92%	13.91%	5.30%
J48	21.22%	15.79%	16.93%	9.90%
NBAYES	14.96%	14.09%	9.44%	7.03%
HPIPES	18.39%	13.47%	13.29%	6.51%
LWLS	27.64%	9.37%	22.94%	4.25%

Table 4.50: Predictive Quality of KC2-13C models on JM1 datasets

	JM1-8850		JM1-23C		JM1-20C	
Methods	Type I	Type II	Type I	Type II	Type I	Type II
CBR	48.28%	25.67%	41.28%	18.54%	35.68%	13.87%
TD	17.58%	56.31%	12.93%	43.59%	11.01%	33.12%
LR	50.83%	24.90%	44.45%	18.47%	39.96%	15.09%
LOC	32.30%	36.10%	22.88%	28.37%	15.50%	22.63%
GP	31.96%	37.46%	22.69%	30.37%	16.29%	25.00%
LBOOST	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
RBM	40.33%	31.65%	32.20%	24.39%	26.44%	19.40%
BAG	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
RSET	32.85%	36.04%	23.56%	28.37%	16.65%	22.77%
MCOST	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
ABOOST	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
DTABLE	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
ADT	34.82%	35.51%	25.95%	28.44%	19.81%	23.13%
SMO	41.25%	30.35%	33.09%	22.26%	26.75%	17.10%
IB1	36.76%	33.79%	29.70%	25.51%	22.94%	21.19%
IBK	38.53%	34.14%	37.61%	21.99%	25.60%	19.68%
PART	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
ONER	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
JRIP	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
RDR	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
J48	35.36%	34.68%	26.57%	27.71%	20.50%	22.49%
NBAYES	42.18%	29.10%	34.19%	21.00%	27.99%	15.66%
HPIPES	38.70%	30.59%	30.16%	22.72%	23.45%	16.95%
LWLS	34.58%	35.21%	25.31%	28.37%	19.27%	23.06%

Table 4.50: Predictive Quality of KC2-13C models on JM1 datasets, contd...

	JM1-17C		JM1-13C	
Methods	Type I	Type II	Type I	Type II
CBR	31.57%	10.45%	26.81%	5.47%
TD	8.90%	26.16%	7.39%	15.97%
LR	36.82%	12.07%	32.75%	8.25%
LOC	10.15%	17.18%	5.14%	9.11%
GP	11.76%	19.74%	7.78%	12.50%
LBOOST	16.20%	17.88%	12.08%	11.46%
RBM	22.30%	15.17%	17.86%	9.38%
BAG	16.20%	17.88%	12.08%	11.46%
RSET	11.99%	17.65%	7.45%	9.81%
MCOST	16.20%	17.88%	12.08%	11.46%
ABOOST	16.20%	17.88%	12.08%	11.46%
DTABLE	16.20%	17.88%	12.08%	11.46%
ADT	15.47%	18.42%	11.31%	11.81%
SMO	22.06%	12.62%	16.83%	6.51%
IB1	17.13%	17.88%	13.35%	10.07%
IBK	19.03%	17.49%	16.48%	8.33%
PART	16.20%	17.88%	12.08%	11.46%
ONER	16.20%	17.88%	12.08%	11.46%
JRIP	16.20%	17.88%	12.08%	11.46%
RDR	16.20%	17.88%	12.08%	11.46%
J48	16.20%	17.88%	12.08%	11.46%
NBAYES	23.28%	10.91%	17.61%	5.47%
HPIPES	18.41%	12.00%	13.12%	5.30%
LWLS	14.61%	18.27%	10.76%	11.63%

Table 4.51: ECM Results for JM1-8850 Models Applied to KC2 datasets, c=10

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	0.8173	0.6489	0.5543	0.5072
TD	0.6442	0.4026	0.3695	0.3852
LR	0.7038	0.4467	0.2818	0.2321
LOC	0.5692	0.2934	0.1316	0.0909
GP	0.5635	0.2870	0.1247	0.1005
LBOOST	0.6423	0.4014	0.2494	0.2297
RBM	0.6788	0.4450	0.2748	0.2656
BAG	0.6827	0.4362	0.3372	0.3134
RSET	0.7558	0.5358	0.3764	0.3206
MCOST	0.8538	0.6917	0.5751	0.5622
ABOOST	0.9865	0.9007	0.8268	0.7847
DTABLE	0.7096	0.4754	0.3095	0.2799
ADT	0.6385	0.3997	0.2517	0.2225
SMO	0.6481	0.3817	0.2079	0.1842
IB1	1.2250	1.1043	1.0600	1.0861
IBK	0.7269	0.5314	0.3764	0.3541
PART	0.8538	0.6508	0.5520	0.5359
ONER	0.7019	0.4754	0.3095	0.3014
JRIP	0.6788	0.4710	0.3048	0.2823
RDR	0.6519	0.4610	0.3441	0.3158
J48	0.7750	0.6139	0.4919	0.4713
NBAYES	0.5538	0.2756	0.1155	0.0670
HPIPES	1.0269	1.0101	0.8915	0.8971
LWLS	0.7635	0.5815	0.4527	0.4569

4.8.3 NECM Results for JM1 Models Applied to the KC2 Datasets, c=10

In this subsection, the predictive performance of the classification models built on the JM1 datasets and evaluated on the on the KC2 datasets are presented in Tables 4.51 to 4.56 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=10.

Table 4.52: ECM Results for JM1-4425 Models Applied to KC2 datasets, c=10

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	0.8346	0.6878	0.6212	0.5813
TD	0.5962	0.3938	0.3279	0.3301
LR	0.6077	0.3365	0.1594	0.1220
LOC	0.5692	0.2934	0.1316	0.0909
GP	0.5500	0.2932	0.1386	0.1292
LBOOST	0.5904	0.3433	0.1871	0.1531
RBM	0.5654	0.2888	0.1293	0.1172
BAG	0.6288	0.4126	0.3118	0.2799
RSET	0.6346	0.3927	0.2263	0.1722
MCOST	0.5942	0.3719	0.2425	0.2321
ABOOST	0.8904	0.7600	0.6744	0.6364
DTABLE	0.6635	0.4060	0.2540	0.2201
ADT	0.6577	0.4165	0.2679	0.2488
SMO	0.6846	0.4251	0.2540	0.2105
IB1	0.9558	0.8420	0.7852	0.7775
IBK	0.7173	0.4664	0.2956	0.2679
PART	0.6788	0.4870	0.4134	0.3828
ONER	0.6712	0.4126	0.2679	0.2321
JRIP	0.5808	0.3520	0.1963	0.1866
RDR	0.6769	0.4804	0.3326	0.3038
J48	0.6615	0.4012	0.2517	0.2321
NBAYES	0.5904	0.3168	0.1594	0.1100
HPIPES	1.0615	0.9550	0.8268	0.8230
LWLS	0.6692	0.4579	0.2910	0.2632

Table 4.53: ECM Results for JM1-23C Models Applied to KC2 datasets, c=10

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	0.7981	0.6246	0.5312	0.4880
TD	0.6962	0.3067	0.2702	0.2392
LR	0.5596	0.2824	0.1201	0.1029
LOC	0.5692	0.2934	0.1316	0.0909
GP	0.6173	0.3498	0.1917	0.1483
LBOOST	0.5981	0.3284	0.1686	0.1459
RBM	0.6385	0.3929	0.2425	0.2392
BAG	0.7038	0.4754	0.3326	0.2727
RSET	0.6788	0.4297	0.2841	0.2584
MCOST	0.7731	0.5620	0.4088	0.3589
ABOOST	0.8096	0.6095	0.4896	0.4426
DTABLE	0.6365	0.3842	0.2379	0.1818
ADT	0.6519	0.3890	0.2333	0.2129
SMO	0.6212	0.3535	0.1801	0.1364
IB1	0.8635	0.7029	0.5797	0.5598
IBK	0.6288	0.3623	0.1848	0.1531
PART	0.7231	0.5012	0.3649	0.3206
ONER	0.5673	0.3128	0.1524	0.1124
JRIP	0.7577	0.5384	0.4042	0.4019
RDR	0.6212	0.3537	0.1755	0.1675
J48	0.6808	0.4712	0.3557	0.3301
NBAYES	0.5692	0.2932	0.1339	0.0718
HPIPES	0.6577	0.3927	0.2379	0.1890
LWLS	0.6442	0.4104	0.2633	0.2273

Table 4.54: ECM Results for JM1-20C Models Applied to KC2 datasets, c=10

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	0.7250	0.5358	0.4365	0.4115
TD	0.8038	0.4124	0.2379	0.2273
LR	0.5673	0.2934	0.1316	0.1124
LOC	0.5635	0.2870	0.1247	0.1053
GP	0.5481	0.2912	0.1293	0.1148
LBOOST	0.6904	0.4362	0.2679	0.2608
RBM	0.6212	0.3516	0.1986	0.1699
BAG	0.7327	0.5056	0.3395	0.2895
RSET	0.6115	0.3406	0.1871	0.1531
MCOST	0.7692	0.5489	0.3903	0.3660
ABOOST	0.6635	0.4598	0.3025	0.2536
DTABLE	0.5904	0.3474	0.2171	0.1842
ADT	0.6635	0.4297	0.2610	0.2273
SMO	0.6212	0.3537	0.1755	0.1388
IB1	0.7038	0.5097	0.3811	0.3541
IBK	0.6462	0.4036	0.2356	0.1986
PART	0.7865	0.5788	0.4319	0.4139
ONER	0.5808	0.3065	0.1455	0.1268
JRIP	0.7231	0.4988	0.3349	0.2871
RDR	0.5788	0.3498	0.1917	0.1627
J48	0.7500	0.5336	0.3949	0.3517
NBAYES	0.6019	0.3299	0.1732	0.1172
HPIPES	0.6981	0.4533	0.2887	0.2656
LWLS	0.6308	0.4189	0.2771	0.2225

Table 4.55: ECM Results for JM1-17C Models Applied to KC2 datasets, c=10

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	0.7692	0.5401	0.4134	0.3612
TD	0.8058	0.4146	0.1547	0.0933
LR	0.5904	0.3172	0.1339	0.1124
LOC	0.5635	0.2870	0.1247	0.1053
GP	0.5788	0.3041	0.1432	0.1053
LBOOST	0.5769	0.3043	0.1455	0.1292
RBM	0.6712	0.3516	0.2356	0.1818
BAG	0.5827	0.3323	0.1501	0.1124
RSET	0.5962	0.3691	0.2148	0.1890
MCOST	0.6635	0.4277	0.2540	0.2201
ABOOST	0.6865	0.4513	0.2864	0.2560
DTABLE	0.7038	0.4515	0.2841	0.2488
ADT	0.5923	0.3236	0.1455	0.1100
SMO	0.6308	0.3647	0.1871	0.1531
IB1	0.7135	0.5099	0.3464	0.3206
IBK	0.6442	0.3800	0.2055	0.1722
PART	0.7135	0.4817	0.3164	0.2943
ONER	0.5942	0.3218	0.1617	0.1411
JRIP	0.5769	0.3019	0.1432	0.1316
RDR	0.5827	0.3365	0.1594	0.1459
J48	0.6115	0.3713	0.2286	0.1938
NBAYES	0.5712	0.2953	0.1363	0.0981
HPIPES	0.6077	0.3804	0.2240	0.1962
LWLS	0.6577	0.3973	0.2240	0.2177

Table 4.56: ECM Results for JM1-13C Models Applied to KC2 datasets, c=10

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	0.7885	0.5530	0.3995	0.3756
TD	0.7500	0.4382	0.1801	0.1459
LR	0.5750	0.2997	0.1386	0.0981
LOC	0.5558	0.2781	0.1155	0.0766
GP	0.5712	0.2956	0.1339	0.1172
LBOOST	0.5769	0.3019	0.1178	0.0766
RBM	0.5885	0.3148	0.1547	0.1459
BAG	0.5904	0.3389	0.1801	0.1435
RSET	0.6500	0.3866	0.2333	0.1938
MCOST	0.6038	0.3321	0.1732	0.1459
ABOOST	0.5962	0.3474	0.1917	0.1603
DTABLE	0.5596	0.2827	0.1201	0.0981
ADT	0.5942	0.3430	0.1640	0.1268
SMO	0.6327	0.3667	0.1894	0.1603
IB1	0.6154	0.3710	0.2194	0.1890
IBK	0.6481	0.3861	0.2125	0.1866
PART	0.5923	0.3411	0.1824	0.1555
ONER	0.5942	0.3218	0.1617	0.1411
JRIP	0.5962	0.3238	0.1640	0.1220
RDR	0.5596	0.3043	0.1432	0.1100
J48	0.5942	0.3216	0.1617	0.1220
NBAYES	0.5654	0.2888	0.1293	0.0933
HPIPES	0.6692	0.4078	0.2564	0.2368
LWLS	0.6404	0.3971	0.2240	0.1890

Table 4.57: ECM Results for JM1-8850 Models Applied to KC2 datasets, c=20

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	1.4519	1.1686	0.9931	0.9139
TD	1.0673	0.6191	0.4850	0.5287
LR	1.2615	0.8365	0.5358	0.4474
LOC	0.9538	0.4883	0.2009	0.1388
GP	0.9288	0.4603	0.1709	0.1483
LBOOST	1.1231	0.7263	0.4573	0.4211
RBM	1.1788	0.7915	0.4827	0.4809
BAG	1.2212	0.7827	0.6143	0.5766
RSET	1.3519	0.9906	0.6998	0.6077
MCOST	1.5654	1.2981	1.0831	1.0646
ABOOST	1.7558	1.6153	1.4734	1.4067
DTABLE	1.2288	0.8436	0.5404	0.4952
ADT	1.0808	0.6812	0.4134	0.3660
SMO	1.1481	0.7066	0.3926	0.3517
IB1	2.2250	2.0139	1.9376	1.9952
IBK	1.3038	0.9862	0.6998	0.6651
PART	1.5462	1.1923	1.0139	0.9904
ONER	1.2212	0.8436	0.5404	0.5407
JRIP	1.1788	0.8392	0.5358	0.4976
RDR	1.0558	0.7425	0.5289	0.4833
J48	1.4096	1.1336	0.9076	0.8780
NBAYES	0.9577	0.4922	0.2079	0.1148
HPIPES	1.5269	1.5082	1.2610	1.2560
LWLS	1.3596	1.0579	0.8222	0.8397

4.8.4 NECM Results for JM1 Models Applied to the KC2 Datasets, c=20

In this subsection, the predictive performance of the classification models built on the JM1 datasets and evaluated on the on the KC2 datasets are presented in Tables 4.57 to 4.62 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=20.

Table 4.58: ECM Results for JM1-4425 Models Applied to KC2 datasets, c=20

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	1.4884	1.2509	1.1293	1.0598
TD	1.0000	0.6104	0.4434	0.4498
LR	1.0500	0.5963	0.2748	0.2177
LOC	0.9538	0.4883	0.2009	0.1388
GP	0.9346	0.5097	0.2309	0.2249
LBOOST	0.9942	0.5815	0.3025	0.2488
RBM	0.9692	0.5053	0.2217	0.2129
BAG	1.0712	0.7158	0.5427	0.4952
RSET	1.1154	0.7175	0.4111	0.3158
MCOST	0.9596	0.5885	0.3580	0.3517
ABOOST	1.5442	1.3231	1.1594	1.0909
DTABLE	1.1442	0.7092	0.4388	0.3876
ADT	1.1577	0.7630	0.4988	0.4641
SMO	1.2231	0.7932	0.4850	0.4019
IB1	1.7058	1.5134	1.4088	1.3995
IBK	1.2750	0.8562	0.5497	0.5072
PART	1.1212	0.7902	0.6674	0.6220
ONER	1.1519	0.7158	0.4527	0.3995
JRIP	0.9654	0.5902	0.3118	0.3062
RDR	1.1000	0.7836	0.5173	0.4713
J48	1.1808	0.7477	0.4827	0.4474
NBAYES	1.0327	0.5767	0.2979	0.2057
HPIPES	1.6769	1.4964	1.2425	1.2297
LWLS	1.1692	0.8260	0.5219	0.4785

Table 4.59: ECM Results for JM1-23C Models Applied to KC2 datasets, c=20

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	1.4519	1.1660	0.9931	0.9187
TD	1.2154	0.4800	0.3395	0.2871
LR	0.9442	0.4773	0.1894	0.1746
LOC	0.9538	0.4883	0.2009	0.1388
GP	1.0404	0.5880	0.3072	0.2440
LBOOST	0.9827	0.5233	0.2379	0.2177
RBM	1.1000	0.6961	0.4273	0.4306
BAG	1.2231	0.8436	0.5866	0.4880
RSET	1.1981	0.7762	0.5150	0.4737
MCOST	1.3692	1.0168	0.7321	0.6459
ABOOST	1.4635	1.1292	0.9053	0.8254
DTABLE	1.1173	0.6874	0.4226	0.3254
ADT	1.0942	0.6489	0.3718	0.3565
SMO	1.1019	0.6567	0.3418	0.2560
IB1	1.5558	1.2876	1.0647	1.0383
IBK	1.1096	0.6655	0.3464	0.2967
PART	1.2808	0.9127	0.6651	0.5837
ONER	0.9519	0.5294	0.2448	0.1842
JRIP	1.3154	0.9499	0.7044	0.7129
RDR	1.0827	0.6353	0.3141	0.3110
J48	1.1615	0.8177	0.6097	0.5694
NBAYES	0.9731	0.5097	0.2263	0.1196
HPIPES	1.1577	0.7175	0.4457	0.3565
LWLS	1.1058	0.7136	0.4480	0.3947

Table 4.60: ECM Results for JM1-20C Models Applied to KC2 datasets, c=20

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	1.3019	0.9905	0.8060	0.7703
TD	1.4962	0.7372	0.3764	0.3708
LR	0.9519	0.4883	0.2009	0.1842
LOC	0.9288	0.4603	0.1709	0.1531
GP	0.9135	0.4861	0.1986	0.1866
LBOOST	1.2096	0.7827	0.4758	0.4761
RBM	1.0827	0.6331	0.3603	0.3134
BAG	1.2904	0.9171	0.6166	0.5287
RSET	1.0731	0.6222	0.3487	0.2967
MCOST	1.3654	1.0037	0.7136	0.6770
ABOOST	1.1827	0.8497	0.5566	0.4689
DTABLE	1.0135	0.6073	0.3788	0.3278
ADT	1.1635	0.7762	0.4688	0.4187
SMO	1.0827	0.6353	0.3141	0.2584
IB1	1.2615	0.9429	0.7044	0.6651
IBK	1.1269	0.7285	0.4203	0.3660
PART	1.4404	1.0986	0.8245	0.7967
ONER	0.9654	0.5014	0.2148	0.1986
JRIP	1.3000	0.9319	0.6351	0.5502
RDR	0.9635	0.5880	0.3072	0.2584
J48	1.3462	0.9884	0.7413	0.6627
NBAYES	1.0442	0.5898	0.3118	0.2129
HPIPES	1.2558	0.8431	0.5427	0.5048
LWLS	1.0923	0.7438	0.4850	0.3900

Table 4.61: ECM Results for JM1-17C Models Applied to KC2 datasets, c=20

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	1.3846	0.9949	0.7598	0.6722
TD	1.4981	0.7394	0.2471	0.1172
LR	0.9942	0.5338	0.2032	0.1842
LOC	0.9288	0.4603	0.1709	0.1531
GP	0.9827	0.5207	0.2356	0.1770
LBOOST	0.9615	0.4992	0.2148	0.2010
RBM	1.1904	0.6331	0.4434	0.3493
BAG	0.9865	0.5705	0.2425	0.1842
RSET	1.0192	0.6506	0.3764	0.3325
MCOST	1.1442	0.7525	0.4388	0.3876
ABOOST	1.2058	0.8195	0.5173	0.4713
DTABLE	1.2231	0.7981	0.4919	0.4402
ADT	1.0154	0.5618	0.2379	0.1818
SMO	1.0923	0.6462	0.3256	0.2727
IB1	1.2519	0.9214	0.6236	0.5837
IBK	1.1058	0.6615	0.3441	0.2919
PART	1.2519	0.8716	0.5704	0.5335
ONER	0.9788	0.5167	0.2309	0.2129
JRIP	0.9808	0.5185	0.2356	0.2273
RDR	1.0058	0.5963	0.2748	0.2656
J48	1.0538	0.6528	0.3903	0.3373
NBAYES	0.9750	0.5119	0.2286	0.1699
HPIPES	0.9923	0.6187	0.3395	0.2919
LWLS	1.1385	0.7005	0.3857	0.3852

Table 4.62: ECM Results for JM1-13C Models Applied to KC2 datasets, c=20

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	1.4231	1.0295	0.7459	0.7105
TD	1.3846	0.8064	0.2956	0.2416
LR	0.9788	0.5163	0.2309	0.1699
LOC	0.9404	0.4730	0.1848	0.1244
GP	0.9558	0.4905	0.2032	0.1890
LBOOST	0.9808	0.5185	0.1871	0.1244
RBM	1.0115	0.5530	0.2702	0.2656
BAG	0.9942	0.5771	0.2956	0.2392
RSET	1.1115	0.6681	0.3949	0.3373
MCOST	1.0462	0.5920	0.3118	0.2656
ABOOST	1.0192	0.6073	0.3303	0.2799
DTABLE	0.9250	0.4559	0.1663	0.1459
ADT	1.0173	0.6029	0.2794	0.2225
SMO	1.1135	0.6698	0.3510	0.3038
IB1	1.0769	0.6742	0.4042	0.3565
IBK	1.1481	0.7110	0.3972	0.3541
PART	0.9962	0.5793	0.2979	0.2512
ONER	0.9788	0.5167	0.2309	0.2129
JRIP	1.0000	0.5404	0.2564	0.1938
RDR	0.9250	0.4992	0.2125	0.1579
J48	0.9981	0.5382	0.2540	0.1938
NBAYES	0.9692	0.5053	0.2217	0.1651
HPIPES	1.1885	0.7543	0.4873	0.4522
LWLS	1.1212	0.7219	0.4088	0.3565

Table 4.63: ECM Results for JM1-8850 Models Applied to KC2 datasets, c=30

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	2.0865	1.6884	1.4319	1.3206
TD	1.4904	0.8357	0.6005	0.6722
LR	1.8192	1.2264	0.7898	0.6627
LOC	1.3385	0.6832	0.2702	0.1866
GP	1.2942	0.6335	0.2171	0.1962
LBOOST	1.6038	1.0511	0.6651	0.6124
RBM	1.6788	1.1380	0.6905	0.6962
BAG	1.7596	1.1292	0.8915	0.8397
RSET	1.9481	1.4453	1.0231	0.8947
MCOST	2.2769	1.9045	1.5912	1.5670
ABOOST	2.5250	2.3300	2.1201	2.0287
DTABLE	1.7481	1.2117	0.7714	0.7105
ADT	1.5231	0.9628	0.5751	0.5096
SMO	1.6481	1.0315	0.5774	0.5191
IB1	3.2250	2.9235	2.8152	2.9043
IBK	1.8808	1.4410	1.0231	0.9761
PART	2.2385	1.7337	1.4758	1.4450
ONER	1.7404	1.2117	0.7714	0.7799
JRIP	1.6788	1.2073	0.7667	0.7129
RDR	1.4596	1.0240	0.7136	0.6507
J48	2.0442	1.6534	1.3233	1.2847
NBAYES	1.3615	0.7088	0.3002	0.1627
HPIPES	2.0269	2.0063	1.6305	1.6148
LWLS	1.9558	1.5344	1.1917	1.2225

4.8.5 NECM Results for JM1 Models Applied to the KC2 Datasets, c=30

In this subsection, the predictive performance of the classification models built on the JM1 datasets and evaluated on the on the KC2 datasets are presented in Tables 4.63 to 4.68 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=30.

Table 4.64: ECM Results for JM1-4425 Models Applied to KC2 datasets, c=30

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	2.1423	1.8139	1.6374	1.5383
TD	1.4038	0.8269	0.5589	0.5694
LR	1.4923	0.8562	0.3903	0.3134
LOC	1.3385	0.6832	0.2702	0.1866
GP	1.3192	0.7263	0.3233	0.3206
LBOOST	1.3981	0.8197	0.4180	0.3445
RBM	1.3731	0.7219	0.3141	0.3086
BAG	1.5135	1.0190	0.7737	0.7105
RSET	1.5962	1.0424	0.5958	0.4593
MCOST	1.3250	0.8051	0.4734	0.4713
ABOOST	2.1981	1.8862	1.6443	1.5455
DTABLE	1.6250	1.0124	0.6236	0.5550
ADT	1.6577	1.1096	0.7298	0.6794
SMO	1.7615	1.1614	0.7159	0.5933
IB1	2.4558	2.1848	2.0323	2.0215
IBK	1.8327	1.2461	0.8037	0.7464
PART	1.5635	1.0934	0.9215	0.8612
ONER	1.6327	1.0190	0.6374	0.5670
JRIP	1.3500	0.8285	0.4273	0.4258
RDR	1.5231	1.0868	0.7021	0.6388
J48	1.7000	1.0942	0.7136	0.6627
NBAYES	1.4750	0.8365	0.4365	0.3014
HPIPES	2.2923	2.0378	1.6582	1.6364
LWLS	1.6692	1.1942	0.7529	0.6938

Table 4.65: ECM Results for JM1-23C Models Applied to KC2 datasets, c=30

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	2.1057	1.7074	1.4550	1.3493
TD	1.7346	0.6532	0.4088	0.3349
LR	1.3288	0.6723	0.2587	0.2464
LOC	1.3385	0.6832	0.2702	0.1866
GP	1.4635	0.8263	0.4226	0.3397
LBOOST	1.3673	0.7182	0.3072	0.2895
RBM	1.5615	0.9993	0.6120	0.6220
BAG	1.7423	1.2117	0.8406	0.7033
RSET	1.7173	1.1227	0.7460	0.6890
MCOST	1.9654	1.4716	1.0554	0.9330
ABOOST	2.1173	1.6490	1.3210	1.2081
DTABLE	1.5981	0.9906	0.6074	0.4689
ADT	1.5365	0.9087	0.5104	0.5000
SMO	1.5827	0.9599	0.5035	0.3756
IB1	2.2481	1.8724	1.5497	1.5167
IBK	1.5904	0.9687	0.5081	0.4402
PART	1.8385	1.3242	0.9654	0.8469
ONER	1.3365	0.7460	0.3372	0.2560
JRIP	1.8731	1.3614	1.0046	1.0239
RDR	1.5442	0.9168	0.4527	0.4545
J48	1.6423	1.1643	0.8637	0.8086
NBAYES	1.3769	0.7263	0.3187	0.1675
HPIPES	1.6577	1.0424	0.6536	0.5239
LWLS	1.5673	1.0168	0.6328	0.5622

Table 4.66: ECM Results for JM1-20C Models Applied to KC2 datasets, c=30

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	1.8789	1.4453	1.1755	1.1292
TD	2.1885	1.0621	0.5150	0.5144
LR	1.3365	0.6832	0.2702	0.2560
LOC	1.2942	0.6335	0.2171	0.2010
GP	1.2788	0.6810	0.2679	0.2584
LBOOST	1.7288	1.1292	0.6836	0.6914
RBM	1.5442	0.9146	0.5219	0.4569
BAG	1.8481	1.3285	0.8938	0.7679
RSET	1.5346	0.9037	0.5104	0.4402
MCOST	1.9615	1.4585	1.0370	0.9880
ABOOST	1.7019	1.2395	0.8106	0.6842
DTABLE	1.4365	0.8672	0.5404	0.4713
ADT	1.6635	1.1227	0.6767	0.6100
SMO	1.5442	0.9168	0.4527	0.3780
IB1	1.8192	1.3760	1.0277	0.9761
IBK	1.6077	1.0533	0.6051	0.5335
PART	2.0942	1.6184	1.2171	1.1794
ONER	1.3500	0.6963	0.2841	0.2703
JRIP	1.8769	1.3651	0.9353	0.8134
RDR	1.3481	0.8263	0.4226	0.3541
J48	1.9423	1.4432	1.0878	0.9737
NBAYES	1.4865	0.8497	0.4503	0.3086
HPIPES	1.8135	1.2329	0.7968	0.7440
LWLS	1.5538	1.0687	0.6928	0.5574

Table 4.67: ECM Results for JM1-17C Models Applied to KC2 datasets, c=30

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	2.0000	1.4497	1.1062	0.9832
TD	2.1904	1.0643	0.3395	0.1411
LR	1.3981	0.7504	0.2725	0.2560
LOC	1.2942	0.6335	0.2171	0.2010
GP	1.3865	0.7372	0.3279	0.2488
LBOOST	1.3462	0.6941	0.2841	0.2727
RBM	1.7096	0.9146	0.6513	0.5167
BAG	1.3904	0.8088	0.3349	0.2560
RSET	1.4423	0.9321	0.5381	0.4761
MCOST	1.6250	1.0774	0.6236	0.5550
ABOOST	1.7250	1.1877	0.7483	0.6866
DTABLE	1.7423	1.1446	0.6998	0.6316
ADT	1.4385	0.8000	0.3303	0.2536
SMO	1.5538	0.9278	0.4642	0.3923
IB1	1.7904	1.3329	0.9007	0.8469
IBK	1.5673	0.9431	0.4827	0.4115
PART	1.7904	1.2614	0.8245	0.7727
ONER	1.3635	0.7116	0.3002	0.2847
JRIP	1.3846	0.7350	0.3279	0.3230
RDR	1.4288	0.8562	0.3903	0.3852
J48	1.4962	0.9343	0.5520	0.4809
NBAYES	1.3788	0.7285	0.3210	0.2416
HPIPES	1.3769	0.8569	0.4550	0.3876
LWLS	1.6192	1.0037	0.5473	0.5526

Table 4.68: ECM Results for JM1-13C Models Applied to KC2 datasets, c=30

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	2.0577	1.5060	1.0924	1.0454
TD	2.0192	1.1745	0.4111	0.3373
LR	1.3827	0.7329	0.3233	0.2416
LOC	1.3250	0.6679	0.2540	0.1722
GP	1.3404	0.6854	0.2725	0.2608
LBOOST	1.3846	0.7350	0.2564	0.1722
RBM	1.4346	0.7913	0.3857	0.3852
BAG	1.3981	0.8153	0.4111	0.3349
RSET	1.5731	0.9496	0.5566	0.4809
MCOST	1.4885	0.8519	0.4503	0.3852
ABOOST	1.4423	0.8672	0.4688	0.3995
DTABLE	1.2904	0.6292	0.2125	0.1938
ADT	1.4404	0.8628	0.3949	0.3182
SMO	1.5942	0.9730	0.5127	0.4474
IB1	1.5385	0.9774	0.5889	0.5239
IBK	1.6481	1.0358	0.5820	0.5215
PART	1.4000	0.8175	0.4134	0.3469
ONER	1.3635	0.7116	0.3002	0.2847
JRIP	1.4038	0.7569	0.3487	0.2656
RDR	1.2904	0.6941	0.2818	0.2057
J48	1.4019	0.7547	0.3464	0.2656
NBAYES	1.3731	0.7219	0.3141	0.2368
HPIPES	1.7077	1.1008	0.7182	0.6675
LWLS	1.6019	1.0468	0.5935	0.5239

Table 4.69: ECM Results for JM1-8850 Models Applied to KC2 datasets, c=50

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	3.3558	2.7279	2.3095	2.1339
TD	2.3365	1.2688	0.8314	0.9593
LR	2.9346	2.0060	1.2979	1.0933
LOC	2.1077	1.0730	0.4088	0.2823
GP	2.0250	0.9801	0.3095	0.2919
LBOOST	2.5654	1.7009	1.0808	0.9952
RBM	2.6788	1.8310	1.1062	1.1268
BAG	2.8365	1.8223	1.4457	1.3660
RSET	3.1404	2.3549	1.6697	1.4689
MCOST	3.7000	3.1173	2.6074	2.5718
ABOOST	4.0635	3.7594	3.4134	3.2727
DTABLE	2.7865	1.9481	1.2333	1.1411
ADT	2.4077	1.5259	0.8984	0.7967
SMO	2.6481	1.6812	0.9469	0.8541
IB1	5.2250	4.7427	4.5704	4.7225
IBK	3.0346	2.3506	1.6697	1.5981
PART	3.6231	2.8165	2.3995	2.3541
ONER	2.7788	1.9481	1.2333	1.2584
JRIP	2.6788	1.9437	1.2286	1.1435
RDR	2.2673	1.5871	1.0831	0.9856
J48	3.3135	2.6929	2.1547	2.0981
NBAYES	2.1692	1.1419	0.4850	0.2584
HPIPES	3.0269	3.0026	2.3695	2.3325
LWLS	3.1481	2.4873	1.9307	1.9880

4.8.6 NECM Results for JM1 Models Applied to the KC2 Datasets, c=50

In this subsection, the predictive performance of the classification models built on the JM1 datasets and evaluated on the on the KC2 datasets are presented in Tables 4.69 to 4.74 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=50.

Table 4.70: ECM Results for JM1-4425 Models Applied to KC2 datasets, c=50

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	3.4499	2.9401	2.6536	2.4952
TD	2.2115	1.2601	0.7898	0.8086
LR	2.3769	1.3760	0.6212	0.5048
LOC	2.1077	1.0730	0.4088	0.2823
GP	2.0885	1.1594	0.5081	0.5120
LBOOST	2.2058	1.2962	0.6490	0.5359
RBM	2.1808	1.1551	0.4988	0.5000
BAG	2.3981	1.6254	1.2356	1.1411
RSET	2.5577	1.6921	0.9654	0.7464
MCOST	2.0558	1.2382	0.7044	0.7105
ABOOST	3.5058	3.0123	2.6143	2.4545
DTABLE	2.5865	1.6188	0.9931	0.8900
ADT	2.6577	1.8026	1.1917	1.1100
SMO	2.8385	1.8977	1.1778	0.9761
IB1	3.9558	3.5275	3.2794	3.2656
IBK	2.9481	2.0257	1.3118	1.2249
PART	2.4481	1.6998	1.4296	1.3397
ONER	2.5942	1.6254	1.0069	0.9019
JRIP	2.1192	1.3049	0.6582	0.6651
RDR	2.3692	1.6932	1.0716	0.9737
J48	2.7385	1.7873	1.1755	1.0933
NBAYES	2.3596	1.3563	0.7136	0.4928
HPIPES	3.5231	3.1207	2.4896	2.4498
LWLS	2.6692	1.9306	1.2148	1.1244

Table 4.71: ECM Results for JM1-23C Models Applied to KC2 datasets, c=50

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	3.4134	2.7903	2.3787	2.2105
TD	2.7731	0.9998	0.5473	0.4306
LR	2.0981	1.0621	0.3972	0.3900
LOC	2.1077	1.0730	0.4088	0.2823
GP	2.3096	1.3027	0.6536	0.5311
LBOOST	2.1365	1.1080	0.4457	0.4330
RBM	2.4846	1.6057	0.9815	1.0048
BAG	2.7808	1.9481	1.3487	1.1340
RSET	2.7558	1.8157	1.2079	1.1196
MCOST	3.1577	2.3812	1.7021	1.5072
ABOOST	3.4250	2.6886	2.1524	1.9737
DTABLE	2.5596	1.5969	0.9769	0.7560
ADT	2.4212	1.4285	0.7875	0.7871
SMO	2.5442	1.5663	0.8268	0.6148
IB1	3.6327	3.0419	2.5196	2.4737
IBK	2.5519	1.5751	0.8314	0.7273
PART	2.9538	2.1471	1.5658	1.3732
ONER	2.1058	1.1791	0.5219	0.3995
JRIP	2.9885	2.1843	1.6051	1.6459
RDR	2.4673	1.4799	0.7298	0.7416
J48	2.6038	1.8573	1.3718	1.2871
NBAYES	2.1846	1.1594	0.5035	0.2632
HPIPES	2.6577	1.6921	1.0693	0.8589
LWLS	2.4904	1.6232	1.0023	0.8971

Table 4.72: ECM Results for JM1-20C Models Applied to KC2 datasets, c=50

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	3.0327	2.3549	1.9146	1.8469
TD	3.5731	1.7118	0.7921	0.8014
LR	2.1058	1.0730	0.4088	0.3995
LOC	2.0250	0.9801	0.3095	0.2967
GP	2.0096	1.0708	0.4065	0.4019
LBOOST	2.7673	1.8223	1.0993	1.1220
RBM	2.4673	1.4777	0.8453	0.7440
BAG	2.9635	2.1515	1.4480	1.2464
RSET	2.4577	1.4668	0.8337	0.7273
MCOST	3.1538	2.3681	1.6836	1.6100
ABOOST	2.7404	2.0191	1.3187	1.1148
DTABLE	2.2827	1.3869	0.8637	0.7584
ADT	2.6635	1.8157	1.0924	0.9928
SMO	2.4673	1.4799	0.7298	0.6172
IB1	2.9346	2.2423	1.6744	1.5981
IBK	2.5692	1.7030	0.9746	0.8684
PART	3.4019	2.6579	2.0023	1.9450
ONER	2.1192	1.0862	0.4226	0.4139
JRIP	3.0308	2.2313	1.5358	1.3397
RDR	2.1173	1.3027	0.6536	0.5455
J48	3.1346	2.3528	1.7806	1.5957
NBAYES	2.3712	1.3694	0.7275	0.5000
HPIPES	2.9288	2.0126	1.3048	1.2225
LWLS	2.4769	1.7184	1.1085	0.8923

Table 4.73: ECM Results for JM1-17C Models Applied to KC2 datasets, c=50

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	3.2308	2.3593	1.7990	1.6052
TD	3.5750	1.7140	0.5242	0.1890
LR	2.2058	1.1835	0.4111	0.3995
LOC	2.0250	0.9801	0.3095	0.2967
GP	2.1942	1.1704	0.5127	0.3923
LBOOST	2.1154	1.0840	0.4226	0.4163
RBM	2.7481	1.4777	1.0670	0.8517
BAG	2.1981	1.2852	0.5196	0.3995
RSET	2.2885	1.4952	0.8614	0.7632
MCOST	2.5865	1.7271	0.9931	0.8900
ABOOST	2.7635	1.9240	1.2102	1.1172
DTABLE	2.7808	1.8376	1.1155	1.0144
ADT	2.2846	1.2765	0.5150	0.3971
SMO	2.4769	1.4909	0.7413	0.6316
IB1	2.8673	2.1559	1.4550	1.3732
IBK	2.4904	1.5062	0.7598	0.6507
PART	2.8673	2.0410	1.3326	1.2512
ONER	2.1327	1.1015	0.4388	0.4282
JRIP	2.1923	1.1682	0.5127	0.5144
RDR	2.2750	1.3760	0.6212	0.6244
J48	2.3808	1.4974	0.8753	0.7679
NBAYES	2.1865	1.1616	0.5058	0.3852
HPIPES	2.1462	1.3334	0.6859	0.5789
LWLS	2.5808	1.6101	0.8707	0.8876

Table 4.74: ECM Results for JM1-13C Models Applied to KC2 datasets, c=50

Methods	KC2-520	KC2-23C	KC2-17C	KC2-13C
CBR	3.3269	2.4589	1.7852	1.7153
TD	3.2885	1.9109	0.6420	0.5287
LR	2.1904	1.1660	0.5081	0.3852
LOC	2.0942	1.0577	0.3926	0.2679
GP	2.1096	1.0752	0.4111	0.4043
LBOOST	2.1923	1.1682	0.3949	0.2679
RBM	2.2808	1.2677	0.6166	0.6244
BAG	2.2058	1.2918	0.6420	0.5263
RSET	2.4962	1.5127	0.8799	0.7679
MCOST	2.3731	1.3716	0.7275	0.6244
ABOOST	2.2885	1.3869	0.7460	0.6388
DTABLE	2.0212	0.9757	0.3048	0.2895
ADT	2.2865	1.3826	0.6259	0.5096
SMO	2.5558	1.5794	0.8360	0.7344
IB1	2.4615	1.5838	0.9584	0.8589
IBK	2.6481	1.6855	0.9515	0.8565
PART	2.2077	1.2940	0.6443	0.5383
ONER	2.1327	1.1015	0.4388	0.4282
JRIP	2.2115	1.1901	0.5335	0.4091
RDR	2.0212	1.0840	0.4203	0.3014
J48	2.2096	1.1879	0.5312	0.4091
NBAYES	2.1808	1.1551	0.4988	0.3804
HPIPES	2.7462	1.7938	1.1801	1.0981
LWLS	2.5635	1.6965	0.9630	0.8589

Table 4.75: ECM Results for KC2-520 Models Applied to JM1 datasets, c=10

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	0.8688	0.6993	0.5813	0.4958	0.3892
TD	1.1027	0.8965	0.7881	0.6966	0.5516
LR	0.9137	0.7022	0.5594	0.4328	0.2741
LOC	0.9590	0.7434	0.5793	0.4257	0.2252
GP	0.9898	0.7795	0.6253	0.4836	0.3002
ANN	1.4840	1.3721	1.3259	1.2670	1.1691
LBOOST	0.9149	0.7360	0.6111	0.5169	0.4071
RBM	0.9363	0.7351	0.5949	0.4741	0.3138
BAG	0.9447	0.7611	0.6145	0.4932	0.3231
RSET	1.0315	0.8754	0.7550	0.6594	0.5129
MCOST	0.9105	0.7506	0.6416	0.5430	0.4211
ABOOST	1.0757	0.9348	0.8401	0.7678	0.6740
DTABLE	0.9099	0.7059	0.5533	0.4311	0.2638
ADT	0.9454	0.7488	0.6053	0.4793	0.3195
SMO	0.9280	0.7197	0.5762	0.4496	0.2925
IB1	0.9920	0.8724	0.8049	0.7848	0.6248
IBK	0.9478	0.7053	0.5806	0.4483	0.2914
PART	1.0134	0.8457	0.7108	0.5846	0.4237
ONER	0.9318	0.7122	0.5455	0.3960	0.1987
JRIP	0.9472	0.7496	0.6048	0.4787	0.3171
RDR	0.9472	0.7496	0.6048	0.4787	0.3171
J48	0.9472	0.7496	0.6048	0.4787	0.3171
NBAYES	0.9131	0.6950	0.5313	0.3905	0.2017
HPIPES	1.1041	0.9258	0.8063	0.7043	0.5530
LWLS	0.9800	0.7931	0.6482	0.5246	0.3727

4.8.7 NECM Results for KC2 Models Applied to the JM1 Datasets, c=10

In this subsection, the predictive performance of the classification models built on the KC2 datasets and evaluated on the on the JM1 datasets are presented in Tables 4.75 to 4.79 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=10.

Table 4.76: ECM Results for KC2-260 Models Applied to JM1 datasets, c=10

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	0.8696	0.7176	0.6034	0.5270	0.4306
TD	1.0553	0.7782	0.5951	0.4536	0.2683
LR	0.8922	0.6952	0.5578	0.4430	0.2962
LOC	0.9496	0.7326	0.5674	0.4165	0.2162
GP	0.9506	0.7963	0.6745	0.5802	0.4512
ANN	1.7985	1.7963	1.8074	1.7883	1.7396
LBOOST	0.9369	0.8157	0.7443	0.6873	0.6208
RBM	0.9155	0.7248	0.6020	0.5057	0.3898
BAG	0.9282	0.7722	0.6441	0.5351	0.3950
RSET	0.9606	0.7520	0.5985	0.4730	0.3000
MCOST	0.9476	0.7833	0.6603	0.5624	0.4279
ABOOST	1.0872	0.9822	0.8991	0.8292	0.7386
DTABLE	0.9469	0.7582	0.6301	0.5303	0.3897
ADT	1.0275	0.8849	0.7621	0.6613	0.5277
SMO	0.9894	0.8028	0.6769	0.5660	0.4234
IB1	1.0527	0.9016	0.8273	0.7940	0.6519
IBK	0.9416	0.7031	0.5592	0.4407	0.2764
PART	0.9845	0.8629	0.7807	0.7230	0.6431
ONER	1.0694	0.9474	0.8645	0.7964	0.6772
JRIP	0.9287	0.7513	0.6392	0.5409	0.4371
RDR	0.8583	0.6676	0.5411	0.4380	0.3223
J48	1.0797	1.0053	0.9713	0.9377	0.9167
NBAYES	0.9315	0.7147	0.5501	0.4062	0.2240
HPIPES	0.8654	0.6613	0.5212	0.4180	0.2829
LWLS	1.1652	1.0700	1.0156	0.9416	0.8342

Table 4.77: ECM Results for KC2-23C Models Applied to JM1 datasets, c=10

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	0.8850	0.7247	0.6142	0.5285	0.4174
TD	1.1096	0.8162	0.6098	0.4602	0.2869
LR	0.9844	0.7876	0.6447	0.5127	0.3449
LOC	0.9590	0.7434	0.5793	0.4257	0.2252
GP	1.0641	0.8684	0.7328	0.6092	0.4367
ANN	1.8355	1.8462	1.8623	1.8471	1.8044
LBOOST	1.0173	0.8657	0.7341	0.6135	0.4437
RBM	0.9376	0.7681	0.6550	0.5538	0.4159
BAG	0.9799	0.7845	0.6340	0.5097	0.3311
RSET	0.9016	0.6989	0.5517	0.4257	0.2634
MCOST	0.8818	0.6863	0.5456	0.4334	0.2887
ABOOST	0.9781	0.8061	0.6742	0.5724	0.4269
DTABLE	0.9750	0.7640	0.6077	0.4676	0.2777
ADT	0.9729	0.7896	0.6436	0.5182	0.3506
SMO	0.9080	0.6878	0.5285	0.3954	0.2247
IB1	1.0156	0.8478	0.8052	0.7708	0.6190
IBK	0.9428	0.6873	0.5431	0.4150	0.2190
PART	0.9998	0.8417	0.7369	0.6374	0.5041
ONER	0.9190	0.7188	0.5689	0.4466	0.2852
JRIP	0.9190	0.7188	0.5689	0.4466	0.2852
RDR	0.9282	0.7243	0.5754	0.4507	0.2847
J48	0.9998	0.8417	0.7369	0.6374	0.5041
NBAYES	0.9247	0.7098	0.5460	0.4046	0.2145
HPIPES	0.9068	0.6948	0.5373	0.4079	0.2382
LWLS	0.9773	0.7905	0.6195	0.5109	0.3070

Table 4.78: ECM Results for KC2-17C Models Applied to JM1 datasets, c=10

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	0.8571	0.6635	0.5296	0.4281	0.3020
TD	1.2027	0.8801	0.6326	0.4257	0.2162
LR	0.9383	0.7337	0.5796	0.4453	0.2751
LOC	0.9718	0.7579	0.5951	0.4432	0.2391
GP	0.9619	0.7640	0.6195	0.5044	0.3461
ANN	1.8177	1.8281	1.8426	1.8260	1.7811
LBOOST	0.9130	0.7133	0.5650	0.4407	0.2819
RBM	0.8919	0.6900	0.5493	0.4269	0.2644
BAG	0.9092	0.7032	0.5473	0.4128	0.2411
RSET	0.9718	0.7579	0.5951	0.4432	0.2391
MCOST	0.9132	0.7083	0.5569	0.4242	0.2518
ABOOST	0.9115	0.7099	0.5608	0.4309	0.2651
DTABLE	0.9423	0.7401	0.5904	0.4563	0.2836
ADT	0.9243	0.7115	0.5474	0.4068	0.2222
SMO	0.9151	0.7002	0.5553	0.4287	0.2681
IB1	0.9031	0.6925	0.5386	0.4343	0.2875
IBK	0.9485	0.7074	0.5625	0.4528	0.2556
PART	0.9440	0.7415	0.5927	0.4727	0.3098
ONER	0.9472	0.7496	0.6048	0.4787	0.3171
JRIP	0.9085	0.7052	0.5585	0.4308	0.2697
RDR	0.8931	0.6863	0.5266	0.3889	0.2139
J48	0.9249	0.7326	0.5889	0.4784	0.3265
NBAYES	0.9200	0.7016	0.5370	0.3949	0.2110
HPIPES	0.9145	0.7003	0.5441	0.4104	0.2322
LWLS	0.8615	0.6665	0.5216	0.4050	0.2669

Table 4.79: ECM Results for KC2-13C Models Applied to JM1 datasets, c=10

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	0.8800	0.6912	0.5583	0.4578	0.3215
TD	1.2157	0.9463	0.7365	0.5813	0.3657
LR	0.8860	0.7154	0.6166	0.5317	0.4228
LOC	0.9496	0.7326	0.5674	0.4165	0.2162
GP	0.9728	0.7696	0.6201	0.4792	0.3023
ANN	1.8433	1.8601	1.8777	1.8637	1.8227
LBOOST	0.9472	0.7496	0.6048	0.4787	0.3171
RBM	0.9298	0.7308	0.5921	0.4751	0.3240
BAG	0.9472	0.7496	0.6048	0.4787	0.3171
RSET	0.9529	0.7382	0.5795	0.4403	0.2481
MCOST	0.9472	0.7496	0.6048	0.4787	0.3171
ABOOST	0.9472	0.7496	0.6048	0.4787	0.3171
DTABLE	0.9472	0.7496	0.6048	0.4787	0.3171
ADT	0.9586	0.7587	0.6119	0.4834	0.3176
SMO	0.9124	0.6970	0.5497	0.4234	0.2608
IB1	0.9416	0.7325	0.5992	0.4863	0.3008
IBK	0.9627	0.7283	0.5910	0.4940	0.2929
PART	0.9472	0.7496	0.6048	0.4787	0.3171
ONER	0.9472	0.7496	0.6048	0.4787	0.3171
JRIP	0.9472	0.7496	0.6048	0.4787	0.3171
RDR	0.9472	0.7496	0.6048	0.4787	0.3171
J48	0.9472	0.7496	0.6048	0.4787	0.3171
NBAYES	0.8962	0.6814	0.5316	0.4000	0.2471
HPIPES	0.8963	0.6823	0.5203	0.3819	0.2076
LWLS	0.9511	0.7523	0.6062	0.4735	0.3098

Table 4.80: ECM Results for KC2-520 Models Applied to JM1 datasets, c=20

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.3185	1.0304	0.8357	0.6948	0.5172
TD	1.9569	1.5588	1.3517	1.1837	0.8892
LR	1.5012	1.1463	0.9122	0.6951	0.4171
LOC	1.6697	1.3172	1.0502	0.7845	0.4214
GP	1.7345	1.3917	1.1397	0.8937	0.5596
ANN	2.9032	2.7121	2.6344	2.5231	2.3316
LBOOST	1.4641	1.1634	0.9583	0.7989	0.6133
RBM	1.5623	1.2280	0.9955	0.7877	0.5017
BAG	1.5719	1.2733	1.0319	0.8249	0.5160
RSET	1.7468	1.4940	1.2905	1.1208	0.8538
MCOST	1.4201	1.1575	0.9789	0.8099	0.5940
ABOOST	1.8508	1.6253	1.4655	1.3362	1.1663
DTABLE	1.5032	1.1642	0.9103	0.7040	0.4068
ADT	1.6031	1.2815	1.0439	0.8261	0.5390
SMO	1.5156	1.1638	0.9276	0.7105	0.4372
IB1	1.6078	1.3935	1.2954	1.2869	0.9674
IBK	1.5625	1.1314	0.9460	0.7393	0.4427
PART	1.7321	1.4631	1.2392	1.0204	0.7281
ONER	1.5815	1.2167	0.9418	0.6855	0.3368
JRIP	1.6082	1.2848	1.0447	0.8271	0.5367
RDR	1.6082	1.2848	1.0447	0.8271	0.5367
J48	1.6082	1.2848	1.0447	0.8271	0.5367
NBAYES	1.5334	1.1699	0.8995	0.6589	0.3231
HPIPES	2.0069	1.7216	1.5231	1.3406	1.0552
LWLS	1.6399	1.3347	1.0881	0.8804	0.5989

4.8.8 NECM Results for KC2 Models Applied to the JM1 Datasets, c=20

In this subsection, the predictive performance of the classification models built on the KC2 datasets and evaluated on the on the JM1 datasets are presented in Tables 4.80 to 4.84 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=20.

Table 4.81: ECM Results for KC2-260 Models Applied to JM1 datasets, c=20

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.2911	1.0334	0.8423	0.7170	0.5570
TD	1.8971	1.3699	1.0238	0.7521	0.3963
LR	1.4459	1.1174	0.8909	0.6948	0.4376
LOC	1.6377	1.2807	1.0101	0.7512	0.3908
GP	1.5382	1.2879	1.0849	0.9240	0.6907
ANN	3.5906	3.5920	3.6149	3.5766	3.4791
LBOOST	1.4194	1.2161	1.0999	1.0024	0.8952
RBM	1.4759	1.1548	0.9505	0.7862	0.5861
BAG	1.4740	1.2214	1.0110	0.8231	0.5679
RSET	1.6419	1.2988	1.0468	0.8319	0.5179
MCOST	1.5442	1.2762	1.0721	0.9032	0.6624
ABOOST	1.8115	1.6432	1.5048	1.3810	1.2209
DTABLE	1.5492	1.2292	1.0180	0.8515	0.6042
ADT	1.7427	1.5202	1.3144	1.1363	0.8886
SMO	1.6504	1.3355	1.1280	0.9400	0.6945
IB1	1.7442	1.4727	1.3670	1.3142	1.0527
IBK	1.5450	1.0920	0.8909	0.7046	0.3928
PART	1.6003	1.4046	1.2656	1.1663	1.0223
ONER	1.8038	1.6148	1.4773	1.3574	1.1345
JRIP	1.4880	1.1915	1.0074	0.8394	0.6632
RDR	1.3205	0.9974	0.7899	0.6160	0.4221
J48	1.7215	1.5970	1.5434	1.4836	1.4538
NBAYES	1.5586	1.1960	0.9226	0.6761	0.3554
HPIPES	1.3479	1.0001	0.7671	0.5959	0.3660
LWLS	2.0206	1.8580	1.7732	1.6353	1.4362

Table 4.82: ECM Results for KC2-23C Models Applied to JM1 datasets, c=20

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.3404	1.0687	0.8869	0.7441	0.5555
TD	2.0215	1.4605	1.0708	0.7829	0.4532
LR	1.7042	1.3844	1.1479	0.9168	0.6110
LOC	1.6697	1.3172	1.0502	0.7845	0.4214
GP	1.9273	1.6155	1.3934	1.1731	0.8492
ANN	3.6660	3.6920	3.7245	3.6942	3.6088
LBOOST	1.7450	1.5126	1.2963	1.0810	0.7597
RBM	1.5049	1.2237	1.0387	0.8629	0.6188
BAG	1.6827	1.3659	1.1189	0.9032	0.5739
RSET	1.4632	1.1237	0.8805	0.6669	0.3815
MCOST	1.3869	1.0572	0.8239	0.6354	0.3852
ABOOST	1.6233	1.3439	1.1240	0.9478	0.6830
DTABLE	1.6993	1.3544	1.0982	0.8551	0.5089
ADT	1.6373	1.3403	1.0989	0.8816	0.5751
SMO	1.4979	1.1268	0.8630	0.6366	0.3394
IB1	1.6506	1.3856	1.3056	1.2699	0.9666
IBK	1.5824	1.1250	0.9071	0.7180	0.3487
PART	1.6777	1.4193	1.2471	1.0716	0.8300
ONER	1.5020	1.1680	0.9189	0.7120	0.4282
JRIP	1.5020	1.1680	0.9189	0.7120	0.4282
RDR	1.5305	1.1902	0.9436	0.7327	0.4394
J48	1.6777	1.4193	1.2471	1.0716	0.8300
NBAYES	1.5541	1.1963	0.9241	0.6805	0.3409
HPIPES	1.5079	1.1543	0.8929	0.6717	0.3729
LWLS	1.6699	1.3630	1.0862	0.8999	0.5282

Table 4.83: ECM Results for KC2-17C Models Applied to JM1 datasets, c=20

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.3057	0.9753	0.7530	0.5834	0.3702
TD	2.2682	1.6592	1.1836	0.7845	0.3908
LR	1.5903	1.2535	0.9985	0.7664	0.4597
LOC	1.7051	1.3573	1.0926	0.8292	0.4553
GP	1.6478	1.3249	1.0862	0.8874	0.6005
ANN	3.6313	3.6559	3.6852	3.6520	3.5623
LBOOST	1.4881	1.1548	0.9093	0.6986	0.4199
RBM	1.4388	1.1008	0.8697	0.6621	0.3759
BAG	1.4854	1.1409	0.8818	0.6526	0.3509
RSET	1.7051	1.3573	1.0926	0.8292	0.4553
MCOST	1.5098	1.1690	0.9195	0.6926	0.3865
ABOOST	1.5025	1.1681	0.9234	0.7024	0.4114
DTABLE	1.6191	1.2907	1.0486	0.8166	0.5014
ADT	1.5751	1.2249	0.9550	0.7144	0.3818
SMO	1.4925	1.1289	0.8898	0.6729	0.3928
IB1	1.4533	1.0724	0.8169	0.6619	0.3907
IBK	1.5858	1.1335	0.9237	0.7544	0.3853
PART	1.5880	1.2536	1.0087	0.8044	0.5127
ONER	1.6082	1.2848	1.0447	0.8271	0.5367
JRIP	1.4802	1.1403	0.8987	0.6811	0.3978
RDR	1.4649	1.1201	0.8555	0.6211	0.3153
J48	1.5271	1.2101	0.9726	0.7860	0.5160
NBAYES	1.5482	1.1842	0.9123	0.6693	0.3458
HPIPES	1.5179	1.1598	0.9011	0.6728	0.3569
LWLS	1.3406	1.0157	0.7830	0.5875	0.3484

Table 4.84: ECM Results for KC2-13C Models Applied to JM1 datasets, c=20

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.3693	1.0493	0.8295	0.6613	0.4263
TD	2.2892	1.7883	1.3844	1.0909	0.6717
LR	1.3606	1.0723	0.9117	0.7669	0.5807
LOC	1.6377	1.2807	1.0101	0.7512	0.3908
GP	1.6869	1.3562	1.1092	0.8637	0.5418
ANN	3.6840	3.7199	3.7554	3.7274	3.6454
LBOOST	1.6082	1.2848	1.0447	0.8271	0.5367
RBM	1.5332	1.2019	0.9716	0.7707	0.5036
BAG	1.6082	1.2848	1.0447	0.8271	0.5367
RSET	1.6399	1.2862	1.0250	0.7841	0.4361
MCOST	1.6082	1.2848	1.0447	0.8271	0.5367
ABOOST	1.6082	1.2848	1.0447	0.8271	0.5367
DTABLE	1.6082	1.2848	1.0447	0.8271	0.5367
ADT	1.6355	1.3080	1.0645	0.8423	0.5438
SMO	1.4910	1.1269	0.8842	0.6692	0.3855
IB1	1.5856	1.2254	1.0138	0.8346	0.4938
IBK	1.6136	1.1531	0.9761	0.8347	0.4525
PART	1.6082	1.2848	1.0447	0.8271	0.5367
ONER	1.6082	1.2848	1.0447	0.8271	0.5367
JRIP	1.6082	1.2848	1.0447	0.8271	0.5367
RDR	1.6082	1.2848	1.0447	0.8271	0.5367
J48	1.6082	1.2848	1.0447	0.8271	0.5367
NBAYES	1.4510	1.0870	0.8379	0.6126	0.3519
HPIPES	1.4793	1.1213	0.8520	0.6157	0.3090
LWLS	1.6223	1.3003	1.0573	0.8293	0.5327

Table 4.85: ECM Results for KC2-520 Models Applied to JM1 datasets, c=30

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.7682	1.3616	1.0901	0.8939	0.6453
TD	2.8112	2.2212	1.9152	1.6707	1.2268
LR	2.0888	1.5904	1.2649	0.9575	0.5601
LOC	2.3805	1.8909	1.5210	1.1434	0.6177
GP	2.4791	2.0040	1.6541	1.3038	0.8191
ANN	4.3224	4.0521	3.9429	3.7791	3.4941
LBOOST	2.0132	1.5908	1.3054	1.0808	0.8196
RBM	2.1882	1.7208	1.3961	1.1013	0.6897
BAG	2.1990	1.7854	1.4493	1.1567	0.7090
RSET	2.4620	2.1127	1.8260	1.5822	1.1947
MCOST	1.9297	1.5644	1.3162	1.0767	0.7670
ABOOST	2.6260	2.3159	2.0909	1.9047	1.6586
DTABLE	2.0964	1.6224	1.2673	0.9769	0.5498
ADT	2.2607	1.8141	1.4824	1.1729	0.7585
SMO	2.1032	1.6079	1.2790	0.9714	0.5819
IB1	2.2236	1.9146	1.7859	1.7891	1.3100
IBK	2.1772	1.5576	1.3115	1.0303	0.5940
PART	2.4507	2.0805	1.7677	1.4561	1.0324
ONER	2.2312	1.7211	1.3382	0.9750	0.4748
JRIP	2.2693	1.8200	1.4846	1.1754	0.7562
RDR	2.2693	1.8200	1.4846	1.1754	0.7562
J48	2.2693	1.8200	1.4846	1.1754	0.7562
NBAYES	2.1538	1.6448	1.2677	0.9273	0.4445
HPIPES	2.9097	2.5174	2.2399	1.9769	1.5575
LWLS	2.2998	1.8764	1.5280	1.2363	0.8250

4.8.9 NECM Results for KC2 Models Applied to the JM1 Datasets, c=30

In this subsection, the predictive performance of the classification models built on the KC2 datasets and evaluated on the on the JM1 datasets are presented in Tables 4.85 to 4.89 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=30.

Table 4.86: ECM Results for KC2-260 Models Applied to JM1 datasets, c=30

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.7125	1.3491	1.0813	0.9069	0.6833
TD	2.7389	1.9616	1.4524	1.0507	0.5244
LR	1.9995	1.5397	1.2240	0.9466	0.5789
LOC	2.3259	1.8288	1.4528	1.0859	0.5654
GP	2.1258	1.7795	1.4953	1.2678	0.9302
ANN	5.3827	5.3876	5.4223	5.3649	5.2187
LBOOST	1.9019	1.6166	1.4555	1.3176	1.1696
RBM	2.0364	1.5848	1.2991	1.0666	0.7823
BAG	2.0198	1.6706	1.3778	1.1111	0.7409
RSET	2.3233	1.8456	1.4952	1.1907	0.7357
MCOST	2.1408	1.7691	1.4839	1.2440	0.8969
ABOOST	2.5358	2.3042	2.1106	1.9329	1.7031
DTABLE	2.1514	1.7003	1.4059	1.1726	0.8187
ADT	2.4580	2.1556	1.8668	1.6113	1.2495
SMO	2.3114	1.8682	1.5792	1.3139	0.9656
IB1	2.4357	2.0439	1.9067	1.8344	1.4535
IBK	2.1484	1.4809	1.2226	0.9685	0.5092
PART	2.2162	1.9462	1.7505	1.6096	1.4015
ONER	2.5383	2.2822	2.0901	1.9183	1.5919
JRIP	2.0473	1.6318	1.3757	1.1380	0.8894
RDR	1.7826	1.3273	1.0387	0.7939	0.5219
J48	2.3633	2.1887	2.1154	2.0294	1.9910
NBAYES	2.1858	1.6773	1.2950	0.9460	0.4868
HPIPES	1.8304	1.3390	1.0131	0.7738	0.4492
LWLS	2.8759	2.6461	2.5307	2.3289	2.0383

Table 4.87: ECM Results for KC2-23C Models Applied to JM1 datasets, c=30

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.7957	1.4126	1.1595	0.9597	0.6935
TD	2.9333	2.1049	1.5318	1.1055	0.6195
LR	2.4240	1.9813	1.6510	1.3209	0.8771
LOC	2.3805	1.8909	1.5210	1.1434	0.6177
GP	2.7906	2.3625	2.0540	1.7370	1.2616
ANN	5.4965	5.5377	5.5868	5.5413	5.4133
LBOOST	2.4727	2.1595	1.8585	1.5484	1.0757
RBM	2.0721	1.6794	1.4223	1.1720	0.8217
BAG	2.3855	1.9474	1.6038	1.2967	0.8167
RSET	2.0247	1.5486	1.2094	0.9082	0.4996
MCOST	1.8920	1.4282	1.1022	0.8375	0.4816
ABOOST	2.2685	1.8817	1.5737	1.3233	0.9391
DTABLE	2.4236	1.9448	1.5888	1.2426	0.7401
ADT	2.3017	1.8909	1.5543	1.2450	0.7996
SMO	2.0877	1.5658	1.1975	0.8779	0.4542
IB1	2.2856	1.9234	1.8059	1.7690	1.3142
IBK	2.2219	1.5627	1.2711	1.0211	0.4785
PART	2.3557	1.9969	1.7573	1.5059	1.1560
ONER	2.0851	1.6173	1.2689	0.9774	0.5713
JRIP	2.0851	1.6173	1.2689	0.9774	0.5713
RDR	2.1328	1.6561	1.3119	1.0146	0.5940
J48	2.3557	1.9969	1.7573	1.5059	1.1560
NBAYES	2.1835	1.6827	1.3022	0.9564	0.4673
HPIPES	2.1090	1.6138	1.2485	0.9356	0.5076
LWLS	2.3626	1.9354	1.5528	1.2889	0.7494

Table 4.88: ECM Results for KC2-17C Models Applied to JM1 datasets, c=30

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.7542	1.2872	0.9765	0.7387	0.4384
TD	3.3338	2.4383	1.7345	1.1434	0.5654
LR	2.2423	1.7733	1.4173	1.0876	0.6443
LOC	2.4384	1.9567	1.5902	1.2152	0.6715
GP	2.3337	1.8858	1.5528	1.2704	0.8550
ANN	5.4449	5.4836	5.5278	5.4780	5.3434
LBOOST	2.0633	1.5963	1.2537	0.9564	0.5580
RBM	1.9856	1.5115	1.1902	0.8973	0.4873
BAG	2.0617	1.5786	1.2163	0.8923	0.4607
RSET	2.4384	1.9567	1.5902	1.2152	0.6715
MCOST	2.1064	1.6298	1.2821	0.9609	0.5212
ABOOST	2.0934	1.6264	1.2860	0.9738	0.5578
DTABLE	2.2959	1.8414	1.5068	1.1770	0.7193
ADT	2.2260	1.7383	1.3626	1.0220	0.5415
SMO	2.0699	1.5576	1.2243	0.9172	0.5175
IB1	2.0036	1.4523	1.0952	0.8896	0.4938
IBK	2.2231	1.5596	1.2849	1.0559	0.5151
PART	2.2321	1.7658	1.4247	1.1362	0.7156
ONER	2.2693	1.8200	1.4846	1.1754	0.7562
JRIP	2.0520	1.5754	1.2388	0.9314	0.5259
RDR	2.0366	1.5540	1.1844	0.8533	0.4168
J48	2.1294	1.6876	1.3563	1.0936	0.7056
NBAYES	2.1765	1.6668	1.2876	0.9438	0.4805
HPIPES	2.1212	1.6193	1.2580	0.9352	0.4816
LWLS	1.8197	1.3648	1.0444	0.7699	0.4299

Table 4.89: ECM Results for KC2-13C Models Applied to JM1 datasets, c=30

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	1.8585	1.4074	1.1008	0.8649	0.5310
TD	3.3626	2.6303	2.0323	1.6006	0.9777
LR	1.8351	1.4291	1.2069	1.0021	0.7387
LOC	2.3259	1.8288	1.4528	1.0859	0.5654
GP	2.4010	1.9428	1.5983	1.2482	0.7813
ANN	5.5246	5.5798	5.6332	5.5911	5.4682
LBOOST	2.2693	1.8200	1.4846	1.1754	0.7562
RBM	2.1366	1.6730	1.3511	1.0662	0.6832
BAG	2.2693	1.8200	1.4846	1.1754	0.7562
RSET	2.3269	1.8343	1.4706	1.1279	0.6240
MCOST	2.2693	1.8200	1.4846	1.1754	0.7562
ABOOST	2.2693	1.8200	1.4846	1.1754	0.7562
DTABLE	2.2693	1.8200	1.4846	1.1754	0.7562
ADT	2.3123	1.8574	1.5171	1.2011	0.7700
SMO	2.0695	1.5569	1.2187	0.9150	0.5102
IB1	2.2297	1.7183	1.4284	1.1829	0.6867
IBK	2.2644	1.5780	1.3612	1.1755	0.6122
PART	2.2693	1.8200	1.4846	1.1754	0.7562
ONER	2.2693	1.8200	1.4846	1.1754	0.7562
JRIP	2.2693	1.8200	1.4846	1.1754	0.7562
RDR	2.2693	1.8200	1.4846	1.1754	0.7562
J48	2.2693	1.8200	1.4846	1.1754	0.7562
NBAYES	2.0058	1.4926	1.1443	0.8252	0.4567
HPIPES	2.0624	1.5603	1.1837	0.8494	0.4104
LWLS	2.2934	1.8484	1.5085	1.1852	0.7555

Table 4.90: ECM Results for KC2-520 Models Applied to JM1 datasets, c=50

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	2.6677	2.0239	1.5989	1.2919	0.9014
TD	4.5197	3.5458	3.0424	2.6448	1.9020
LR	3.2640	2.4786	1.9705	1.4822	0.8462
LOC	3.8019	3.0384	2.4627	1.8611	1.0101
GP	3.9684	3.2285	2.6829	2.1241	1.3379
ANN	7.1608	6.7321	6.5599	6.2912	5.8191
LBOOST	3.1115	2.4456	1.9997	1.6448	1.2320
RBM	3.4402	2.7066	2.1972	1.7286	1.0655
BAG	3.4532	2.8097	2.2842	1.8201	1.0948
RSET	3.8925	3.3500	2.8970	2.5050	1.8766
MCOST	2.9489	2.3781	1.9909	1.6105	1.1129
ABOOST	4.1763	3.6970	3.3418	3.0416	2.6431
DTABLE	3.2828	2.5388	1.9813	1.5228	0.8359
ADT	3.5759	2.8795	2.3594	1.8666	1.1976
SMO	3.2783	2.4961	1.9817	1.4931	0.8713
IB1	3.4553	2.9569	2.7670	2.7933	1.9952
IBK	3.4066	2.4098	2.0423	1.6123	0.8967
PART	3.8880	3.3152	2.8246	2.3277	1.6411
ONER	3.5306	2.7299	2.1309	1.5540	0.7509
JRIP	3.5913	2.8905	2.3644	1.8720	1.1952
RDR	3.5913	2.8905	2.3644	1.8720	1.1952
J48	3.5913	2.8905	2.3644	1.8720	1.1952
NBAYES	3.3945	2.5947	2.0042	1.4641	0.6873
HPIPES	4.7154	4.1090	3.6735	3.2495	2.5619
LWLS	3.6195	2.9597	2.4079	1.9480	1.2774

4.8.10 NECM Results for KC2 Models Applied to the JM1 Datasets, c=50

In this subsection, the predictive performance of the classification models built on the KC2 datasets and evaluated on the on the JM1 datasets are presented in Tables 4.90 to 4.94 in terms of the Normalized Expected Cost of Misclassification (NECM) measure, at c=50.

Table 4.91: ECM Results for KC2-260 Models Applied to JM1 datasets, c=50

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	2.5555	1.9807	1.5591	1.2869	0.9361
TD	4.4225	3.1450	2.3098	1.6478	0.7805
LR	3.1069	2.3843	1.8902	1.4502	0.8616
LOC	3.7021	2.9249	2.3383	1.7554	0.9147
GP	3.3009	2.7627	2.3161	1.9554	1.4091
ANN	8.9669	8.9790	9.0372	8.9415	8.6978
LBOOST	2.8669	2.4175	2.1667	1.9478	1.7184
RBM	3.1573	2.4447	1.9962	1.6276	1.1748
BAG	3.1113	2.5691	2.1115	1.6871	1.0868
RSET	3.6860	2.9392	2.3918	1.9085	1.1715
MCOST	3.3340	2.7548	2.3075	1.9255	1.3659
ABOOST	3.9844	3.6262	3.3221	3.0366	2.6677
DTABLE	3.3559	2.6424	2.1817	1.8150	1.2478
ADT	3.8885	3.4263	2.9715	2.5612	1.9712
SMO	3.6334	2.9335	2.4815	2.0618	1.5077
IB1	3.8188	3.1862	2.9861	2.8748	2.2551
IBK	3.3551	2.2588	1.8860	1.4962	0.7421
PART	3.4478	3.0295	2.7203	2.4962	2.1598
ONER	4.0072	3.6171	3.3157	3.0401	2.5066
JRIP	3.1660	2.5123	2.1122	1.7351	1.3418
RDR	2.7069	1.9870	1.5362	1.1497	0.7214
J48	3.6469	3.3721	3.2595	3.1211	3.0654
NBAYES	3.4400	2.6400	2.0399	1.4858	0.7495
HPIPES	2.7954	2.0167	1.5050	1.1297	0.6155
LWLS	4.5867	4.2223	4.0458	3.7161	3.2423

Table 4.92: ECM Results for KC2-23C Models Applied to JM1 datasets, c=50

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	2.7065	2.1006	1.7049	1.3910	0.9696
TD	4.7571	3.3935	2.4538	1.7509	0.9521
LR	3.8635	3.1749	2.6573	2.1291	1.4093
LOC	3.8019	3.0384	2.4627	1.8611	1.0101
GP	4.5172	3.8565	3.3751	2.8649	2.0865
ANN	9.1575	9.2291	9.3113	9.2355	9.0221
LBOOST	3.9280	3.4533	2.9829	2.4833	1.7076
RBM	3.2066	2.5907	2.1897	1.7903	1.2275
BAG	3.7912	3.1103	2.5736	2.0838	1.3023
RSET	3.1479	2.3983	1.8672	1.3907	0.7357
MCOST	2.9021	2.1701	1.6587	1.2416	0.6745
ABOOST	3.5589	2.9573	2.4732	2.0742	1.4514
DTABLE	3.8722	3.1257	2.5698	2.0176	1.2024
ADT	3.6305	2.9922	2.4651	1.9718	1.2486
SMO	3.2673	2.4437	1.8665	1.3604	0.6837
IB1	3.5557	2.9990	2.8066	2.7672	2.0093
IBK	3.5010	2.4381	1.9992	1.6273	0.7379
PART	3.7116	3.1521	2.7777	2.3744	1.8079
ONER	3.2512	2.5157	1.9688	1.5081	0.8573
JRIP	3.2512	2.5157	1.9688	1.5081	0.8573
RDR	3.3373	2.5880	2.0483	1.5786	0.9034
J48	3.7116	3.1521	2.7777	2.3744	1.8079
NBAYES	3.4423	2.6556	2.0583	1.5083	0.7201
HPIPES	3.3113	2.5328	1.9597	1.4634	0.7770
LWLS	3.7479	3.0803	2.4860	2.0669	1.1918

Table 4.93: ECM Results for KC2-17C Models Applied to JM1 datasets, c=50

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	2.6514	1.9110	1.4234	1.0493	0.5748
TD	5.4649	3.9965	2.8364	1.8611	0.9147
LR	3.5462	2.8130	2.2550	1.7299	1.0135
LOC	3.9051	3.1556	2.5852	1.9872	1.1039
GP	3.7054	3.0076	2.4860	2.0363	1.3639
ANN	9.0720	9.1391	9.2129	9.1300	8.9057
LBOOST	3.2136	2.4794	1.9424	1.4721	0.8340
RBM	3.0794	2.3329	1.8311	1.3678	0.7101
BAG	3.2142	2.4540	1.8853	1.3718	0.6802
RSET	3.9051	3.1556	2.5852	1.9872	1.1039
MCOST	3.2997	2.5514	2.0073	1.4977	0.7906
ABOOST	3.2754	2.5428	2.0112	1.5166	0.8505
DTABLE	3.6496	2.9426	2.4232	1.8978	1.1550
ADT	3.5277	2.7651	2.1778	1.6372	0.8608
SMO	3.2247	2.4150	1.8933	1.4058	0.7670
IB1	3.1042	2.2122	1.6517	1.3450	0.7000
IBK	3.4976	2.4119	2.0073	1.6591	0.7745
PART	3.5202	2.7900	2.2568	1.7996	1.1214
ONER	3.5913	2.8905	2.3644	1.8720	1.1952
JRIP	3.1955	2.4456	1.9190	1.4320	0.7820
RDR	3.1801	2.4216	1.8422	1.3177	0.6197
J48	3.3339	2.6425	2.1237	1.7088	1.0848
NBAYES	3.4330	2.6320	2.0381	1.4926	0.7499
HPIPES	3.3280	2.5383	1.9720	1.4599	0.7311
LWLS	2.7779	2.0630	1.5673	1.1348	0.5929

Table 4.94: ECM Results for KC2-13C Models Applied to JM1 datasets, c=50

Methods	JM1-8850	JM1-23C	JM1-20C	JM1-17C	JM1-13C
CBR	2.8371	2.1236	1.6433	1.2720	0.7406
TD	5.5095	4.3143	3.3282	2.6199	1.5897
LR	2.7843	2.1427	1.7972	1.4726	1.0547
LOC	3.7021	2.9249	2.3383	1.7554	0.9147
GP	3.8293	3.1159	2.5765	2.0172	1.2603
ANN	9.2060	9.2994	9.3886	9.3185	9.1136
LBOOST	3.5913	2.8905	2.3644	1.8720	1.1952
RBM	3.3434	2.6151	2.1100	1.6573	1.0424
BAG	3.5913	2.8905	2.3644	1.8720	1.1952
RSET	3.7009	2.9304	2.3616	1.8154	0.9998
MCOST	3.5913	2.8905	2.3644	1.8720	1.1952
ABOOST	3.5913	2.8905	2.3644	1.8720	1.1952
DTABLE	3.5913	2.8905	2.3644	1.8720	1.1952
ADT	3.6660	2.9561	2.4222	1.9189	1.2224
SMO	3.2266	2.4169	1.8877	1.4065	0.7597
IB1	3.5179	2.7040	2.2576	1.8795	1.0725
IBK	3.5661	2.4277	2.1314	1.8571	0.9315
PART	3.5913	2.8905	2.3644	1.8720	1.1952
ONER	3.5913	2.8905	2.3644	1.8720	1.1952
JRIP	3.5913	2.8905	2.3644	1.8720	1.1952
RDR	3.5913	2.8905	2.3644	1.8720	1.1952
J48	3.5913	2.8905	2.3644	1.8720	1.1952
NBAYES	3.1154	2.3038	1.7571	1.2505	0.6662
HPIPES	3.2285	2.4382	1.8471	1.3168	0.6133
LWLS	3.6358	2.9446	2.4108	1.8969	1.2012

4.8.11 Discussion

One obvious pattern that can be seen in the results presented in this section is the improvement in the predictive accuracy as the quality of training dataset improves, for a given classification model. This is an indication that one needs to be vigilant of the quality of evaluation dataset when evaluating the classification models built. While it was hoped that as the quality of fit improves, for a given dataset, the predictive accuracy would improve, it was not the case.

Chapter 5

CONCLUSIONS

The empirical investigation reveals that the predictive performance of classification techniques improves as more and more (inherent) noise is removed. Use of relatively large number of classifiers, i.e., 25, provides certain degree of freedom and flexibility to explore different levels of filtering from most conservative to the least conservative to achieve the desired level of conservativeness while removing the instances suspect of being noisy. With twenty five base-level classifiers, it is highly unlikely for the noise elimination process to get influenced by predictions of a few classifiers which may not have the appropriate inductive bias for the dataset at hand. Thus, experimenting with relatively large number of classifiers to base the noise elimination process gives a higher level of confidence in the process.

The two case studies in Software Quality Classification presented here very closely approximate a real-world scenario, where appropriate noise-handling technique(s) need to be employed on a dataset with inherent noise. Normalized Expected Cost of Misclassification is used as a practical performance evaluation measure, taking the disparity between the two types of misclassification (very common in software

quality classification and many other domains) into account. Also, the datasets on which performance of different classifiers is evaluated are *noise-free*, as they are generated by impartially splitting the given dataset after noise removal. This gives a better insight into the *true* predictive performance.

Two-way ANOVA: Randomized Complete Block Design revealed that at significance level $\alpha = 5\%$, predictive performance of all the classification techniques is significantly different on the datasets for the JM1 system, but not for the KC2 system, which is a little surprising. The other finding that there was significant difference ($\alpha = 1\%$) between the datasets with different levels of noise filtering statistically confirmed our intuitive assumption that the classification performance would improve as more and more noise is eliminated.

It is evident from the Multiple Pairwise Comparison results that there is a lot of overlap between different clusters, and that classification methods performing well on a particular dataset may not necessarily perform as well on some other dataset(s), even if the datasets are from the same domain. And hence, in our opinion, basing the noise elimination procedure on a few selected (base-level) classification techniques may not be the most appropriate strategy.

Secondly, there may not be much to choose between different classifiers in terms of their performance, evinced by the big overlaps of clusters of classifiers for JM1 and (in most cases) no significant difference in predictive performance of classifiers for KC2. Trying to explore new classification techniques that may (not

necessarily) marginally improve the classification accuracy may not be worth the effort if the training data are noisy to begin with.

We also found that for the JM1 Software System, there is significant difference ($p < 0.01$) in the proportion of the noise removed by consensus filtering with 25 classifiers and consensus filtering with only 5 classifiers, suggesting that consensus filtering with relatively large number of classifier is more conservative than with a few classifiers. Similarly, the proportion of the noisy instances removed by consensus filter with 5 base-classifiers was significantly less than that with 3 base-classifiers. Similar observations were also made for the KC2 system. This indicates the change in the conservativeness at the consensus level of filtering as the number of base-classifiers changes.

While it was not the focus of our study to address the issue of exceptions, we feel that our most conservative level of filtering provides for handling exceptions to a certain degree, as it is likely that at least three out of the twenty five base-level classifiers can correctly classify the instances that are "hard-to-classify", or are "exceptions". However, further research is necessary to address this issue more directly in the context of noise elimination with the ensemble-classifier approach.

When the classification models build on the JM1 datasets were applied to KC2 datasets and vice versa, it was clear that for a given model, as the quality of test dataset improves, classification (predictive) accuracy improves.

Future work can address how to come up with a few representative base-classifiers to perform noise elimination based on the ensemble-classifier approach, maintaining the same degree of confidence in the noise elimination procedure. Use of Partition-based filtering Scheme can also be explored in conjunction with Ensemble-classifier filtering approach to reduce the number of base-classifiers required. A comprehensive comparative study of different noise handling techniques can also be carried out to facilitate selection of appropriate noise handling procedure.

BIBLIOGRAPHY

[1] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[3] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming: An Introduction On the Automatic Evolution of Computer Programs and its Application.* PWS Publishing Company, New York, 1998.

[4] J. Benediktsson and P. Swain. Consensus theoretic classification methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(4):668–704, 1992.

[5] M. L. Berenson, D. M. Levine, and M. Goldstein. *Intermediate Statistical Methods and Applications: A Computer Package Approach.* Prentice Hall, Englewood Cliffs, NJ, USA, 1983.

[6] M. Bobrowski, M. Marr, and D. Yankelevich. A software engineering view of data quality. citeseer.nj.nec.com/277636.html.

[7] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[8] L. Breiman. The heuristics of instability in model selection. *Machine Learning*, (24):2350–2383, 1996.

[9] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[10] C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Machine Learning*, 19:45–77, 1995.

[11] W. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.

[12] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[13] W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July 9-12 1995. Morgan Kaufmann.

[14] P. Compton and R. Jansen. Knowledge in context: a strategy for expert system maintenance. In C. J. Barter and M. J. Brooks, editors, *AI'88: 2nd Australian Joint Artificial Intelligence Conference*, pages 292–306, Adelaide, Australia, November 1990. Springer.

[15] P. Compton and R. Jansen. A philosophical basis for knowledge acquisition. 2(3):241–258, 1990.

[16] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[17] A. Danyluk and F. Provost. Small disjuncts in action: Learning to diagnose errors in the telephone network local loop. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 81–88, Amherst, MA, 1993. Morgan Kaufmann.

[18] T. DeMarco. *Controlling Software Projects.* Yourdon Press, New York, 1982.

[19] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[20] G. Drastal. Informed pruning in constructive induction. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 132–136, 1991.

[21] C. Ebert. Classification techniques for metric-based software development. *Software Quality Journal*, 5(4):255–272, 1996.

[22] J. English. Plain english on data quality, 1999.

[23] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* PWS Publishing Company, Boston, MA, 2nd edition edition, 1997.

[24] E. Frank, L. Trigg, G. Holmes, and I. H. Witten. Naive bayes for regression. *Machine Learning*, 41(1):5–25, 2000.

[25] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proc. 15th International Conf. on Machine Learning*, pages 144–151. Morgan Kaufmann, San Francisco, CA, 1998.

[26] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proc. 16th International Conference on Machine Learning*, pages 124–133, Bled, Slovenia, 1999. Morgan Kaufmann, San Francisco, CA.

[27] Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, pages 771–780, 1999.

[28] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.

[29] J. Friedman, J. Stochastic, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1999.

[30] B. R. Gaines and P. Compton. Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems*, 5(3):211–228, 1995.

[31] D. Gamberger, N. Lavrač, and S. Džeroski. Noise elimination in inductive concept learning: a case study in medical diagnosis. In *Algorithmic Learning Theory, 7th International Workshop, ALT '96, Sydney, Australia, October 1996, Proceedings*, volume 1160, pages 199–212. Springer, 1996.

[32] D. Gamberger, N. Lavrač, and C. Grošelj. Experiments with noise filtering in a medical domain. In *Proc. 16th International Conf. on Machine Learning*, pages 143–151. Morgan Kaufmann, San Francisco, CA, 1999.

[33] D. Gamberger and N. Lavrač. Conditions for occam's razor applicability and noise elimination. In *European Conference on Machine Learning*, pages 108–123, 1997.

[34] E. Geleyn. Combining decision trees for software quaity classification: An emprical study. Master's thesis, Florida Atlantic University, Boca Raton, FL, USA, May 2002. Advised by Taghi M. Khoshgoftaar.

[35] I. Guyon, N. Matic, and V. Vapnik. Discovering informative patterns and data cleaning. In *Advances in Knowledge Discovery and Data Mining*, pages 181–203. 1996.

[36] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[37] A. J. Hayter. A proof of the conjecture that tukey-kramer method is conservative. *The Annals of Statistics*, 12:61–75, 1984.

[38] J. Hipp, U. Güntzer, and U. Grimmer. Data quality mining - making a virtue of necessity. In *Proceedings of the 6th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD 2001)*, pages 52–57, Santa Barbara, California, May 20 2001.

[39] R. Holte, L. Acker, and B. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989. Morgan Kaufmann.

[40] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[41] S. I. Inc. *SAS/STAT User's Guide.* Cary, NC, USA, 4th edition, 1990. Volume 2.

[42] G. H. John. Robust decision trees: Removing outliers from data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 174–179, Montreal, Quebec, 1995. AAAI Press.

[43] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, Englewood Cliffs, NJ, USA, 2nd edition, 1992.

[44] W. D. Jones, J. P. Hudepohl, T. M. Khoshgoftaar, and E. B. Allen. Application of a usage profile in software quality models. In *Proceedings of the Third European Conference on Software Maintenance and Reengineering*, pages 148–157, Amsterdam, Netherlands, Mar. 1999. IEEE Computer Society.

[45] G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.

[46] T. M. Khoshgoftaar and E. B. Allen. Logistic regression modeling of software quality. *International Journal of Reliability, Quality and Safety Engineering*, 6(4):303–317, Dec. 1999.

[47] T. M. Khoshgoftaar and E. B. Allen. A practical classification rule for software quality models. *IEEE Transactions on Reliability*, 49(2):209–216, June 2000.

[48] T. M. Khoshgoftaar and E. B. Allen. Controlling overfitting in classification tree models of software quality. *Empirical Software Engineering*, 6(1):59–79, 2001.

[49] T. M. Khoshgoftaar, E. B. Allen, and J. Deng. Using regression trees to classify fault-prone software modules. *IEEE Transactions on Reliability*, 51(4):455–462, 2002.

[50] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. P. Hudepohl. Classification tree models of software-quality over multiple releases. *IEEE Transactions on Reliability*, 49(1):4–11, 2000.

[51] T. M. Khoshgoftaar and L. A. Bullard. A survey of boosting algorithms. Technical Report TR-CSE-02-09, Florida Atlantic University, 2001.

[52] T. M. Khoshgoftaar and N. Seliya. Software quality classification modeling using the SPRINT decision tree algorithm. In *14th International Conference on Tools with Artificial Intelligence*, pages 365–374, Washington, DC, USA, 2002.

[53] T. M. Khoshgoftaar and N. Seliya. Comparative assessment of software quality classification techniques: An empirical case study. *Empirical Software Engineering*, in press. Kluwer Academic Publishers.

[54] T. M. Khoshgoftaar, X. Yuan, E. B. Allen, W. D. Jones, and J. P. Hudepohl. Uncertain classification of fault-prone software modules. *Empirical Software Engineering*, 7(4):297–318, December 2002.

[55] R. Kohavi. The power of decision tables. In N. Lavrač and S. Wrobel, editors, *Proceedings of the European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, pages 174–189. Springer Verlag, 1995.

[56] R. Kohavi and C. Kunz. Option decision trees with majority votes. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 161–169, 1997.

[57] J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, Inc., San Mateo, California USA, 1993.

[58] J. Komorowski, L. Polkowski, and A. Skowron. *Rough Set: A Tutorial.* Springer-Verlag, 1998.

[59] J. R. Koza. *Genetic Programming*, volume I. MIT Press, New York, 1992.

[60] C. Y. Kramer. Extension of multiple range tests to group means with unequal number of replications. *Biometrics*, 12:307–310, 1956.

[61] K. C. Laudon. Data quality and the due process in large interorganizational record systems. *Communications of the ACM*, 29(1):4–11, 1986.

[62] N. Lavrač and D. Gamberger. Saturation filtering for noise and outlier detection.

[63] D. B. Leake, editor. *Case-Based Reasoning: Experience, Lessons, and Future Directions.* MIT Press, Cambridge, MA USA, 1996.

[64] D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 148–156, New Brunswick, NJ, 1994. Morgan Kaufmann.

[65] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, 1993.

[66] C. T. Lin and C. S. G. Lee. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems.* Prentice Hall Inc., Upper Saddle River, NJ USA, 1996.

[67] R. P. Lippmann. An introduction to computing with neural networks. *Accoustics, Speech and Signal Processing Magazine*, 4(2):4–22, 1987.

[68] W. Mao. Classification of software quality using tree modeling with the sprint-sliq algorithm. Master's thesis, Florida Atlantic University, Boca Raton, Florida USA, May 2000. Advised by Taghi M. Khoshgoftaar.

[69] A. Marcus and J. I. Maletic. Utilizing association rules for identification of possible errors in data sets. Technical Report, CS-00-03, Division of Computer Science, The University of Memphis.

[70] R. C. Morey. Estimating and improving the quality of information in MIS. *Communications of the ACM*, 25(5):337–342, 1982.

[71] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied Linear Statistical Models.* 1996.

[72] C. G. Nevill-Manning, G. Holmes, and I. H. Witten. The development of holte's 1r classifier. In *Proc. Artificial Neural Networks and Expert Systems*, pages 239–242, Dunedin, NZ, 1995.

[73] R. H. Nielsen. Counter propagation network. *Applied Optics Journal*, 26(23), 1987.

[74] N. Ohlsson, M. Helander, and C. Wohlin. Quality improvement by identification of fault-prone modules using software design metrics. In *International Conference on Software Quality*, pages 1–13, Ottawa, Ontario, Canada, 1996.

[75] N. Ohlsson, M. Zhao, and M. Helander. Application of multivariate analysis for software fault prediction. *Software Quality Journal*, 7(1):51–66, 1998.

[76] N. Oka and K. Yoshida. Learing regular and irregular examples separately. In *Proceedings of the 1993 IEEE International Joint Conference on Neural Networks*, pages 171–174. IEEE Press, 1993.

[77] N. Oka and K. Yoshida. A noise-tolerant hybrid model of a global and a local learning model. In *Proceedings of the AAAI-96 Workshop: Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, pages 95–110. AAAI Press, 1996.

[78] K. Orr. Data quality and systems theory. *CACM*, 41(2):66–71, February 1998.

[79] D. B. Owen. *Data Quality Control: Theory and Pragmatics.* Marcel Dekker, New York, N.Y., 1990.

[80] J. Peng, F.Ertl, S. Bhagotra, A. Mosam, N. Vijayaratnam, and I. Kanwal. Classification of U.S. census data. Data Mining Project CS4TF3.

[81] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report 98-14, Microsoft Research, Redmond, Washington, April 1998.

[82] J. C. Platt. *Advances in Kernel Methods - Support Vector Training*, chapter 12, pages 185–208. MIT Press, 1999.

[83] V. Ponnuswamy. Classification of software quality with tree modeling using C4.5 algorithm. Master's thesis, Florida Atlantic University, Boca Raton, FL, 2001. Advised by T. M. Khoshgoftaar.

[84] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[85] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[86] J. R. Quinlan. Baggin, boosting and c4.5. In *Proc. of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

[87] T. Redman. The impact of poor data quality on the typical enterprise. *CACM*, 41(2):79–82, February 1998.

[88] L. T. Reinwald and R. M. Soland. Conversion of limited-entry decision tables to optimal computer programs i: Minimum average processing time. *Journal of the ACM*, 13(3):339–358, 1966.

[89] J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.

[90] F. D. Ross. An empirical study of analogy based software quality classification models. Master's thesis, Florida Atlantic University, Boca Raton, FL USA, August 2001. Advised by T. M. Khoshgoftaar.

[91] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, 1987.

[92] D. E. Rumelhart, G. E. Hinton, and R. Williams. *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, MA, USA, 1962.

[93] H. Scheffe. *The Analysis of Variance.* John Wiley & Sons, New York, 1959.

[94] N. F. Schneidewind. Software metrics validation: Space shuttle flight software example. *Annals of Software Engineering*, 1:287–309, 1995.

[95] D. Skalak and E. Rissland. Inductive learning in a mixed paradigm setting. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 840–847, Boston, MA, 1990. Morgan Kaufmann.

[96] A. Srinivasan, S. Muggleton, and M. Bain. Distinguishing exceptions from noise in non-monotonic learning. In *Proceedings of the Second Inductive Logic Programming Workshop*, pages 97–107, Tokyo, Japan, 1992.

[97] D. M. Strong, Y. W. Lee, and R. Y. Wang. Data quality in context. *Communications of the ACM*, 40(5):103–110, 1997.

[98] G. K. Tayi. Data quality management. Research Seminar at Universidad de Buenos Aires, July 22 1998.

[99] G. K. Tayi and D. P. Ballou. Examining data quality. *Communications of the ACM*, 41(2):54–57, 1998.

[100] C. M. Teng. Correcting noisy data. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 239–248, 1999.

[101] C. M. Teng. Evaluating noise correction. In *Lecture notes in Artificial Intelligence: Proceedings of the Sixth Pacific Rim International Conference on Artificial Intelligence*. Springler-Verlag, 2000.

[102] C. M. Teng. A comparison of noise handling techniques. In *Proceedings of the International Florida Artificial Intelligence Research Symposium*, pages 269–273, 2001.

[103] I. Tomek. An experiment with edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):448–452, 1976.

[104] R. A. Waller and D. B. Duncan. A bayes rule for the symmetric multiple comparison problem. *Journal of the American Statistical Association*, 64:1484–1499, 1969.

[105] Y. Wand and R. Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11), November 1996.

[106] R. Wang, D. Strong, and L. Guarascio. A framework for analysis of data quality research. *Journal of Management Information Systems*, 12(4):5–34, Spring 1996.

[107] R. Y. Wang, V. C. Storey, and C. P. Firth. A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):623–639, August 1995.

[108] S. Weisberg. *Applied Linear Regression.* John Wiley & Sons, 1985.

[109] D. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2:408–421, 1972.

[110] D. R. Wilson and T. R. Martinez. Instance pruning techniques. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 404–411, Nashville, TN, 1997. Morgan Kaufmann.

[111] D. R. Wilson and T. R. Martinez. Reduction techniques for exemplar-based learning algorithm. *Machine Learning - in press*, 1999.

[112] P. H. Winston. Learning structural descriptions from examples. In P. H. Winston, editor, *The Psychology of Computer Vision*, New York, 1975. McGraw-Hill.

[113] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2000.

[114] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[115] J. H. Zar. *Biostatistical Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1984.

[116] Q. Zhao and T. Nishida. Using qualitative hypotheses to identify inaccurate data. *Journal of Artificial Intelligence Research*, 3:119–145, 1995.

[117] X. Zhu, X. Wu, and Q. Chen. Eliminating class noise in large datasets. In *Proceedings of the Twentieth International Conference on Machine Learning*, Washington, DC, 2003.